INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Cloud Contact Centre - IVR 2.0

JOEL ALEXANDRE CAMPOS TEIXEIRA
Outubro de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Cloud Contact Centre - IVR 2.0

Joel Teixeira

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

Orientador: Dr. Jorge Coelho

Dedicatória

Para todos aqueles que sofrem de insónias!

Resumo

Nesta dissertação é apresentado o desenvolvimento de um sistema de IVR. No contexto em que este projeto foi desenvolvido, o objetivo do uso do IVR é classificar e afunilar os visitantes recorrendo a estratégias de personalização. No contexto dos *contact centers*, os IVRs são utilizados com o objetivo de classificar e afunilar os chamadores.

Com os progressos tecnológicos, o reconhecimento e síntese de voz ganharam um novo estatuto para a tecnologia de IVR, existindo já alguns sistemas de IVR a suportar este tipo de tecnologia.

Surgindo no início da década de 1950, tecnologias de reconhecimento e síntese de voz foram alvo de grande investigação durante o resto do século XX, mas apenas tornaram-se viáveis em termos financeiros no início do século XXI. Com o aparecimento da computação na nuvem, o reconhecimento e síntese de voz passou a estar disponível a qualquer momento demonstrando um custo de implementação baixo para os benefícios que proporciona.

Com a integração das tecnologias de reconhecimento e síntese de voz, no sistema de IVR abre-se a possibilidade de tornar a jornada do chamador pessoal e única.

Nesta dissertação são apresentadas as várias fases do desenvolvimento do sistema de IVR, que consistiu na investigação e análise de tecnologias existentes, enumeração dos requisitos funcionais e não funcionais, desenho da arquitetura, implementação e avaliação do sistema.

No final da dissertação é feita uma apreciação sobre os objetivos definidos e o trabalho desenvolvido, assim como a enumeração das limitações e planos futuros para diminuir as limitações do sistema.

Palavras-chave: IVR, síntese de voz, reconhecimento de voz, automação, marketing

Abstract

In this dissertation the development of an IVR system is described. In the context in which this project was developed, the purpose of using IVR is to classify and narrow the visitors using customization strategies. In the context of contact centers, IVRs are used for the purpose of classifying and narrowing down callers.

With technological advances, voice recognition and synthesis have gained a new status for IVR technology, and some IVR systems already operate with this type of technology.

Appearing in the early 1950s, speech recognition and synthesis technologies were the subject of extensive research during the rest of the 20th century, but only became financially sustainable in the early 21st century. With the emergence of cloud computing, speech recognition and synthesis became available at any time showing a low implementation cost for the benefits it provides.

With the integration of speech recognition and synthesis technologies, in the IVR system it opens up the possibility of making the caller's journey personal and unique.

In this dissertation the various phases of the development of the IVR system are presented, which consisted of research and analysis of existing technologies, elicitation of functional and non-functional requirements, architecture design, implementation and evaluation of the system.

At the end of the dissertation an assessment is made on the defined objectives and the work accomplished, as well as listing the limitations and future plans to diminish the limitations of the system.

Agradecimentos

Gostaria de agradecer à BySide pela oportunidade e disponibilidade para desenvolver este projeto, em especial ao Daniel Ramos e ao Vítor Vidal pelo acompanhamento e pelo apoio dado na realização da dissertação. Agradecer o apoio dos meus colegas de equipa, Eduardo Vasco, Filipe Moutinho e Mário Sousa.

Agradecer aos meus amigos, em especial ao Alexandre Abreu e ao João Dias pela ajuda e apoio na revisão da dissertação. Agradecer à Margarida Guerra e à Rita Sousa pelo apoio moral e pelas batalhas e desafios conjuntos que partilhamos na realização da dissertação.

Por último agradecer ao meu orientador, professor Jorge Coelho, por todo o apoio prestado durante a elaboração desta dissertação.

Conteúdo

Glossário

ACD	Automatic Call Distribution.
AHP	Analytic Hierarchy Process.
API	Application Programming Interface.
B2BUA	Back to Back User Agent.
CD	Continuous Delivery.
CI	Continuous Integration.
CPU	Central Processing Unit.
CPV	Customer Perceived Value.
DTMF	Dual-Tone Multi Frequency.
DTO	Data Transfer Object.
FFE	Fuzzy Front End.
HTTP	Hypertext Transfer Protocol.
IAX	Inter-Asterisk Exchange Protocol.
IP	Internet Protocol.
IVR	Interactive Voice Response.
JSON	JavaScript Object Notation.
MB	Megabyte.
MVC	Model View Controller.
MVP	Minimum Viable Product.
NCD	New Concept Development.
NLP	Natural Language Processing.
NPD	New Product Development.
PBX	Private Branch Exchange.
PSTN	Public Switched Telephone Network.
REST	Representational State Transfer.
ROI	Return On Investment.
RTP	Real-Time Protocol.
SBC	Session Border Controller.
SEO	Search Engine Optimization.

SIP	Session Initiation Protocol.
SMS	Short Message Service.
SSML	Speech Synthesis Markup Language.
TCP	Transmission Control Protocol.
TSG	Technology Stage Gate.
UDP	User Datagram Protocol.
URL	Uniform Resource Locator.
VM	Virtual Machine.

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este capítulo tem como objetivo introduzir e contextualizar a presente dissertação, com o intuito de fornecer uma possível solução para o problema que tenta resolver. A secção de introdução está dividida em 5 partes: contexto, problema, objetivos, abordagem e estrutura do documento. A parte do contexto tem como objetivo enquadrar a dissertação no ambiente onde esta foi desenvolvida. A parte do problema descreve os desafios e constrangimentos que estão na origem do desenvolvimento desta dissertação. A parte dos objetivos define todas as metas que se espera cumprir durante o desenvolvimento da dissertação. A parte da abordagem tem como objetivo apresentar o processo de desenvolvimento do projeto que foi produzido nesta dissertação. Por último, a parte da estrutura do documento permite avaliar o que é explorado em cada capítulo da dissertação.

1.1 Contexto

A BySide é uma empresa de marketing digital, de média dimensão e que está no mercado há 15 anos. O seu foco é desenvolver estratégias que permitam qualificar, priorizar e converter uma *lead*. Este processo deu origem a que a BySide fosse a primeira empresa no mercado do marketing digital a criar e implementar o conceito de *lead activation*.

Lead activation [1] é a capacidade de maximizar as receitas dos visitantes que já se encontram num website, numa aplicação ou numa *landing page* e concentrar-se na qualidade das *leads* e na melhoria do retorno do investimento.

A empresa possui uma plataforma com dois produtos base: o Marketing Cloud e Cloud Contact Centre.

O Marketing Cloud é um produto que permite angariar e qualificar *leads*, construir experiências personalizadas através de segmentação para fomentar o aumento de audiências.

O Cloud Contact Centre é um produto que permite criar um contacto mais próximo com o visitante, permitindo ao mesmo entrar em contacto através de diversos canais (e.g. voz, chat, video, e-mail e SMS) com os agentes de *contact center*.

O foco do Cloud Contact Centre é otimizar a forma como os agentes de contact center executam o seu trabalho, proporcionando dados sobre a jornada do cliente no website. A jornada permite aos agentes visualizarem quais produtos ou serviços que o visitante está interessado. Com este tipos de dados os agentes são mais eficazes e eficientes em recomendar e esclarecer qualquer das dúvidas do visitante. Como informação é poder, aqui é dado o poder ao agente de conseguir vender o melhor produto com base nas preferências

do visitante. Este grau de personalização contribui para uma maior taxa de conversão e aumento do ROI (*Return On Investment*).

Uma vez que o sistema antigo não corresponde às novas necessidades do mercado, tanto em termos tecnológicos como em termos de sustentabilidade para o negócio, surge a necessidade de aumentar as capacidades do Cloud Contact Center com a inclusão de um novo sistema de IVRs.

1.1.1 IVR

O *Interactive Voice Response* ou IVR [2] é um sistema de voz desenhado para atender de forma automática os chamadores, através de mensagens automáticas, selecção de dígitos no teclado do telemóvel ou reconhecimento de voz.

Como forma de contribuir para o aumento dos resultados das vendas e da satisfação dos visitantes, a introdução de um IVR [3] pode reduzir eficazmente os tempos de espera e otimizar a alocação dos agentes. Apostando na classificação do motivo do contacto por parte do visitante, é possível encaminhar o mesmo para o grupo de agentes mais qualificados para responder às necessidade do visitante.

1.2 Problema

Tornando-se popular na década de 1970, o uso da tecnologia de IVR [4] era complexa e exigia um custo acrescido para os *call centers*. A primeira aplicação de IVR foi desenhada e desenvolvida por Steven Schmidt em 1973, com o principal objetivo de controlar o armazenamento de produtos. Durante esta década o sistema estava limitado à transmissão de algumas palavras, dificultando a transmissão das informações às pessoas que utilizavam estes sistemas.

Na década de 1980, existiu uma evolução na tecnologia de armazenamento de dados [4], o que tornou o armazenamento dos conteúdos digitais mais suportável para as organizações. Isto motivou a que a voz das pessoas começasse a ser a guardada digitalmente, permitindo o uso de mais palavras no IVR.

Na década de 1990, as organizações começaram a investir mais nas tecnologias de telefonia [4]. Com isto a tecnologia de IVR ganhou um maior número de funcionalidades (e.g. encaminhamento de chamadas, mecanismo de filas de espera). As novas funcionalidades permitiram que os *contact centers* pudessem diminuir os seus custos, uma vez que algumas das tarefas habituais dos agentes foram automatizadas. Se configurados, os IVRs permitiam recolher informação sobre o chamador, com objetivo de o segmentar. O uso da segmentação permitia escolher o caminho mais indicado para o chamador. Caso o chamador necessitasse de ajuda, ele seria encaminhado para agentes qualificados para atender os pedidos para o qual o chamador foi segmentado. Isto possibilitou uma grande otimização dos recursos humanos do *contact center*.

Na década de 2000, devido ao aumento da capacidade de processamento [4], tornou-se mais comum as soluções de reconhecimento e de síntese de voz estarem presentes nas soluções de IVR. O que neste momento torna a inclusão de reconhecimento e síntese de voz uma funcionalidade com extrema importância na tecnologia de IVR.

A evolução da tecnologia faz com que o mercado em torno da mesma seja muito competitivo, no entanto a solução de IVRs da BySide não acompanhou a evolução tecnológica deste mercado.

As principais melhorias a implementar na solução de IVRs da BySide são:

- Criação de uma configuração fácil e uniforme, para que qualquer pessoa consiga configurar um IVR;
- Permitir a integração com motores de NLP e algoritmos de *machine learning* para facilitar navegação nos *workflows*;
- Criação de um modelo estatístico e definição de métricas que permitam avaliar a construção do fluxo de IVR.

1.3 Objetivos

Este projeto apresenta os seguintes desafios tecnológicos e de aprendizagem individual:

- Investigação da integração de tecnologias de reconhecimento e síntese de voz com o mundo da telefonia;
- Investigação sobre *workflows* que permitam automatizar processos utilizando tecnologias VoIP (e.g. IVR, encaminhamento inteligente de chamadas, geração de estatísticas e métricas);
- Otimização do custo temporal para a construção de um fluxo de IVR, através da análise de requisitos, para encontrar uma configuração uniforme entre todos os fluxos de IVR.

A nível académico o principal objetivo é desenhar e implementar uma solução que cumpra as boas práticas de engenharia de software - nomeadamente em termos de integração de sistemas, qualidade e organização de software - com base nos conhecimentos adquiridos ao longo do mestrado.

A expectativa é que no final do projeto, tenha aprendido mais sobre as tecnologias VoIP, tecnologias de reconhecimento e síntese de voz e ganho uma maior sabedoria na aplicação de boas práticas de engenharia de software.

1.4 Abordagem

Neste projeto é adotada uma metodologia ágil e incremental. O objetivo passa por entregar o maior valor possível em espaços de tempo curtos. A framework Scrum foi a *framework* adotada pela equipa de desenvolvimento em que este projeto está enquadrado.

Seguindo a *framework* do Scrum, todas as funcionalidades deste projeto deram origem a *user stories*. Na BySide a *framework* do Scrum é utilizada da seguinte maneira:

1. A *user story* passa pelo processo de definição;
2. É apresentada à equipa para ser estimado o esforço que a mesma terá no seu desenvolvimento;
3. Com base na prioridade, é ela candidata a entrar no próximo *sprint*, durante a sessão de planeamento;

4. Ela começa a ser desenvolvida;
5. Passa por *code review*;
6. Passa para *quality assurance*;
7. É exportada para produção numa *release*;
8. Finalmente é considerada como completada.

No final do sprint será feita uma revisão de todas as tarefas finalizadas, seguindo-se de uma reflexão sobre a performance da equipa durante o sprint, de forma a encontrar estratégias de melhoria (*sprint retrospective*).

Depois deste ciclo, é feita uma nova iteração, ou seja é criado um novo *sprint*, novas *user stories* serão realizadas e o processo vai se repetindo ao longo do tempo.

Ao longo dos *sprints* é feito um ponto de situação para validar se o desenvolvimento está alinhado com as necessidades da equipa de produto, com o objetivo de detetar e realizar as alterações necessárias o mais cedo possível.

1.5 Estrutura da dissertação

Nesta secção é feita uma breve apresentação sobre a estrutura da dissertação. Em cada subsecção é resumido o conteúdo de cada capítulo que constitui esta dissertação.

1.5.1 Introdução

Este capítulo tem como objetivo dar uma visão geral sobre o problema a ser resolvido, em que contexto se enquadra, quais os objetivos a serem cumpridos e apresentar a abordagem seguida para desenvolver a solução para o problema.

1.5.2 Estado da arte

Este capítulo tem como objetivo apresentar as tecnologias existentes que podem ajudar a desenhar e implementar uma solução para o problema. São abordados alguns conceitos teóricos de forma a sustentar o desenho e implementação da solução. Além disso, é apresentada uma análise comparativa dos serviços semelhantes que dão resposta ao problema desta dissertação.

1.5.3 Análise de valor

Este capítulo tem como objetivo apresentar a análise de valor. A análise de valor começa com uma introdução sobre a definição de análise de valor. De seguida é feita uma introdução do processo de inovação seguindo o modelo NCD e o seu enquadramento no contexto do projeto. Consequentemente, é feita uma definição de valor, valor percecionado e valor para o cliente. Em seguida, é feita uma proposta de valor com o auxílio do modelo canvas. Por último, é usada a metodologia AHP para decidir a melhor solução a implementar tendo em conta a solução que proporciona o maior valor para o cliente.

1.5.4 Análise da solução

Este capítulo tem como objetivo apresentar os requisitos funcionais, não funcionais e o modelo de domínio. Nos requisitos funcionais existe uma divisão clara entre os casos de uso de cada ator. Cada caso de uso vem acompanhado de uma descrição e um diagrama que detalha a funcionalidade. Os requisitos não funcionais são descritos utilizando o modelo FURPS+. Por fim é apresentado o modelo de domínio para o projeto.

1.5.5 Desenho da solução

Este capítulo tem como objetivo apresentar arquitetura do sistema que visa resolver os requisitos funcionais e não funcionais. Para ajudar a ilustrar a arquitetura, recorreu-se a diagramas de componentes e de implantação. Após a definição da arquitetura, foram desenhados os casos de uso. O desenho vem acompanhado de diagramas de sequência, de classe e de *packages*.

1.5.6 Implementação da solução

Este capítulo tem como objetivo apresentar a implementação da solução. Na implementação é contemplado todos os padrões de software utilizados no desenvolvimento de cada serviço integrante da solução. É feita uma breve apresentação da integração com as tecnologias de reconhecimento e síntese de voz. Por último é apresentado o modelo de dados que grava a configuração de um IVR, incluindo as configurações de reconhecimento e síntese de voz.

1.5.7 Avaliação da solução

Este capítulo tem como objetivo apresentar avaliação da solução. Na avaliação da solução é abordado o uso de testes unitários para garantir a qualidade do código assim como a integração dos mesmos com as *pipelines* de CI/CD, para que este processo de validação seja feito de forma automática a cada *commit*. Por fim é descrito o processo de integração do sistema de IVR com o sistema de monitorização da BySide.

1.5.8 Conclusão

Este capítulo tem como objetivo apresentar as conclusões sobre a presente dissertação. É feita uma apreciação sobre os objetivos atingidos quer a nível pessoal, profissional e académico. De seguida é apresentado as limitações do projeto assim como o trabalho a ser realização no futuro. Por fim, é feita uma apreciação global sobre o desenvolvimento do projeto e da dissertação.

Capítulo 2

Estado da Arte

Este capítulo tem como objetivo explorar as tecnologias que ajudam a resolver o problema descrito anteriormente, para isso é feita uma análise e comparação considerando as funcionalidades especificadas na documentação disponibilizada por cada tecnologia. Além disso, é feita uma análise de serviços que oferecem IVRs como parte do seu produto. O objetivo de analisar outros serviços de IVR permite perceber como o produto tem evoluído e de que forma é possível melhorar e inovar na área de voz e de *customer self-service*.

2.1 Workflow

Um *workflow* [5] é uma sequência de tarefas que processam um conjunto de dados. Os *workflows* verificam-se em todos os tipos de actividades económicas e industriais. Sempre que os dados são transferidos entre seres humanos e sistemas, é criado um *workflow*. Os *workflows* são um conjunto de caminhos que transformam dados em bruto para dados processados.

2.1.1 Workflow automatizado

Um *workflow* automatizado [6] é a implementação de um *workflow* de forma automática, ou seja, qualquer sequência de tarefas é feita sem qualquer intervenção manual. Baseado em regras pré-definidas os *workflows* automatizados podem executar tarefas como: agendamentos, envio de e-mails, envio de campanhas, etc...

A figura 2.1 apresenta um exemplo de um *workflow* automatizado.

Figura 2.1: Exemplo de *workflow* automatizado

2.1.2 IVR como um workflow

Os *workflows* estão presentes em todas as atividades económicas e industriais como já foi referido anteriormente. O IVR é um *workflow* que pode incluir tarefas automáticas assim como tarefas manuais.

Em termos de tarefas manuais, quem inicia a chamada precisa de introduzir várias opções e fornecer alguns dados, para que a chamada seja entregue no agente habilitado para atender a chamada.

Durante a chamada, os pedidos a serviços externos, as decisões baseadas em capacidade do *contact center* e o processo de distribuição automática de chamadas (ACD) são considerados como processos automáticos, pois quem iniciou a chamada não teve qualquer intervenção no decorrer dos processos.

Logo, considera-se o IVR como um *workflow* misto, pois pode possuir processos automáticos como manuais. A figura 2.2 apresenta um exemplo de um *workflow* de IVR.

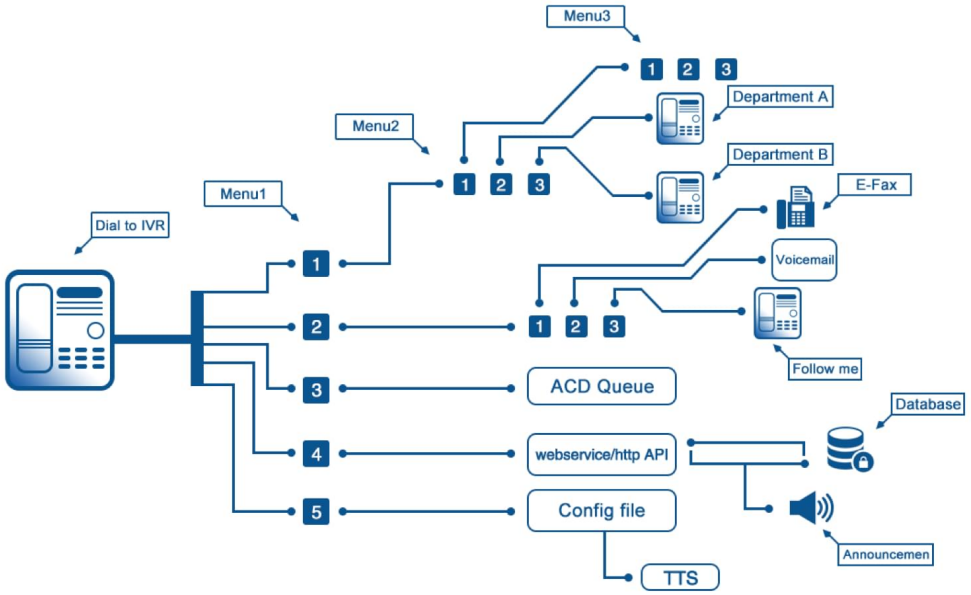

Figura 2.2: Exemplo de um *workflow* em IVR

2.2 VoIP

VoIP ou *Voice over Internet Protocol* [7][8] é um sistema de comunicação que permite efetuar chamadas telefónicas usando a internet ou, mais especificamente, utilizando o *Internet Protocol*. Existem vários tipos de sistemas de VoIP, uns permitem efetuar chamadas para as extensões que estão registados no sistema, enquanto que outros permitem efetuar chamadas para números de rede pública.

2.2.1 Funcionamento

Em vez de usar sinais analógicos para transmissão de áudio e de informação, o VoIP [7][8] utiliza sinais digitais, isto é, pacotes IP para transmitir áudio e outras informações relacionados com a sessão da chamada.

Para a gestão de sessões foram criados vários protocolos [8] (e.g. SIP, H.323), que facilitam a troca de informação entre os intervenientes durante a chamada (e.g. sinalização da chamada, *codecs* usados, ...).

Para a gestão dos pacotes de áudio foram criados outros protocolos [9] (e.g. RTP, RTCP, SRTP), que permitem codificar o áudio e fazer a transmissão do mesmo. De forma a prevenir perda de pacotes, implementou-se mecanismos de compensação de *jitter* e deteção de pacotes desordenados.

Exemplos de serviços e aplicações baseadas em VoIP:

- Skype (aplicação de áudio/vídeo chamadas);
- Asterisk (software de PBX);
- Twilio (fornecedor de comunicações na nuvem);

2.3 DTMF

DTMF ou *Dual-Tone Multi-Frequency* [10] é um sistema que converte dígitos em sinais sonoros utilizando duas frequências de som. Com este sistema as redes telefónicas conseguem identificar o destino da chamada e estabelecer a comunicação com esse destino. Além disso este sistema permite efetuar escolhas durante um IVR, pelo facto que cada dígito produz um sinal único.

Na figura 2.3 é possível visualizar o conjunto de frequências sonoras que constituem cada dígito de 0 a 9, e #.

	1209 Hz	1336 Hz	1477 Hz
697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#

Figura 2.3: Pares de frequências para os dígitos 0 a 9, e #

2.4 Drachtio

O Drachtio [11] é um servidor de SIP. Ele permite construir aplicações SIP como se fossem aplicações web. É um servidor com alta performance e de fácil configuração.

As aplicações SIP são construídas usando uma *framework* para Node.js, denominada de drachtio-srf. A *framework* disponibiliza uma interface para registar uma cadeia de *middlewares*, à semelhança das aplicações web.

O Drachtio suporta o protocolo SIP por UDP, TCP e WebSockets.

2.5 Asterisk

O Asterisk [12][13] é um software *open-source* que permite construir aplicações de comunicações (e.g. serviços VoIP, aplicações de chat). O Asterisk implementa diversos protocolos entre eles o SIP, o IAX, o RTP, WebSockets, HTTP, etc.

O Asterisk [8] consegue criar chamadas de VoIP, interagir com a PSTN, gerir filas de agentes, criar conferências de áudio e vídeo, SMS, Voicemail, etc.

Asterisk possui uma funcionalidade denominada de *Dialplan* [14], que é um conjunto de regras e instruções que permite manipular os canais de uma chamada. Com este conjunto de regras e instruções é possível encaminhar as chamadas para vários destinos. Além disso é possível fazer integração com serviços externos no Asterisk.

Para a integração com aplicações externas o Asterisk possui 3 interfaces de comunicação:

- AGI (Asterisk Gateway Interface)
- AMI (Asterisk Manager Interface)
- ARI (Asterisk REST Interface)

2.5.1 AGI

AGI ou *Asterisk Gateway Interface* [15] é uma interface aplicacional que permite interagir com um canal de uma chamada presente no Asterisk.

A comunicação com a AGI pode ser desenvolvida em várias linguagens de programação (e.g. PHP, Java, C#). A AGI faz uso do *stdin* e *stdout* para transmitir informação entre a aplicação externa e o Asterisk.

Toda a informação transmitida para o Asterisk é designado de ação. Existem vários tipos de ações como por exemplo:

- Reprodução de áudio;
- Gravação de um canal;
- Esperar por DTMFs;

2.5.2 AMI

AMI ou *Asterisk Manager Interface* [16] é uma interface comunicação que permite gerir o Asterisk. Esta interface segue o modelo servidor/cliente e as comunicações entre o cliente e servidor são efetuadas através de TCP/IP.

A AMI [16] possui um protocolo aplicacional próprio. Existem dois tipos de mensagens que são trocadas entre o cliente e o servidor, as *actions* e os *events*.

As *actions* são pedidos, que se traduzem em comandos executados pelo Asterisk (e.g. Originar uma chamada, verificar o estado das filas de agentes, apresentar o estado do sistema, apresentar o número total de chamadas a decorrer). As *actions* são acompanhadas sempre de uma resposta a indicar o sucesso ou insucesso da mesma efetuada.

Os *events* são um conjunto de mensagens que o Asterisk envia para o cliente. Alguns dos *events* são despoletados pelas ações, outros são despoletados periodicamente pelo Asterisk.

Existem várias bibliotecas desenvolvidas em várias linguagens de programação [17], que permitem estabelecer este tipo de comunicação com o Asterisk (e.g. asterisk-ami-client, AmiClient, Asterisk-Java, PAMI).

2.5.3 ARI

ARI ou *Asterisk RESTful Interface* [18] é uma interface de comunicação que permite interagir com alguns recursos do Asterisk, seguindo uma abordagem de cliente/servidor igual ao AMI, mas utilizando os protocolos HTTP e WebSockets.

A ARI [18] é a junção de algumas funcionalidades da AGI e AMI, sendo desenhada para disponibilizar uma interface intuitiva que expões vários recursos primitivos de forma transparente para permitir construir aplicações comunicação à medida.

Com a ARI [18] é possível controlar canais, *bridges*, extensões, gravações, reprodução de áudio, etc.

À semelhança do AMI, existem eventos e ações. [18] Os eventos são envidados para o cliente através dos WebSockets no formato de JSON. As ações sobre são efetuadas através de pedidos HTTP.

Existem algumas bibliotecas que permitem interagir com esta interface [19] (e.g. node-ari, AsterNET.ARI, ari4java, phpari).

2.6 Stasis

O Stasis [18] é uma instrução de *Diaplan* que permite enviar um canal para uma aplicação externa através do uso da ARI. O canal passa a pertencer à aplicação externa, tomando esta o controlo total sobre este canal.

A partir do momento em que um canal entra numa aplicação Stasis, o controlo da chamada segue as regras desenhada na aplicação externa. Isto torna bastante versátil e dinâmico o processamento de chamadas.

2.7 Speech To Text

O *speech to text* (fala para texto) [20] é um conjunto de processos que permitem que um sistema computacional reconheça palavras e frases de uma gravação de áudio ou *stream* de áudio com vários locutores.

No início da década de 1950 [20], alguns avanços foram concretizados nesta área, com o objetivo de reconhecer algumas palavras e fonemas. Estes avanços tecnológicos foram

incorporados em telefones fixos desenvolvidos pela IBM, Bell e AT&T no início da década de 1970, para automatizar o processo de discagem de números telefónicos.

No início dos anos 80 [20], surge uma mudança de paradigma em relação ao reconhecimento automático de voz, que até então, era baseado em pattern matching e templates. Com o desenvolvimento do HMM (Hidden Markov Model), a deteção de voz começou a seguir um modelo estatístico e probabilístico. Este modelo é utilizado nos dias de hoje, tendo sido melhorado e aperfeiçoado com o desenvolvimento da computação paralela e a evolução das redes neuronais.

O *speech to text* pode ser aplicado em várias áreas e embutido em várias aplicações. Neste momento o reconhecimento de voz é aplicado em sistemas de telefonia (e.g. IVR, transcrições de conversas, etc.), controlo de voz de carros, assistentes virtuais, casas inteligentes, etc. [21]

Serviços que permitem construir aplicações baseadas em reconhecimento de voz:

- Google Cloud Speech-to-Text
- Azure Cognitive Services (Speech)
- IBM Watson
- Amazon Transcribe

2.7.1 Google Cloud Speech-to-Text

A API de *speech to text* da Google Cloud [22] permite ao utilizador construir aplicações de reconhecimento de voz.

Este serviço oferece vários modos de utilização [23]:

- Reconhecimento de curta duração para ficheiros pequenos, em que a duração máxima suportada é de 1 minuto;
- Reconhecimento de longa duração para ficheiros maiores que estão hospedados na Google Cloud Storage com a duração máxima de 480 minutos;
- Reconhecimento em tempo real, em que são enviados vários blocos de áudio, sendo a duração máxima destes blocos de 5 minutos.

O serviço suporta vários idiomas, existindo 120 combinações de línguas e variações linguísticas.

Funcionalidades disponibilizadas [24][25][26][27][28]:

- Deteção automática de idioma;
- Filtragem de conteúdo e de linguagem abusiva;
- Pontuação automática: Permite interpretar os silêncios dos áudios e traduzi-los para pontuação;
- Suporte para adaptação de modelos: Permite adicionar aos pedidos de reconhecimento palavras chaves ou frases para obter um melhor resultado na deteção;
- Identificação de locutor: Cria um fluxo de conversação onde é distinguido através de uma *label* o locutor que proferiu certa expressão ou frase.

Recursos de integração disponíveis [29][30]:

- HTTP REST API;
- SDK (e.g. Node.js, Java, Python, Go).

2.7.2 Azure Cognitive Services (Speech)

O Azure Cognitive Services (Speech) [31] é um serviço que permite construir aplicações de reconhecimento de voz utilizando uma API. Este serviço faz parte de um leque de serviços de inteligência artificial explorando áreas de *deep learning*.

Esta API possui dois modos de funcionamento [32]:

- Reconhecimento de voz até ao primeiro período de silêncio detetado, obtendo assim apenas uma frase ou expressão;
- Reconhecimento de voz contínua em que a paragem do reconhecimento é feita através de um intervalo de silêncio configurável, isto é, o limite de silêncio foi atingido.

Ambos os modos suportam *streaming* de áudio em tempo real.

Funcionalidades disponibilizadas [32][33][34][35]:

- Deteção automática de idioma;
- Filtragem de conteúdo e de linguagem abusiva;
- Suporte para adaptação de modelos: Permite adicionar aos pedidos de reconhecimento palavras chaves ou frases para obter um melhor resultado na deteção;
- Suporte para modelos customizados: É possível criar um modelo de deteção customizado diferente dos predefinidos. Isto pode melhorar a precisão em certos negócios com linguagem técnica muito específica. Também é disponibilizada uma plataforma que permite treinar os modelos;

Recursos de integração disponíveis [32]:

- HTTP REST API;
- SDK (e.g. Node.js, Java, C#, Python).

2.7.3 IBM Watson

O IBM Watson [36][37] disponibiliza um conjunto de serviços de inteligência artificial, entre eles encontra-se o serviço de reconhecimento de voz. Este serviço disponibiliza uma API, para fazer o reconhecimento de voz.

O IBM Watson [37] disponibiliza dois modos de funcionamento:

- Reconhecimento de voz único, que efetua um reconhecimento de voz até que seja detetado o primeiro silêncio, extraindo apenas uma frase ou expressão;
- Reconhecimento de voz assíncrono, que permite reconhecimento de frases e expressões em tempo real que acaba quando atinge um limite de silêncio configurável.

O serviço disponibiliza a hipótese de escolher entre dois tipos de protocolos aplicacionais: HTTP ou WebSocket.

Segundo a IBM [37], para fazer o reconhecimento recorrendo ao protocolo HTTP são necessários 4 pedidos/conexões, enquanto que o protocolo WebSocket utiliza uma única conexão para efetuar os pedidos de reconhecimento.

Neste momento, o suporte de idiomas é bastante limitado, sendo disponibilizados apenas 11 idiomas.

Funcionalidades disponibilizadas [37][38][39]:

- Filtragem de conteúdo e de linguagem abusiva (apenas em inglês);
- Suporte para modelos customizados: É possível criar dois tipos de modelo customizados. Existe a possibilidade de utilizar um modelo linguístico próprio com uma linguagem mais técnica ou até mesmo introduzir um modelo para suportar um idioma. Existe a possibilidade de utilizar um modelo que se adapte à acústica do ambiente em que as gravações de áudio se encontram. Qualquer um destes modelos pode ser treinado para que sejam melhorados os resultados das deteções;
- Identificação de locutor: Cria um fluxo de conversação onde é distinguido através de uma tag o locutor que proferiu certa expressão ou frase;
- Correção automática de sinais de áudio na existência de má qualidade do mesmo;
- Suporte para adaptação de modelos: Permite adicionar aos pedidos de reconhecimento palavras chaves ou frases para obter um melhor resultado na deteção.

Recursos de integração disponíveis [37]:

- HTTP REST API (síncrona, assíncrona);
- WebSockets API;
- SDK (e.g. Node.js, Java, C#, Python).

2.7.4 Amazon Transcribe

O Amazon Transcribe [40] é um serviço que disponibiliza uma API para construção de aplicações de reconhecimento de voz. A API é disponibilizada por HTTP e por WebSocket.

O Amazon Transcribe [41] possui dois modos de funcionamento:

- Reconhecimento de voz em tempo real - permite reconhecimento de frases e expressões em tempo real que acaba quando é atingido um período de silêncio;
- Reconhecimento de voz para ficheiros de áudio - permite que seja feito o reconhecimento de voz de um ficheiro de áudio na sua totalidade, com limite de 4h de duração ou tamanho menor que 2GB.

O serviço está limitado a 31 idiomas, sendo que apenas 12 destes têm suporte para reconhecimento de voz em tempo real.

Funcionalidades disponibilizadas [40][42][43]:

- Deteção automática de idioma;
- Pontuação automática;
- Filtragem de conteúdo e linguagem abusiva;

- Suporte para modelos customizados: Permite a construção e treino de um modelo de deteção customizado diferente dos predefinidos. Isto pode melhorar a precisão das deteções em certos negócios com linguagem técnica muito específica;
- Suporte para adaptação de modelos: Permite adicionar aos pedidos de reconhecimento palavras chaves ou frases para obter um melhor resultado na deteção;
- Ofuscação automática de informação sensível: Consegue detetar informação sensível como número de cartão de crédito, número de telemóvel, número de segurança social, quando instruído para tal. Esta funcionalidade só está disponível para o idioma inglês (Estados Unidos) e apenas funciona com ficheiros de áudio;
- Identificação de locutor: Cria um fluxo de conversação onde é distinguido, através de uma *label*, o locutor que proferiu certa expressão ou frase.

Recursos de integração disponíveis [44]:

- HTTP REST API;
- WebSockets API;
- SDK (e.g. Node.js, Java, C#, Python).

2.7.5 Tabela comparativa

O objetivo da tabela 2.1 é fornecer uma visão comparativa de fácil compreensão. Na tabela 2.1 são mostrados dados dos vários serviços de *speech to text* analisados anteriormente relativamente às funcionalidades fornecidas. As mesmas estão descriminadas na primeira coluna da tabela 2.1. O símbolo "X"representa que o fornecedor de serviço implementa a funcionalidade, enquanto que o símbolo -"representa que o fornecedor de serviço não implementa a funcionalidade descrita.

	Google Cloud Speech To Text	Azure Cognitive Services (Speech)	IBM Watson	Amazon Transcribe
Suporte HTTP	X	X	X	X
Suporte WebSocket	-	-	X	X
Suporte gRPC	X	-	-	-
Suporte para ficheiros	X	X	X	X
Suporte para áudio em tempo real	X	X	X	X
Duração máxima por ficheiro	480 minutos	15 segundos para reconhecimento único Sem limites no reconhecimento contínuo	100MB por WebSocket 100MB por HTTP síncrono 1GB por HTTP assíncrono	240 minutos / 2GB
Duração máxima em tempo real	5 minutos	15 segundos para reconhecimento único Sem limites no reconhecimento contínuo	100MB por WebSocket 100MB por HTTP síncrono 1GB por HTTP assíncrono	-
Deteção automática de idioma	X	X	-	X
Pontuação automática	X	-	-	X
Filtragem de linguagem abusiva	X	X	X (inglês)	X
Modelos customizados		X	X	X
Adaptação de modelos	X	X	X	X
Identificação de locutor	X	-	X	X

Correção de sinal de áudio	-	-	X	-
Ofuscação de informação sensível	-	-	-	X (inglês - EUA)
Total de idiomas/variações linguísticas	120	95	11	31 (12 em tempo real)

Tabela 2.1: Tabela comparativa de funcionalidades de *speech to text* de serviços de nuvem

2.8 Text To Speech

O *text to speech* (síntese de voz) [45] é um processo de geração de fala de forma automática e sintética. O texto é introduzido numa máquina, sendo processado e interpretado. A máquina, por sua vez, produz os fonemas e sons associados às palavras contidas no texto.

No início do século XIX surgiram as primeiras máquinas falantes. [45] De forma bastante pioneira, estas máquinas utilizavam modelos mecânicos para reproduzir os sons das palavras introduzidas. Com a evolução da tecnologia, os modelos mecânicos foram substituídos por modelos elétricos na segunda metade do século XX, o que provocou um avanço significativo nesta tecnologia. Surgem então duas abordagens: *articulatory synthesis* e *formant synthesis*.

Durante os anos 70 [45], com o desenvolvimento dos computadores, surgiram os primeiros modelos computacionais de *text to speech*. Com estes avanços científicos, aparece uma nova abordagem, disruptiva de abordagens previamente desenvolvidas: *concatenative synthesis*.

Nos dias de hoje [45], o *text to speech* apresenta-se como um conjunto de algoritmos baseados nas técnicas previamente apresentadas. As melhorias proporcionadas na síntese de voz devem-se à inclusão de modelos estatísticos e probabilísticos nos algoritmos (e.g. Hidden Markov Model), ao avanço da tecnologia de deep learning e à performance da computação na nuvem.

Embora o avanço tecnológico tenha proporcionado melhorias significativas, o *text to speech* possui algumas limitações. [46] O ser humano, nos dias de hoje, ainda consegue distinguir uma voz real de uma voz artificial. Com o intuito de diminuir a distância entre a máquina e o ser humano, foram desenvolvidos critérios que permitem classificar os sistemas de *text to speech*.

Os critérios utilizados para classificar este tipo de sistemas, segundo a Sciforce [46], são:

- *Intelligibility*: Classifica a qualidade do áudio produzido ou das palavras produzidas;
- Naturalidade: Classifica o áudio produzido tendo em conta o tempo, pontuação, emoção e pronúncia;
- Preferência: Classificação baseada na preferência do ouvinte, por certo sistema de *text to speech*;
- Compreensão: Classifica o grau de compreensão de uma mensagem produzida por *text to speech*.

O *text to speech* é utilizado em várias áreas da sociedade. [47] A medicina é uma das áreas que tira grande proveito desta tecnologia. Ela ajuda pessoas com surdez e mudez, ajuda crianças no desenvolvimento da fala e permite a pessoas com patologias relacionadas com as cordas vocais a possibilidade de comunicar.

Para além da medicina, esta tecnologia é aplicada nos assistentes virtuais, nos sistemas de entretenimento de carros, na aquisição de uma nova linguagem, etc.

Serviços que permitem construir aplicações baseadas em síntese de voz:

- Google Cloud Text-to-Speech;
- Azure Cognitive Services (Speech);
- IBM Watson;

- Amazon Polly.

2.8.1 Google Cloud Text-to-Speech

A API de text to speech da Google Cloud [48][49] permite a um utilizador construir aplicações de síntese de voz. Ela suporta síntese através de texto ou através de SSML. Atualmente são suportados 40 idiomas/variações linguísticas, incluindo português, inglês e espanhol.

Funcionalidades disponibilizadas [48][49]:

- Seleção da voz: Permite escolher de um leque de vozes (e.g. feminino, masculino);
- Seleção de idioma;
- WaveNet Voices: Vozes geradas por *deep learning*, que são mais realistas dos que as restantes vozes;
- Suporte para SSML;
- Suporte para vários formatos de áudio (e.g. MP3, WAV, OGG);
- Ajuste de timbre;
- Ajuste de velocidade;
- Ajuste de volume;
- Vozes customizadas (beta): Permite carregar um modelo de voz customizado e treinar esse modelo.

Recursos de integração disponíveis [50][51]:

- HTTP REST API;
- gRPC;
- SDK (e.g. Node.js, Java, Python, Go).

2.8.2 Azure Cognitive Services (Speech)

O Azure Cognitive Services (Speech) [52] é um serviço que permite construir aplicações de baseadas em voz utilizando uma API. Este serviço faz parte de um leque de serviços de inteligência artificial.

O serviço suporta síntese através de texto ou através de SSML. Atualmente são suportados 60 idiomas/variações linguísticas [52], incluindo português, inglês e espanhol.

Funcionalidades disponibilizadas [52]:

- Seleção da voz: Permite escolher de um leque de vozes (e.g. feminino, masculino);
- Seleção de idioma;
- Neural Voices: Vozes geradas por *deep learning*, que são mais realistas dos que as restantes vozes;
- Suporte para SSML;
- Suporte para vários formatos de áudio (e.g. MP3, WAV, OGG);

- Ajuste de timbre;
- Ajuste de velocidade;
- Ajuste de volume;
- Vozes customizadas: Permite carregar um modelo de voz customizado e treinar esse modelo.

Recursos de integração disponíveis [53]:

- HTTP REST API;
- SDK (e.g. Node.js, Java, C#, Python).

2.8.3 IBM Watson

O IBM Watson [54] disponibiliza um conjunto de serviços de inteligência artificial, entre eles encontra-se o serviço de síntese de voz. Este serviço disponibiliza uma API, para fazer a conversão de texto para voz.

O serviço suporta síntese através de texto ou através de SSML. [54] Atualmente são suportados 16 idiomas/variações linguísticas, incluindo português do Brasil, inglês e espanhol.

Funcionalidades disponibilizadas [55][54][56]:

- Seleção da voz: Permite escolher de um leque de vozes (e.g. feminino, masculino);
- Seleção de idioma;
- Neural Voices: Vozes geradas por *deep learning*, que são mais realistas dos que as restantes vozes;
- Suporte para SSML;
- Suporte para vários formatos de áudio (e.g. MP3, WAV, OGG);
- Ajuste de timbre (apenas em SSML);
- Ajuste de velocidade (apenas em SSML);
- Ajuste de volume (apenas em SSML);
- Estilos de fala (apenas em SSML): Permite definir o estilo de fala (e.g. amigável, informativo);
- Vozes customizadas: Permite carregar um modelo de voz customizado e treinar esse modelo.

Recursos de integração disponíveis [54]:

- HTTP REST API;
- WebSockets API;
- SDK (e.g. Node.js, Java, C#, Python).

2.8.4 Amazon Polly

O Amazon Polly [57] é um serviço que disponibiliza uma API para construção de aplicações de síntese de voz. O serviço suporta síntese através de texto ou através de SSML. Atualmente são suportados 29 idiomas/variações linguísticas, incluindo português, inglês e espanhol.

Funcionalidades disponibilizadas [57]:

- Seleção da voz: Permite escolher de um leque de vozes (e.g. feminino, masculino);
- Seleção de idioma;
- NTTS voices: Vozes geradas por *deep learning*, que são mais realistas dos que as restantes vozes;
- Suporte para SSML;
- Suporte para vários formatos de áudio (e.g. MP3, WAV, OGG);
- Áudio em tempo real: Permite reproduzir áudio à medida que o serviço vai convertendo o texto;
- Ajuste de timbre (apenas em SSML);
- Ajuste de velocidade (apenas em SSML);
- Ajuste de volume (apenas em SSML);
- Estilos de fala (apenas em SSML): Permite definir o estilo de fala (e.g. amigável, informativo);
- Customização de vocabulário: Permite definir uma pronúncia diferente para uma palavra;
- Vozes customizadas: Permite carregar um modelo de voz customizado e treinar esse modelo;

Recursos de integração disponíveis [58]:

- HTTP REST API;
- SDK (e.g. Node.js, Java, C#, Python).

2.8.5 Tabela comparativa

O objetivo da tabela 2.2 é fornecer uma visão comparativa de fácil compreensão. Na tabela 2.2 são mostrados dados dos vários serviços de *text to speech* analisados anteriormente relativamente às funcionalidades fornecidas. As mesmas estão descriminadas na primeira coluna da tabela 2.2. O símbolo "X"representa que o fornecedor de serviço implementa a funcionalidade, enquanto que o símbolo -"representa que o fornecedor de serviço não implementa a funcionalidade descrita.

	Google Cloud Speech To Text	Azure Cognitive Services (Speech)	IBM Watson	Amazon Transcribe
Suporte HTTP	X	X	X	X
Suporte WebSocket	-	-	X	-
Suporte gRPC	X	-	-	-
Seleção da voz	X	X	X	X
Seleção de idioma	X	X	X	X
Neural voices	X	X	X	X
Suporte para SSML	X	X	X	X
Formatos de áudio	MP3, WAV, OGG	MP3, WAV, OGG	MP3, WAV, OGG	MP3, WAV, OGG
Ajuste de timbre	X	X	Apenas SSML	Apenas SSML
Ajuste de velocidade	X	X	Apenas SSML	Apenas SSML
Ajuste de volume	X	-	Apenas SSML	Apenas SSML
Vozes customizadas	X	X	X	X
Áudio em tempo real	-	-	-	X
Vocabulário customizado	-	-	-	X

Tabela 2.2: Tabela comparativa de funcionalidades de *text to speech* de serviços de nuvem

2.9 Análise de produtos semelhantes aos IVRs BySide

Nesta secção são apresentados alguns produtos, existentes no mercado, semelhantes aos IVRs da BySide. Para cada produto é feita uma breve apresentação da empresa que os comercializa assim como as principais funcionalidades presentes em cada produto. No final é feita uma tabela comparativa das funcionalidades presentes em cada produto.

2.9.1 Genesys

É uma empresa que disponibiliza um serviço de *contact center* na nuvem [59], que permite a gestão de agentes (humanos e virtuais), chamadas, chats, emails e SMS. Além destas funcionalidades ainda permite recolher métricas e dados que ajudam a gerir os pontos anteriormente enunciados.

Em relação ao serviço de IVR [60][61][62], este possui as seguintes funcionalidades:

- Suporte de DTMFs;
- Nós de selecção;
- Áudios personalizados;
- Entrega de chamadas em filas de agentes;
- Reconhecimento de voz em várias línguas (*speech to text*);
- Suporte para linguagem natural;
- Suporte de *text-to-speech*;
- Integrações externas durante o IVR (e.g. HTTP, Salesforce, Microsoft Dynamics 365);
- Análise de sentimental;
- Métricas de atendimento;
- Pagamentos de serviços e compras.

2.9.2 Zendesk

É um sistema de CSP (Customer Support Platform) [63], que permite às empresas fazer suporte aos clientes em múltIPlos canais (e.g. voz, chat, SMS, email). Além da funcionalidade de CSP, ainda permite gestão de clientes (CRM - Customer Relationship Management) e proporciona um software de atendimento de call center na nuvem.

Em relação ao serviço de IVR [64], este possui as seguintes funcionalidades:

- Suporte de DTMFs;
- Nós de selecção;
- Áudios personalizados;
- Entrega de chamadas em filas de agentes.

2.9.3 Talkdesk

É uma empresa que disponibiliza um serviço de *contact center* na nuvem [65], que permite gerir a operação de *contact center* numa só plataforma. Além de que, o serviço oferece a possibilidade de contactar com clientes através de múltiplos canais (e.g. voz, chat, SMS, email).

Em relação ao serviço de IVR [66], este possui as seguintes funcionalidades:

- Suporte de DTMFs;
- Nós de selecção;
- Áudios personalizados;
- Entrega de chamadas em filas de agentes;
- Reconhecimento de voz em várias línguas (*speech to text*);
- Suporte para linguagem natural;
- Suporte de *text-to-speech*;
- Integrações externas durante o IVR (e.g. HTTP, Salesforce, Microsoft Dynamics 365);
- Métricas de atendimento.

2.9.4 E-goi

É uma empresa que fornece um serviço de automatização de marketing [67], isto é envio de campanhas de marketing através de diferentes canais (e.g. SMS, email, voz, anúncios, notificações).

Em relação ao serviço de IVR [68], este possui as seguintes funcionalidades:

- Suporte de DTMFs;
- Nós de selecção;
- Áudios personalizados;
- Entrega de chamadas em filas de agentes;
- Suporte de *text-to-speech*.

2.9.5 Twilio

É uma empresa que disponibiliza um serviços de CPaaS (*Communications Platform as a Service*) [69] e uma solução de *contact center* na nuvem. Existe a possibilidade de interagir com vários canais (e.g. SMS, voz, chat), além de ser possível fazer integração e personalização dos vários serviços disponíveis.

Em relação ao serviço de IVR [70][71], este possui as seguintes funcionalidades:

- Suporte de DTMFs;
- Nós de selecção;
- Áudios personalizados;

- Entrega de chamadas em filas de agentes;
- Reconhecimento de voz em várias línguas (*speech to text*);
- Suporte de *text-to-speech*;
- Integrações externas durante o IVR (e.g. HTTP);
- Métricas de atendimento;
- Pagamentos de serviços e compras.

2.9.6 Tabela comparativa

O objetivo da tabela 2.3 é fornecer uma visão comparativa de fácil compreensão sobre os vários serviços de IVR existentes no mercado. Na tabela 2.3 são apresentados dados sobre os vários serviços e que funcionalidades estão implementadas para suportar o produto de IVR. As mesmas estão descriminadas na primeira coluna da tabela 2.3. O símbolo "X"representa que o fornecedor de serviço implementa a funcionalidade, enquanto que o símbolo -"representa que o fornecedor de serviço não implementa a funcionalidade descrita.

	Genesys	Zendesk	Talkdesk	E-goi	Twilio
Suporte de DTMFs	X	X	X	X	X
Nós de selecção	X	X	X	X	X
Áudios personalizados	X	X	X	X	X
Entrega de chamadas em filas de agentes	X	X	X	X	X
Reconhecimento de voz em várias línguas (speech to text)	X	-	X	-	X
Suporte de text-to-speech	X	-	X	X	X
Integrações externas durante o IVR	HTTP, Salesforce, Microsoft Dynamics 365	-	HTTP, Salesforce, Microsoft Dynamics 365	-	HTTP
Métricas de atendimento	X	-	X	-	X
Pagamentos de serviços e compras	X	-	-	-	X
Suporte para linguagem natural	X	-	-	-	-
Analise sentimental	X	-	-	-	-

Tabela 2.3: Tabela comparativa de funcionalidades de serviços que disponibilizam IVRs

Capítulo 3

Análise de Valor

Neste capítulo é apresentada uma análise de valor. O capítulo começa com a definição de análise de valor e o processo de geração de inovação. É apresentado o modelo NCD (*New Concept Development*) e enquadrado ao problema desta dissertação. De seguida foi apresentado o método AHP (*Analytic Hierarchy Process*). Este método foi utilizado para selecionar de forma objetiva as ideias geradas pela aplicação do modelo NCD. Por fim é realizada uma breve introdução sobre valor, valor percecionado e valor para o cliente e elaborada uma proposta de valor seguindo o *business model canvas*.

3.1 Definição de análise de valor

Segundo Nick Rich e Matthias Holweg [72], a análise de valor pode ser definida como um processo de revisão sistemático que é aplicado a produtos e serviços, a fim de comparar as funcionalidades do produto com as exigências do cliente. O objetivo é satisfazer os requisitos do cliente, com o menor custo possível, mantendo o desempenho especificado e a fiabilidade necessária para o produto ou serviço.

Adicionalmente, na análise de valor [72] existe a preocupação de identificar e eliminar produtos ou funcionalidades de serviços, que não adicionam valor ao cliente, e que simultaneamente, afetam os custos do processo de produção ou fornecimento do serviço.

Rich e Holweg [72], referem ainda que a descoberta e inovação de novos processos pode ser incorporada em produtos existentes com o objetivo de aumentar a qualidade dos mesmos e otimizar os custos dos mesmos.

Contudo a inovação possui sempre um grau de risco elevado, quer a nível económico, quer a nível logístico. Para mitigar este tipo de situações, foi desenvolvido um processo para a inovação. A figura 3.1 apresenta o processo tradicional e iterativo de inovação.

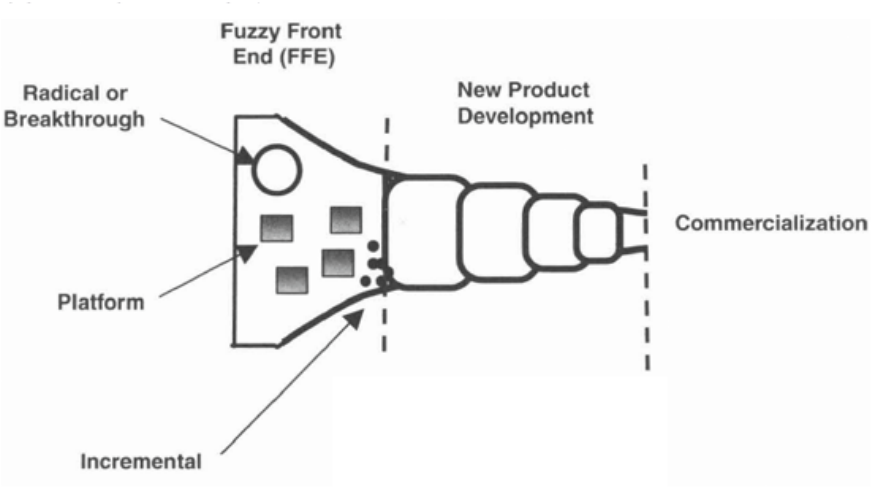

Figura 3.1: Processo tradicional e iterativo de inovação

O processo de inovação apresentado na figura 3.1 encontra-se dividido em três fases fundamentais apresentadas na lista seguinte:

1. Fuzzy Front End (FFE) – Fase na qual são geradas ideias e identificadas oportunidades, existe um nível de incerteza, imprevisibilidade e especulação. Nesta etapa são realizadas pesquisas com o objetivo de minimizar riscos e otimizar o potencial da inovação;
2. New Product Development (NPD) – Ambiente bem planeado, orientado a objetivos e com foco no desenvolvimento do produto ou serviço;
3. Comercialização – Etapa do processo na qual é realizada a promoção e venda do produto ou serviço.

Nesta dissertação é abordado o modelo NCD (New Concept Development) de Koen [73], no qual são retratadas as várias etapas de um processo de inovação, desde a geração de ideias à sua seleção.

3.1.1 Modelo NCD

O modelo NCD fornece uma linguagem comum para os elementos constituintes do FFE. Este modelo encontra-se dividido em três áreas:

- Motor;
- Elementos fundamentais (identificação de oportunidade, análise de oportunidade, geração de ideias e enriquecimento, seleção de ideias, definição de conceito);
- Fatores que influenciam o processo.

Na figura 3.2 é apresentada uma imagem ilustrativa do modelo NCD.

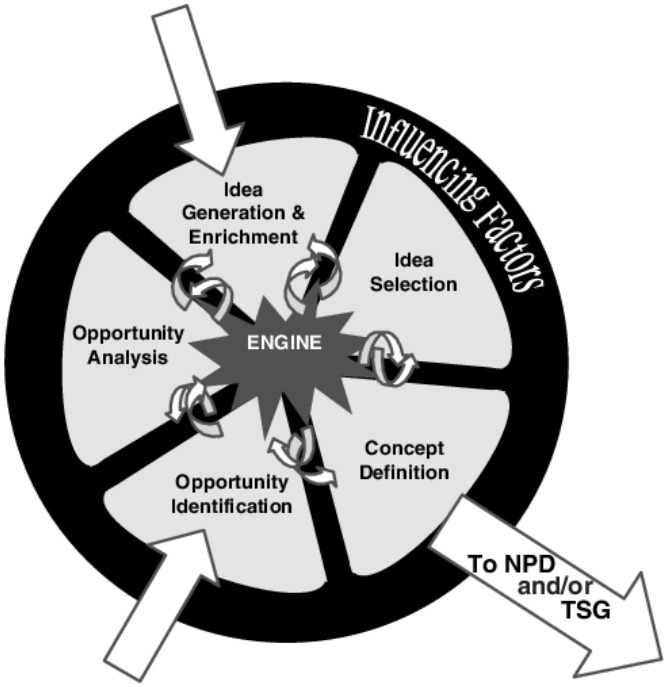

Figura 3.2: Modelo NCD

Através da observação da figura 3.2 é possível constatar que o processo pode ser iniciado pela geração de ideias, ou pela identificação das oportunidades. Isto é visível pelas setas com direção de entrada.

A seta de saída indica que a definição dos conceitos abandona o modelo, entrando nos processos NPD e/ou TSG (Technology Stage Gate).

Na figura 3.2 observa-se as três áreas que constituem o modelo, nomeadamente o motor (*engine*), os elementos fundamentais (*wheel*) e os fatores de influência (*rim*). Estas três áreas encontram-se explicadas nos sub-tópicos a seguir apresentados.

Motor

O motor é a parte central do modelo e representa um nível executivo de gestão, concedendo poder aos cinco elementos fundamentais enumerados na figura 3.2.

Elementos fundamentais

Definidos na "*wheel*", os elementos fundamentais do modelo NCD podem ser definidos da seguinte forma:

- **Identificação de oportunidade**: uma das actividades em que um conceito de projecto pode iniciar segundo o NCD. Fase de descoberta de um fosso comercial ou tecnológico que existe entre a situação actual do mercado e um futuro previsível;
- **Análise de oportunidade**: análise da oportunidade identificada;
- **Geração de ideias e enriquecimento**: uma das actividades em que um conceito de projecto pode iniciar segundo o NCD. Refere-se à criação da forma mais básica e embrionária de um produto ou serviço, incluindo algumas alternativas possíveis;

- **Seleção de ideias**: selecção das principais funcionalidades e métodos para benefício do cliente;
- **Definição de conceito**: é a realização de um produto ou serviço bem definido que inclua as suas principais características e benefícios para o cliente

Fatores de influência

Representa os elementos do mercado externo que influenciam directa ou indirectamente o fluxo entre as fases da "*wheel*". O fatores podem ser de carácter político, económico, social e tecnológico.

3.1.2 Enquadramento no modelo

O objetivo desta sub-secção é enquadrar o problema desta dissertação no modelo NCD, para efetuar uma análise sobre o projeto e a inovação que apresenta para o mercado do marketing e das comunicações.

Identificação de oportunidade

A introdução de um novo sistema de IVR serve para gerir de forma inteligente os recursos ao nível dos *contact centers*. Um IVR permite aos clientes de voz a oportunidade de *self-service*.

Para um *contact center* [74] é vantajoso possuir estes processos automáticos de *self-service*, uma vez que, está comprovado que baixam custos com a operação, provocando uma diminuição nas filas de espera e aumento da satisfação que por eles são servidos.

Na área de marketing, o IVR é um método comprovado que aumenta a conversão de *leads* [74], além de melhorar a relação do cliente com a marca/organização.

Para a BySide, que possui clientes nas áreas das telecomunicações, banca e retalho, a criação de um novo sistema de IVR surge da necessidade de inovar e incorporar novas tecnologias no sistema de IVR.

Com o uso de tecnologias como o reconhecimento e síntese de voz cria-se uma grande oportunidade para aumentar a personalização adequado ao segmento do visitante e aplicada a um método, que por vezes é aborrecido e repetitivo.

Tendo em conta todos estes fatores, a BySide decidiu que existe uma grande oportunidade para criar valor nos clientes atuais e em futuros clientes com o desenvolvimento de um novo sistema de IVR.

Análise de oportunidade

Após a identificação da oportunidade, iniciou-se a análise da oportunidade. Começando por analisar o sistema atual de IVR, foram identificados alguns pontos de melhoria na solução existente. A seguir são enumerados os pontos a melhorar:

- Configurações de IVR devem ser uniformes;
- Disponibilização de uma interface gráfica para configurar o IVR;
- Integração do sistema de IVR com os vários produtos da BySide (Cloud Contact Centre e Bytalk);

- Integração com reconhecimento e síntese de voz.

Após identificação dos pontos de melhoria na solução, concluí-se que não é possível com o sistema atual de IVR efetuar as melhorias identificadas. Por isso, as melhorias serão desenvolvidas na nova versão do sistema de IVR, tendo em atenção que algumas das funcionalidades do sistema atual serão incorporadas no novo sistema.

Geração de ideias e enriquecimento

Após a realização da análise de oportunidade foram efetuadas algumas reuniões de *brainstorm* realizadas entre a equipa de desenvolvimento e a equipa de produto, onde foram concebidas algumas ideias que acrescentam valor e que vão de encontro aos pontos de melhoria identificados.

As ideias que surgiram do *brainstorm* encontram-se enumeradas na seguinte lista:

- Ideia 1: Criar todos os componentes do sistema de raiz, entre os componentes estaria a interface gráfica, o motor de IVR e o reconhecimento e síntese de voz;
- Ideia 2: Criar a interface gráfica e o motor de IVR e integrar com serviços externos de reconhecimento e síntese de voz.

Seleção de ideias

Após a geração de ideias, nesta etapa foi realizada uma reunião entre a equipa de desenvolvimento, equipa de produto e administração, para avaliar o custo do desenvolvimento e alocação de recursos humanos e financeiros no projeto, assim como uma estimativa para a realização do MVP (Minimum Viable Product).

Definição de conceito

O conceito para esta dissertação adicionar o módulo de IVR à solução de *contact center* da BySide. Com isto o produto torna-se mais atrativo para os vários clientes que a BySide possui, além de ser mais o elemento fundamental na comercialização do serviço a novos clientes, uma vez que já se comprovou que as vantagens deste serviço são uma mais valia para as empresas que o utilizam.

Fatores de influência

Os fatores de influência para desenvolvimento do sistema de IVR são enquadrados nos parâmetros descritos na secção que define o modelo NCD. Posto isto, a lista seguinte apresenta para cada tópico os fatores que geram influência nesta dissertação:

- Político-Legais: Existe a preocupação que o sistema cumpra todas as normas de RGPD, pois informação sensível poderá ser fornecida pelos os utilizadores durante um IVR;
- Económicos: Existem vários concorrentes no ramo de *contact center* e com a tendência económica de transição para processos automatizados é necessário oferecer um produto a preços competitivos;
- Social: Pessoas com deficiencias visuais incapazes de introduzir os DTMFs

- Tecnológicos: A automatização é a tendência global e as empresas procuram soluções que permitam atingir esse objetivo, os *contact centers* não são a exceção. A automatização do atendimento através de um IVR é vantajoso para eles, pois reduz os custos de operação.

3.2 Método AHP

O método AHP (Analytic Hierarchy Process) [75] é uma forma eficaz de definir prioridades entre decisões de grande complexidade, auxiliando a tomada de decisão. As prioridades são estabelecidas por critérios qualitativos e quantitativos. A ideia é dividir o problema em questão em níveis hierárquicos, para facilitar a sua avaliação e escolha da decisão mais correta.

Saaty [75], como autor do método, definiu uma escala que consiste num conjunto de níveis de importância com o propósito de comparar vários critérios definidos para definir as prioridades. Esta escala encontra-se representada na figura 3.3.

Escala	Avaliação	Recíproco	Comentário
Igual importância	1	1	Os dois critérios contribuem igualmente para os objetivos
Importância moderada	3	1/3	A experiência e o julgamento favorecem um critério levemente sobre outro
Mais importante	5	1/5	A experiência e o julgamento favorecem um critério fortemente em relação a outro
Muito importante	7	1/7	Um critério é fortemente favorecido em relação a outro e pode ser demonstrado na prática
Importância extrema	9	1/9	Um critério é favorecido em relação a outro com o mais alto grau de certeza
Valores intermediários	2, 4, 6 e 8		Quando se procura condições de compromisso (*compromise*) entre duas definições. É necessário acordo.

Figura 3.3: Escala fundamental de Satty

Os próximos tópicos apresenta o processo AHP e como foi aplicado na dissertação

3.2.1 Construção da árvore hierárquica

Nesta fase do método AHP é pretendida a definição do problema e a definição de critérios que irão servir para analisar as diversas soluções, construindo uma árvore hierárquica de decisão que contém o problema, os critérios e as alternativas.

Aplicando ao contexto desta dissertação:

- Problema: Criar um novo sistema de IVR
- Critérios: Tempo, Custo, Performance, Confiabilidade
- Alternativas: Ideia 1, Ideia 2

A figura 3.4 apresenta a árvore hierárquica de decisão desta dissertação.

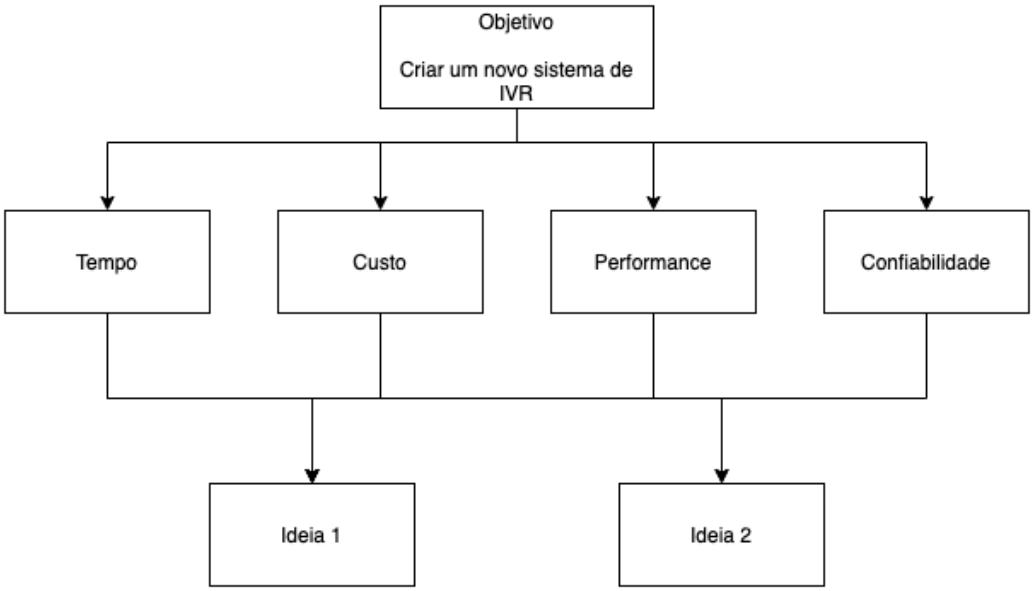

Figura 3.4: Árvore hierárquica de decisão

3.2.2 Avaliação dos critérios

Todos os critérios não possuem a mesma relevância para o sistema, por isso é necessário comparar os diversos critérios para definir quais os mais prioritários a ter em conta no desenvolvimento do sistema.

Por isso, foi elaborada uma matriz de comparação, onde a importância de cada critério é dada em função de cada um dos outros critérios. A matriz pode ser observada na tabela 3.1.

Critérios	Tempo	Custo	Performance	Confiabilidade
Tempo	1	3	5	5
Custo	1/3	1	3	3
Performance	1/5	1/3	1	1/3
Confiabilidade	1/5	1/3	3	1
Total	1.733	4.666	12	9.333

Tabela 3.1: Tabela classificação dos critérios usando a escala de fundamental de Satty

3.2.3 Obtenção do peso de cada critério

Para obter o peso de cada critério foi necessário normalizar os valores da tabela 3.1. A tabela 3.2 apresenta os valores normalizados de cada critério.

Critérios	Tempo	Custo	Performance	Confiabilidade
Tempo	0.58	0.64	0.42	0.54
Custo	0.2	0.21	0.25	0.32
Performance	0.11	0.07	0.08	0.04
Confiabilidade	0.11	0.07	0.25	0.11
Total	1	1	1	1

Tabela 3.2: Tabela normalizada dos critérios

O cálculo dos pesos de cada critério pode ser observado na tabela 3.3

Critérios	Tempo	Custo	Performance	Confiabilidade	Peso
Tempo	0.58	0.64	0.42	0.54	0.545
Custo	0.2	0.21	0.25	0.32	0.245
Performance	0.11	0.07	0.08	0.04	0.075
Confiabilidade	0.11	0.07	0.25	0.11	0.135

Tabela 3.3: Tabela com o peso dos critérios

Com o calculo dos pesos de cada critério é possível observar que o tempo de implementação é o critério mais importante para o novo sistema de IVR.

3.2.4 Escolha da alternativa

Não existe dúvida que a implementação de um sistema de raiz que inclua a implementação de uma interface gráfica, um motor de IVR, um sistema de reconhecimento de voz e um sistema de síntese de voz é mais demorado do que implementar apenas uma interface gráfica, um motor de IVR e integrar tecnologias de reconhecimento e síntese de voz. O facto da Ideia 1 ter um custo temporal maior prende-se pelo facto de, existir uma complexidade associada na elaboração de algoritmos de reconhecimento e síntese de voz. Não sendo a principal área de negócio da BySide, a implementação de reconhecimento e síntese de voz exige uma equipa dedicada na implementação. A BySide não possui a equipa necessário, por isso seria necessário contratar pessoas, o que provoca um aumneto no custo temporal desta ideia.

Por um lado a Ideia 1 é melhor em termos de custo, mas por outro esse não é o critério mais importante a ser considerado na implementação deste novo sistema de IVR. Assim sendo a Ideia 2 é a que será implementada tendo em visto todas as razões já enumeradas no parágrafo anterior.

3.2.5 Avaliação de consistência das prioridades

Para confirmar que todas as avaliações dos critérios foram consistente e que, os intervenientes na classificação dos mesmos não forneceram julgamentos subjetivos é necessário validar a consistência relativa da prioridades dos critérios. Este objetivo é atingido calculando o valor do rácio de consistência (*consistency ratio* ou CR) [76]. Caso este valor seja menor do que 0.1, então podemos considerar que os critérios se encontram classificados corretamente.

O CR é dado pela fórmula matemática:

$$CR = \frac{CI}{RI} \tag{3.1}$$

Em que o CI é o índice de consistência ou *consistency index*. O CI é dado pela seguinte fórmula:

$$CI = \frac{\lambda - n}{n - 1} \tag{3.2}$$

O RI é o índice aleatório ou *random index* é um valor normalizado de consistência que depende do tamanho do número de critérios de uma matriz. A figura 3.5 apresenta os valores de RI para os diferentes tamanhos de matrizes.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

Figura 3.5: Tabela de valores do RI por tamanho de matriz (n)

Aplicando as fórmulas aos valores obtidos nas tabelas 3.1 e 3.3 o CI é igual:

$$CI = \frac{4.2044 - 4}{4 - 1} \tag{3.3}$$

Então obtemos o valor do CI apróximado:

$$CI \cong 0.6816 \tag{3.4}$$

Assim sendo o valor do CR é igual:

$$CR = \frac{0.6816}{0.9} \tag{3.5}$$

O valor de CR aproximado é:

$$CR \cong 0.0757 \tag{3.6}$$

Comparando com o valor de referência de consistência verificamos que:

$$0.0757 < 0.1 \tag{3.7}$$

Logo comprova-se que avaliação dos critérios foram consistentes.

3.3 Valor da solução

Nesta secção é apresentada a proposta de valor sobre o projeto desenvolvido nesta dissertação. Primeiramente é introduzido o conceito de valor, o conceito de valor percecionado e conceito de perspectiva longitudinal de valor para o cliente. Por fim é aplicando o modelo CANVAS ao projeto relatado por esta dissertação

3.3.1 Valor

O valor possui uma definição subjetiva, visto que cada indivíduo avalia a aquisição de um produto ou serviço de maneira única. A troca de algum produto ou serviço tangível ou intangível, é indispensável para o funcionamento de qualquer negócio, pelo que a criação de valor para quem usufrui desse bem ou serviço é essencial [77].

3.3.2 Valor percecionado

A noção de *Customer Perceived Value* ou CPV, foi definida, segundo Kotler e Keller [78], como a diferença entre a avaliação do cliente de todos os benefícios e de todos os custos de uma certa oferta com as alternativas existentes.

O valor total percecionado pelo cliente é o valor monetário de todos os benefícios (e.g. económicos, funcionais) que os clientes esperam de um dado produto, serviço, ou mesmo da imagem da marca. O custo total do cliente é entendido como o pacote de custos que os clientes esperam assumir no esforço de identificar e efetuar um negócio específico. Logo é possível traduzir o CPV como uma formula matemática:

$$CPV = ValorTotalPercecionadoPeloCliente - CustoTotalDoCliente \quad (3.8)$$

3.3.3 Valor para o cliente

Woodall [79] em 2003 introduziu a noção de variação de tempo na perspectiva do que é valor para uma pessoa. A noção de Woodall para a variação temporal de "*Value of the Costumer*"foi dividida em quatro fases, antes, durante e após compra, assim como o após utilização.

Aplicando este modelo ao projecto em questão, a tabela 3.4 apresenta os benefícios e os sacrifícios nas várias fases deste modelo que os clientes da BySide sentirão.

	Benefícios	Sacrifícios
Pré compra	- Melhoria na configuração de um IVR; - Integração com reconhecimento de voz; - Integração com síntese de voz;	
Durante a compra	- Disponibilização do serviço de IVR; - Ajuda na configuração do IVR;	- Tempo de configuração do serviço;
Pós compra	- Entendimento das funcionalidades e limitações do sistema; - Sistema completamente funcional;	
Depois de utilização	- Implementação de novas funcionalidade para acompanhar a evolução do mercado e tecnológica;	- Tempo de implementação;

Tabela 3.4: Tabela com a perspectiva longitudinal de valor para o cliente

3.3.4 Proposta de valor

Uma proposta de valor pode ser definida, segundo o dicionário de Cambridge [80], como uma razão para comprar um produto ou serviço de quem o comercializa, com base no valor que este oferece para os clientes.

Logo, a proposta de valor é uma parte bastante importante para a divulgação de um produto ou serviço que se pretende comercializar. Deve responder de forma clara as necessidades dos clientes, apresentando razões que convençam os clientes a comprar o produto ou subscrever o serviço.

Business Model Canvas

Criado em 2005 por Alex Osterwalder [81], o *business model canvas* é um modelo estratégico de gestão, para desenvolver uma ideia de negócio, ou documentar uma ideia de negócio já existente.

Este modelo define uma representação gráfica onde pode ser observado a proposta de valor, a infraestrutura, os clientes e as fontes de rendimento e de custo. [82] A exposição gráfica de todos estes componentes permite as empresas identificar *trade-offs* no negócio, para construir um plano de execução do negócio.

A representação gráfica do *business model canvas* está dividida em:

- Proposta de valor (*Value Proposition*);
- Segmentos de clientes (*Customer Segments*);
- Relações com os clientes (*Customer Relationships*);

- Canais de comunicação (*Channels*);
- Atividades chave (*Key Activites*);
- Recursos chave (*Key Resources*);
- Parceiros chave (*Key Partners*);
- Fontes de receita (*Revenue Streams*);
- Custos de estrutura (*Costs Structure*).

Proposta de valor (Value Proposition)

A proposta de valor [83] é a descrição do valor que o negócio provoca no cliente que compra o produto ou subscreve o serviço de uma empresa. Normalmente o valor do produto/serviço é trocado por dinheiro proveniente de um cliente, pois o valor adquirido aliviou ou solucionou um problema existente para o cliente.

Nesta dissertação, a proposta de valor passa por: atendimento self-service, otimização de tempos de espera, personalização por segmentos e baixo custo de serviço.

Segmento de clientes (Customer Segments)

A segmentação de clientes [83] é a prática de dividir uma base de clientes em grupos de indivíduos que são semelhantes de formas específicas, tais como idade, sexo, interesses e hábitos de consumo.

Nesta dissertação, o segmento de clientes é diferente da definição, pois o produto é vendido a um grupo de empresas em vez de um grupo de indivíduos. Posto isto, o segmentos de clientes para este produto são: empresas de telecomunicações, banca e retalho.

Relações com os clientes (Customer Relationships)

A relação com os clientes [83] é definida como a forma como uma empresa interage com os seus clientes. Uma empresa pode interagir com os seus clientes através de:

- Canais online;
- Em pessoa;
- Eventos;
- Por telefone.

Nesta dissertação, a relação com os clientes é feita através do uso de canais online e em pessoa.

Canais de comunicação (Channels)

Os canais de comunicação [83] são definidos como as vias através das quais os clientes entra em contacto com o negócio e como se torna parte do ciclo de vendas da empresa.

Nesta dissertação, os canais de comunicação disponíveis para os clientes são: site, redes sociais, SEO.

Atividades chave (Key Activites)

As atividades chave [83] de um negócio/produto são as acções que o negócio desenvolve para alcançar a proposta de valor para os seus clientes.

Nesta dissertação, as atividades chaves desenvolvidas para atingir a proposta de valor são: reconhecimento de voz, síntese de voz e gestão de *workflows*.

Recursos chave (Key Resources)

Os recursos chave [83] são todos os recursos necessários para executar as atividades chave que são necessárias para o negócio.

Nesta dissertação os recursos chaves são: servidores, SIP trunks, base de clientes.

Parceiros chave (Key Partners)

Os parceiros chave [83] são uma lista de outras empresas/fornecedores/partes externas que são necessários para realizar as atividades chave do negócio e fornecer valor ao cliente.

Nesta dissertação não existe uma empresa ou fornecedor que é parceiro da BySide e que contribui para a realização das atividades chave.

Fontes de receita (Revenue Streams)

As fontes de receitas [83] são definidas como a forma pela qual o negócio converte a proposta de valor ou a solução para o problema do cliente em ganho financeiro. É também importante compreender o valor do negócio e traduzi-lo num preço justo que, tanto provoca ganhos financeiros no negócio, como é aceitável para o cliente pagar.

Nesta dissertação a principal fonte de receita é a utilização do sistema de IVR nas chamadas fornecidas pela plataforma BySide.

Custos de estrutura (Costs Structure)

Os custos de estrutura [83] são todos os custos inerentes à prática do negócio (e.g. sálarios, aluguer de escritório).

Nesta dissertação os custos de estrutura associados ao projeto são: hardware, salário e serviços de nuvem.

A figura 3.6 apresenta a representação gráfica do *business model canvas*.

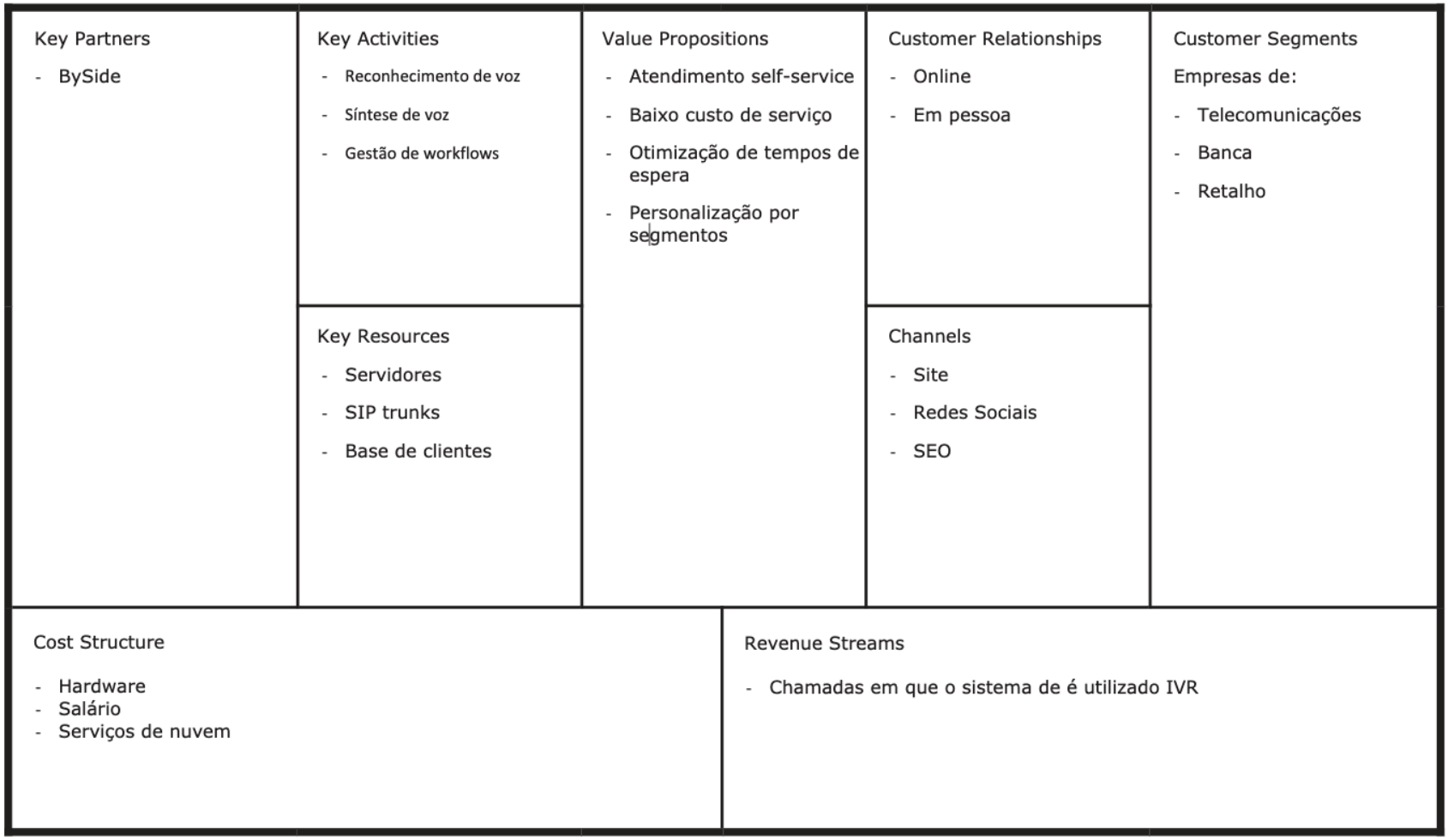

Figura 3.6: Business model canvas

Capítulo 4

Engenharia de Requisitos

Este capítulo tem como objetivo enumerar todos os requisitos funcionais e os requisitos não funcionais do sistema. Na secção dos requisitos funcionais é apresentado o diagrama de casos de uso assim como uma descrição dos casos de uso. Na secção dos requisitos não funcionais é apresentado o modelo FURPS+ para este projeto.

4.1 Requisitos funcionais

Nesta secção são apresentados os requisitos funcionais da solução, recorrendo ao uso de vários diagramas, para facilitar a interpretação e análise dos mesmos. Esta secção começa por, sucintamente definir o que é um requisito funcional e que partes estão envolvidas na sua definição. De seguida é apresentado o diagrama de casos de uso. Para finalizar, são enumerados todos os casos de uso com uma descrição com o fluxo de ações assim como um diagrama de interação de ator com sistema.

4.1.1 Definição de requisito funcional

Um requisito funcional [84] é uma descrição de ações/funções do sistema que descreve as interacções dos valores de entrada com os valores produzidos pelo sistema. Um requisito funcional possui uma parte interessada (*stakeholder*) que é o responsável por detalhar as ações e resultados esperados quando o mesmo interage com o sistema. Com o objetivo de obter uma visão global de todos os requisitos funcionais é normalmente utilizado um diagrama de casos uso, onde cada ator é um *stakeholder* e cada balão descreve de forma sucinta a ação produzida pelo ator no sistema.

4.1.2 Diagrama de casos de uso

Na figura 4.1 é apresentado o diagrama de casos de uso para o sistema em análise onde se pode observar os requisitos de cada actor.

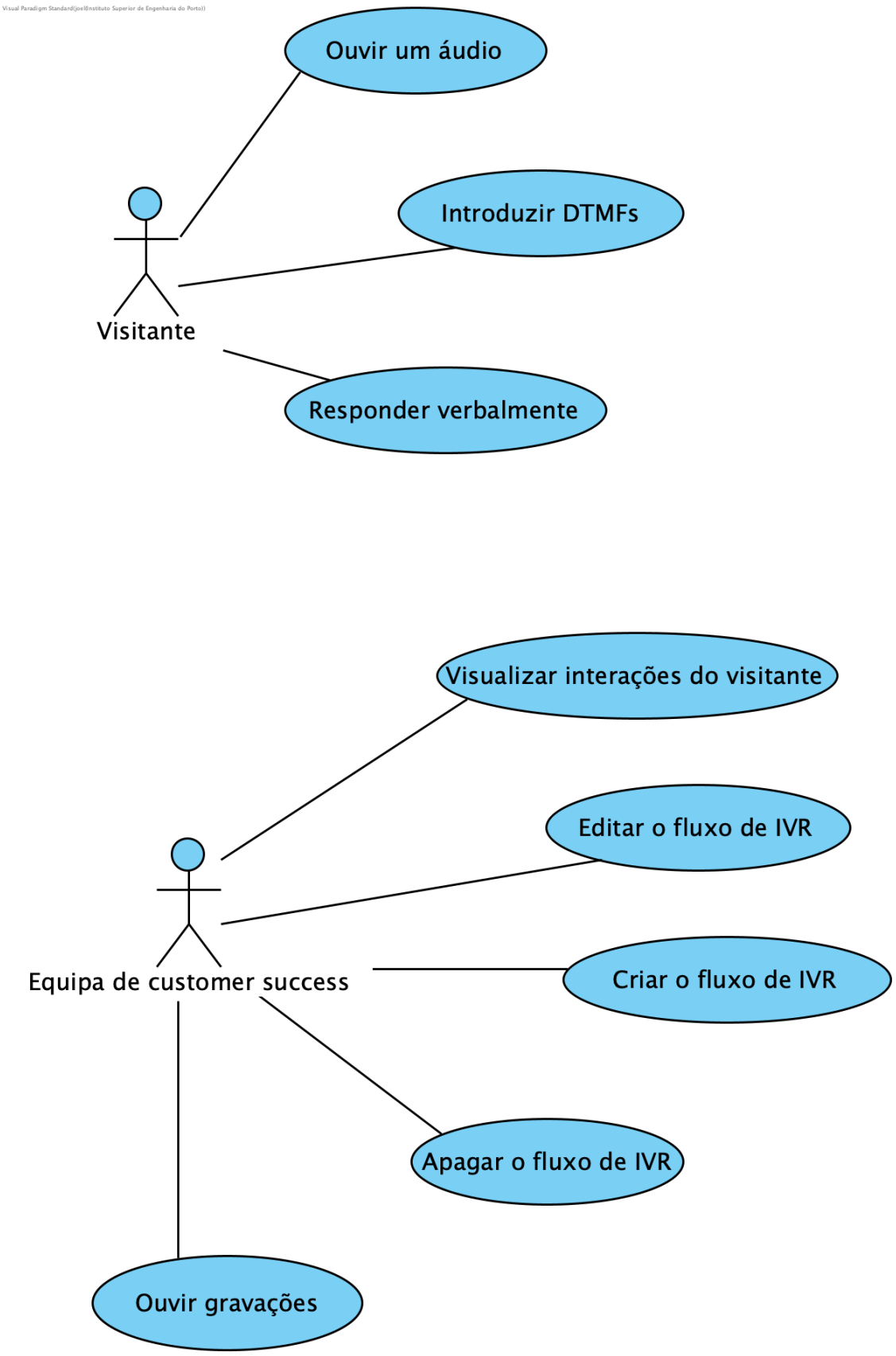

Figura 4.1: Diagrama de casos de uso

4.1.3 Visitante

Neste ponto é apresentado todos os requisitos funcionais para o visitante.

UC1 - Ouvir um áudio

O requisito funcional de ouvir um áudio consiste em permitir ao visitante ouvir um áudio, em que a origem do áudio pode ser um ficheiro previamente armazenado ou um áudio gerado recorrendo ao uso de *text-to-speech*.

Pré-condições:

- O visitante deve ter uma chamada ativa
- O ficheiro de áudio tem que estar disponível para ser reproduzido ou as configurações de *text-to-speech* devem estar corretas.

Fluxo de eventos

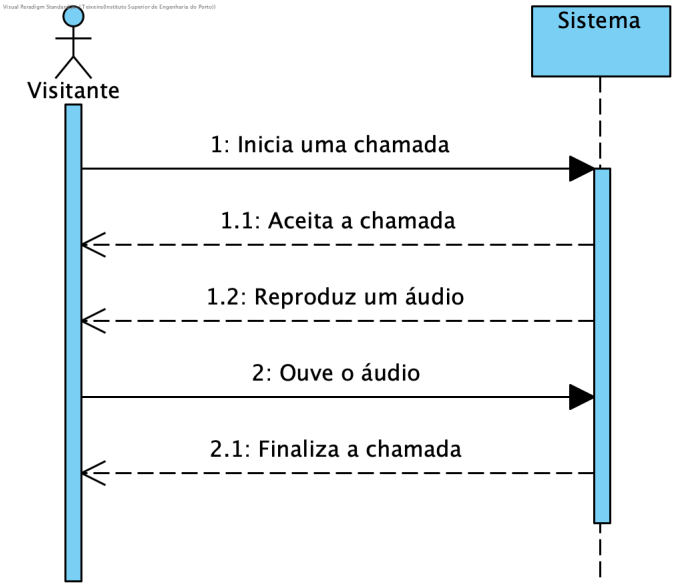

Figura 4.2: Fluxo de eventos para o requisito ouvir um áudio

Pós-condições

- É guardada uma interação do visitante na base de dados

UC2 - Introduzir DTMFs

O requisito funcional de introduzir um DTMF permite ao visitante durante a chamada de introduzir dígitos para selecionar opções ou para passar informação (e.g. número de conta de cliente, número de contribuinte).

Pré-condições:

- O visitante deve ter uma chamada ativa

Fluxo de eventos

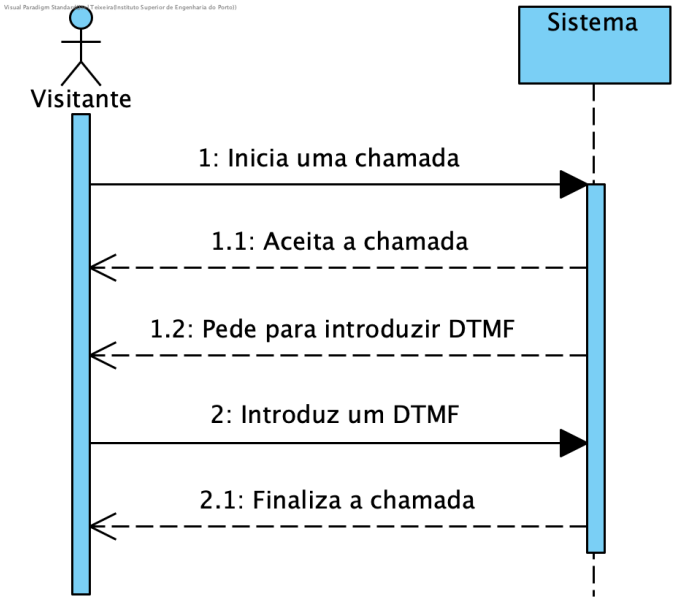

Figura 4.3: Fluxo de eventos para o requisito introduzir DTMFs

Pós-condições

- É guardada uma interação do visitante na base de dados

UC3 - Responder verbalmente

O requisito funcional de responder verbalmente permite ao visitante responder a perguntas usando linguagem natural. Através do uso do *speech-to-text* é possível converter a resposta dada pelo visitante em texto e analisar o mesmo para entender a intenção e a informação passada pelo mesmo.

Pré-condições:

- O visitante deve ter uma chamada ativa
- As configurações de *speech-to-text* devem estar corretas.

Fluxo de eventos

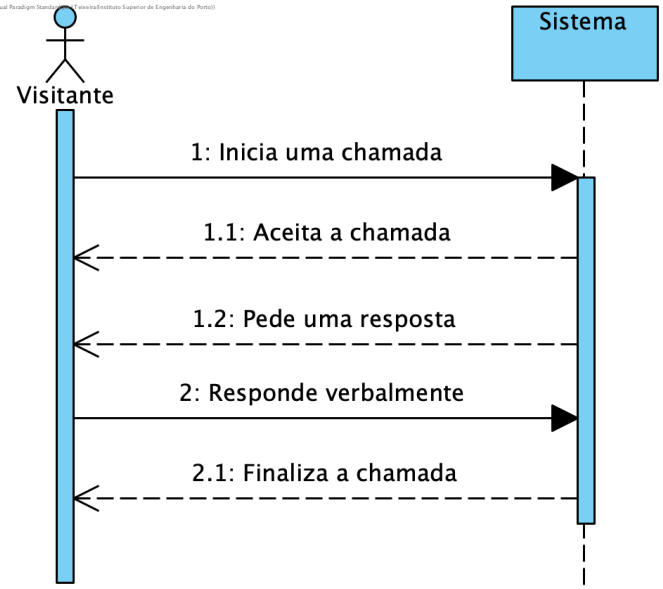

Figura 4.4: Fluxo de eventos para o requisito responder verbalmente

Pós-condições

- É guardada uma interação do visitante na base de dados

4.1.4 Equipa de customer success

Neste ponto é apresentado todos os requisitos funcionais para a equipa de *customer success*.

UC4 - Criar um fluxo de IVR

O requisito funcional de criar um fluxo de IVR permite à equipa de *customer success* configurar um fluxo de IVR. Isto permite definir os áudios, as opções e perguntas a serem feitas aos visitantes.

Pré-condições:

- A equipa de *customer success* tem que ter um utilizador registado no sistema com permissões para criar um fluxo de IVR.

Fluxo de eventos

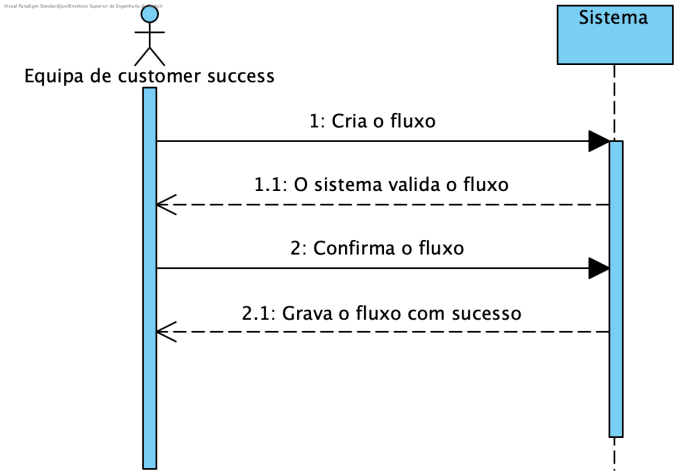

Figura 4.5: Fluxo de eventos para o requisito criar um fluxo de IVR

Pós-condições

- O fluxo fica guardado na base de dados

UC5 - Editar um fluxo de IVR

O requisito funcional de editar um fluxo de IVR permite à equipa de *customer success* atualizar um fluxo de IVR já existente. De forma fácil será possível redefinir os áudios, opções e perguntas feitas a um visitante.

Pré-condições:

- A equipa de *customer success* tem que ter um utilizador registado no sistema com permissões para editar um fluxo de IVR.
- O fluxo deve existir no sistema.

Fluxo de eventos

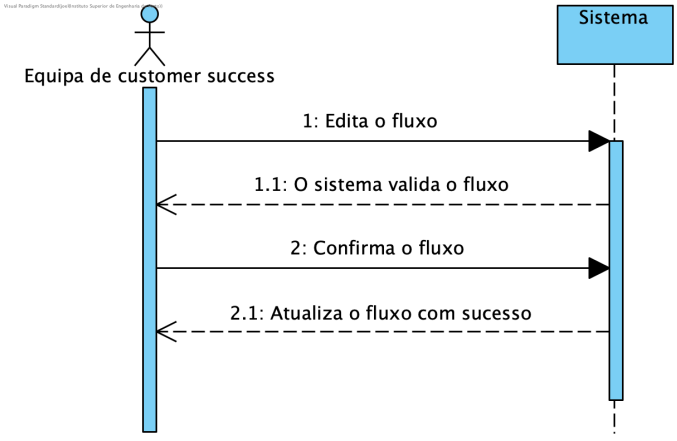

Figura 4.6: Fluxo de eventos para o requisito editar um fluxo de IVR

Pós-condições

- É atualizado o fluxo previamente guardado na base de dados

UC6 - Apagar o fluxo de IVR

O requisito funcional de apagar um fluxo de IVR permite à equipa de *customer success* apagar um fluxo de IVR já existente, ou seja, o fluxo deixa de estar disponível no sistema.

Pré-condições:

- A equipa de *customer success* tem que ter um utilizador registado no sistema com permissões para apagar um fluxo de IVR.
- O fluxo deve existir no sistema.

Fluxo de eventos

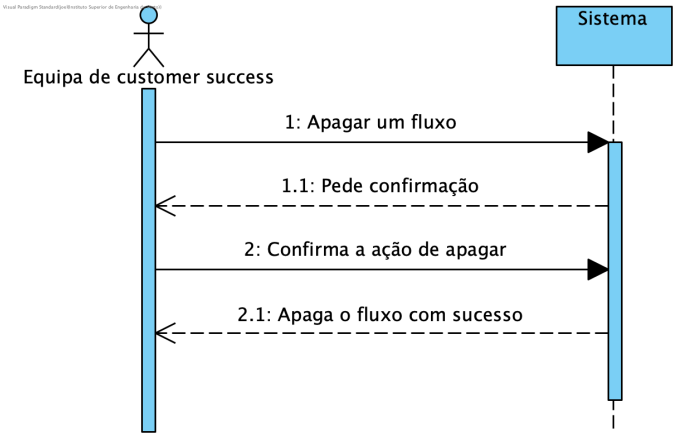

Figura 4.7: Fluxo de eventos para o requisito apagar um fluxo de IVR

Pós-condições

- O fluxo é apagado da base de dados

UC7 - Ouvir gravações

O requisito funcional de ouvir gravações permite à equipa de *customer success* ouvir todas as interações do visitante com um fluxo de IVR. O objetivo deste requisito é disponibilizar um meio de identificar pontos fracos do fluxo de IVR, para que o mesmo seja melhorado.

Pré-condições:

- A equipa de *customer success* tem que ter um utilizador registado no sistema com permissões para ouvir as gravações.
- A chamada ter sido gravada desde que o visitante começou a interagir com o fluxo de IVR
- Os ficheiros de gravação estarem guardados para serem consultados.

Fluxo de eventos

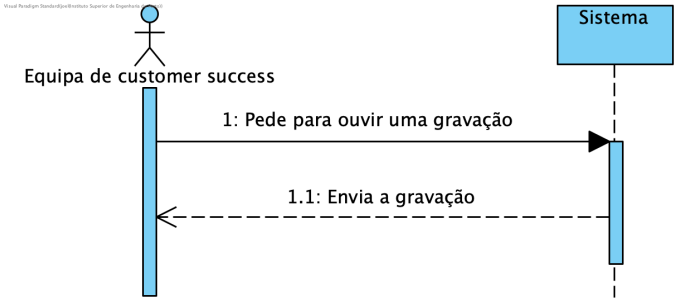

Figura 4.8: Fluxo de eventos para o requisito ouvir gravações

Pós-condições

- N/A

UC8 - Visualizar as interações dos visitantes durante um IVR

O requisito funcional de visualizar as interações dos visitantes durante um IVR permite à equipa de *customer success* observar todas a interações de um visitante com um fluxo de IVR. Deve ser possível verificar os valores introduzidos pelo visitante durante as interações.

Pré-condições:

- A equipa de *customer success* tem que ter um utilizador registado no sistema com permissões para visualizar as interações.

Fluxo de eventos

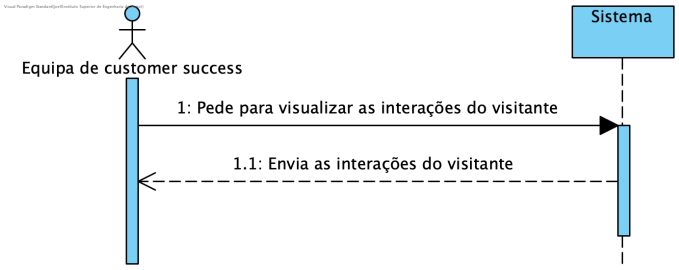

Figura 4.9: Fluxo de eventos para o requisito visualizar as interações do visitante num IVR

Pós-condições

- N/A

4.2 Requisitos não funcionais

Para representar os requisitos não funcionais do sistema foi utilizado o modelo FURPS+. É dada uma breve abordagem de como este modelo funciona. De seguida são apresentados os requisitos não funcionais do sistema em análise enquadrados nas várias categorias do FURPS+.

4.2.1 FURPS+

FURPS [85] é um acrónimo para (*Functionality, Usability, Reliability, Performance, Supportability*), desenvolvido por Robert Grady, com o objetivo de criar e melhorar o processo de identificação de requisitos não funcionais para um sistema. Este modelo foi revisitado posteriormente para incluir mais categorias que especificam mais alguns dos requisitos não funcionais. As novas categorias adicionadas incluem: Requisitos de design, Requisitos de implementação, Requisitos de interface, Requisitos físicos.

4.2.2 Requisitos não funcionais do sistema

Funcionalidade

- A configuração de um fluxo de IVR deve ser genérica e uniforme;
- Todos os nós de um fluxo devem suportar vários idiomas;
- Devem ser gravadas as ações dos utilizadores e estatísticas durante o IVR;
- Em cada nó deve ser possível aceder a informações de contexto (e.g. dados do visitante, ações em todos os nós);
- A gestão dos fluxos deve ser controlada e gerida tendo em conta o perfil do utilizador.

Usabilidade

- N/A

Confiabilidade

- O sistema deve ser robusto para conseguir lidar com erros provenientes de serviços externos.

Desempenho

- O sistema deve conseguir processar 100 IVRs em simultâneo.

Suportabilidade

- A cobertura de testes unitários deve ser 100%;
- Deve incluir monitorização para avaliar a saúde do sistema e deve ser integrada com o sistema atualmente usado na BySide;
- O sistema deve escalar horizontalmente de forma automática;
- Fazer integração com outros produtos BySide.

Requisitos de design

- N/A

Requisitos de implementação

- Adoção de Node.JS;
- O fluxo de IVR é carregado para memória do motor quando a chamada é iniciada.

Requisitos de interface

- Utilização do serviço Asterisk para controlo dos IVRs.

Requisitos físicos

- N/A

4.3 Modelo de domínio

O conceito de modelo de domínio pode ser definido como a representação apropriada de vários conceitos inerentes a um determinado problema ou negócio e a relação estabelecida entre eles. Segundo Eric Evans [86], o modelo de domínio poder ser defini como uma "*rigorously organized and selective abstraction of the knowledge in a domain expert's head*".

No contexto desta dissertação, o modelo de domínio representa os vários conceitos em torno de um IVR. Um IVR é um fluxo de trabalho (*workflow*), mas ao mesmo tempo também é uma tecnologia do mundo das telecomunicações, mais especificamente das telecomunicações por voz. O objetivo deste modelo de domínio é documentar a interligação entre os dois conceitos: os *workflows* e as tecnologias de voz.

A figura 4.10 apresenta os vários elementos de domínio e as ligações entres eles.

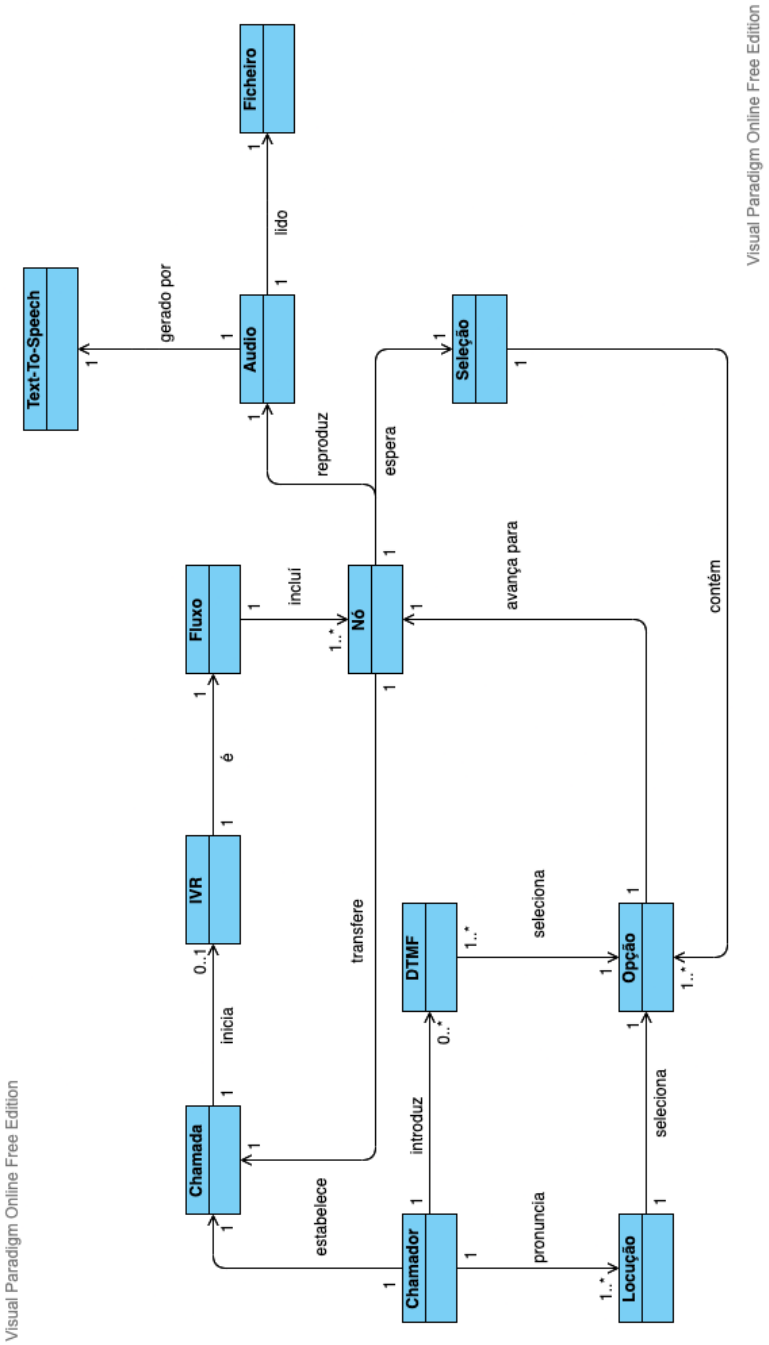

Figura 4.10: Modelo de domínio

Capítulo 5

Desenho da solução

Nesta secção são apresentadas as várias alternativas de solução para resolver o problema exposto anteriormente. Depois de analisadas várias alternativas de arquitetura para resolver o problema, será feito o design dos casos de uso com base na arquitetura escolhida. Para ajudar à compreensão das várias alternativas de solução e do design dos casos de uso são utilizados vários tipos de diagramas.

5.1 Arquitetura da solução

A definição mais comum e adotada pela maioria da comunidade de engenheiros de software é de que arquitetura de software [87] é descrição de todos os componentes importantes de um sistema assim como as suas relações e de que forma cada um desses componentes interage entre si.

Contudo, Ralph Johnson e Martin Fowler [88] defendem que arquitetura de software não é um conceito fechado, mas sim o conhecimento e a percepção conjunta de vários engenheiros de como um sistema funciona.

"Architecture is about the important stuff. Whatever that is", disse Ralph Johnson [88] e Fowler complementa esta afirmação, expondo que a capacidade de decidir o que é importante deve consequentemente ser a área onde toda a energia deve ser gasta, para que os elementos integrantes da arquitetura continuem em boas condições.

5.1.1 Propostas de arquitetura

Com base nas várias posições em contraste sobre arquitetura de software, foram desenhadas várias soluções de arquitetura para resolver o problema central desta dissertação. Cada proposta apresenta os elementos essenciais do sistema é acompanhada de um diagrama de componentes e de um diagrama de implantação, além disso é enumerado os prós e contras de cada solução. No final é indicada qual a solução que faz sentido a ser implementada no contexto de negócio tendo em conta todos os requisitos funcionais e não funcionais enumerados no capítulo 4.

5.1.2 Premissas

Aqui estão enumeradas algumas das premissas base para desenho do sistema de IVRs.

- No sistema de voz da BySide todas as chamadas são entregues na aplicação Drachtio que encaminha as mesmas para um dos Asterisks configurados. O Drachtio neste contexto não é um proxy, mas sim um B2BUA (*Back-to-Back User Agent*.

- A persistência de dados é feita recorrendo ao motor de persistência MySQL.
- A BySide não utiliza tecnologia de *containers* em produção.
- A BySide utiliza ferramentas de provisionamento automático de VMs (Ansible).

Primeira proposta

A primeira proposta de implementação tem como objetivo permitir escalabilidade horizontal dos serviços de forma independente. O ponto de entrada no sistema de IVRs é feito através da SBC, que é responsável por gerir o tráfego de chamadas para as diversas VMs de voz presentes no sistema.

Todos os serviços para o funcionamento do sistema estão isolados e disponibilizam interfaces de comunicação bem definidas. O isolamento é obtido através da disponibilização de VMs para cada componente do sistema.

Aplicando esta proposta estamos a promover uma alta coesão e um baixo acoplamento dos componentes do sistema.

A resiliência do sistema é assegurada pelo uso de proxies/balanceadores de carga, onde a sua principal função passa por assegurar a distribuição de tráfego uniforme pelos vários serviços que constituem o sistema. O sistema com a implementação desta proposta tem a possibilidade de ser elástico de forma automática baseado em métricas de qualidade de serviço.

A utilização dos proxies/balanceadores carga é um fator que contribui para o baixo acoplamento e alta coesão do sistema

A figura 5.1 apresenta o diagrama de componentes para a primeira proposta. Com este diagrama é possível visualizar todos os componentes e as suas interfaces de comunicação.

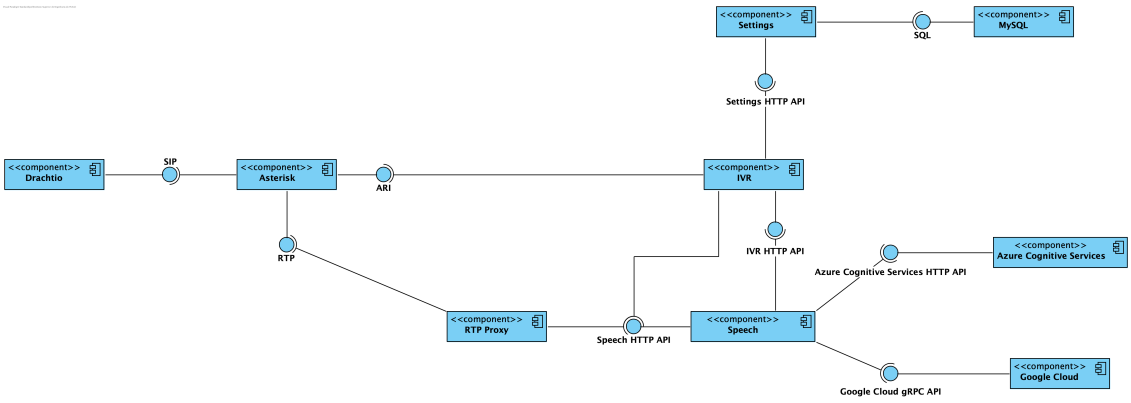

Figura 5.1: Diagrama de componentes da proposta 1

A figura 5.2 apresenta o diagrama de implantação da primeira proposta assim como todos os tipos de ligação presentes entre os várias instâncias do sistema.

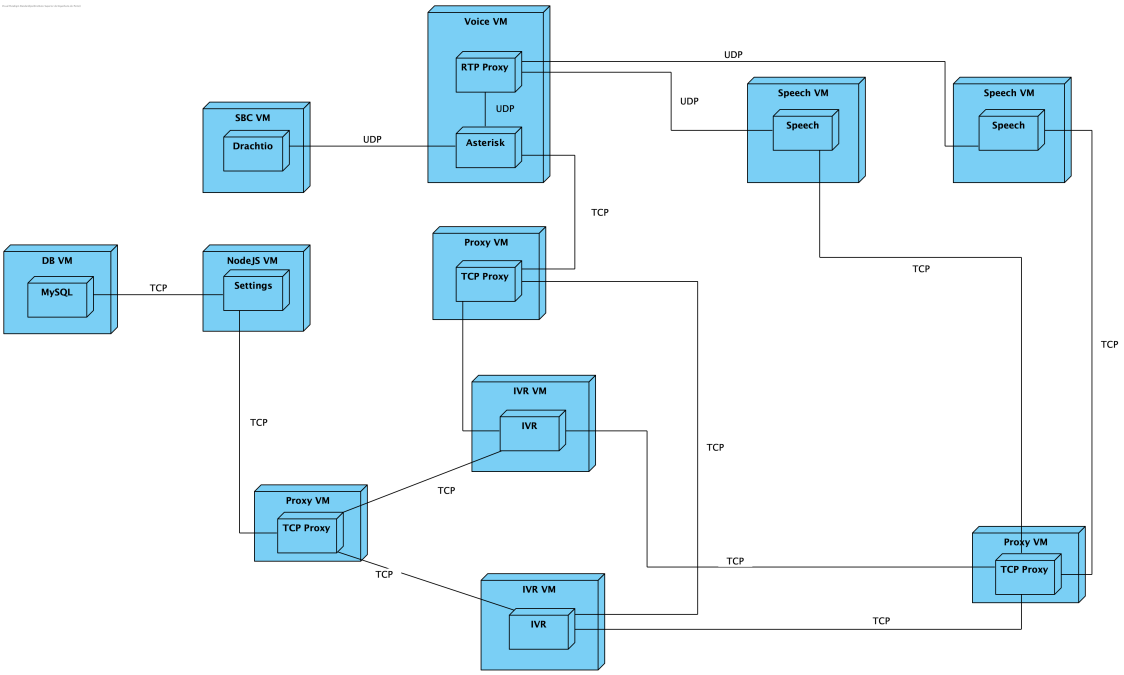

Figura 5.2: Diagrama de implantação da proposta 1

Prós

- Escalabilidade orientada a serviços
- Resiliência

Contras

- Maior latência no processamento de ações
- Maior complexidade do sistema
- Maior infraestrutura para gerir

Segunda proposta

A segunda proposta de implementação tem como objetivo permitir escalabilidade horizontal mas reduzindo alguns elementos de infraestrutura (e.g. proxies/balanceadores de carga). O ponto de entrada no sistema de IVRs é feito através da SBC, que é responsável por gerir o tráfego de chamadas para as diversas VMs de voz presentes no sistema.

De forma a reduzir a quantidade de infraestrutura necessária no sistema, esta proposta assume que a escalabilidade do sistema será feita tendo em conta infraestrutura em vez de serviços, ou seja os componentes Asterisk, IVR e Speech são alojados na mesma VM. O sistema será escalado tendo em conta o consumo dos recursos da VM. Nesta solução existe na mesma a separação de responsabilidades dos vários serviços, a grande vantagem é a diminuição de infraestrutura que tem que ser gerida.

A resiliência é assegurada pela redundância de VMs de voz em que cada uma contêm três componentes (Asterisk, IVR e Speech), o balanceamento do tráfego é assegurado pela SBC. O sistema consegue ser na mesma elástico, contudo não existe camadas distintas entre os vários componentes do sistemas em termos de implantação. O que pode contribuir para uma má gestão dos recursos de infraestrutura.

Tendo em conta que o sistema que se encontra a ser desenhado é um sistema de comunicações em real-time é bastante vantajoso para estes sistemas que a latência de comunicação entre os serviços seja bastante reduzida. Esta proposta ajuda a que as latências entre serviços sejam reduzidas, uma vez que todos os serviços essenciais se encontram alojados na mesma VM.

A figura 5.3 apresenta o diagrama de componentes para a segunda proposta. Com este diagrama é possível visualizar todos os componentes e as suas interfaces de comunicação.

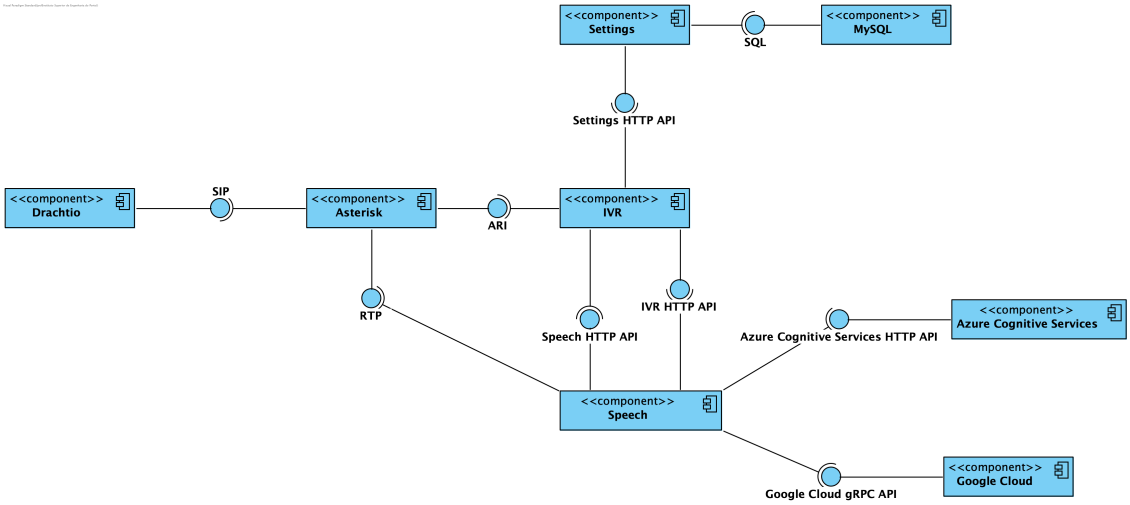

Figura 5.3: Diagrama de componentes da proposta 2

A figura 5.4 apresenta o diagrama de implantação da segunda proposta assim como todos os tipos de ligação presentes entre os várias instâncias do sistema.

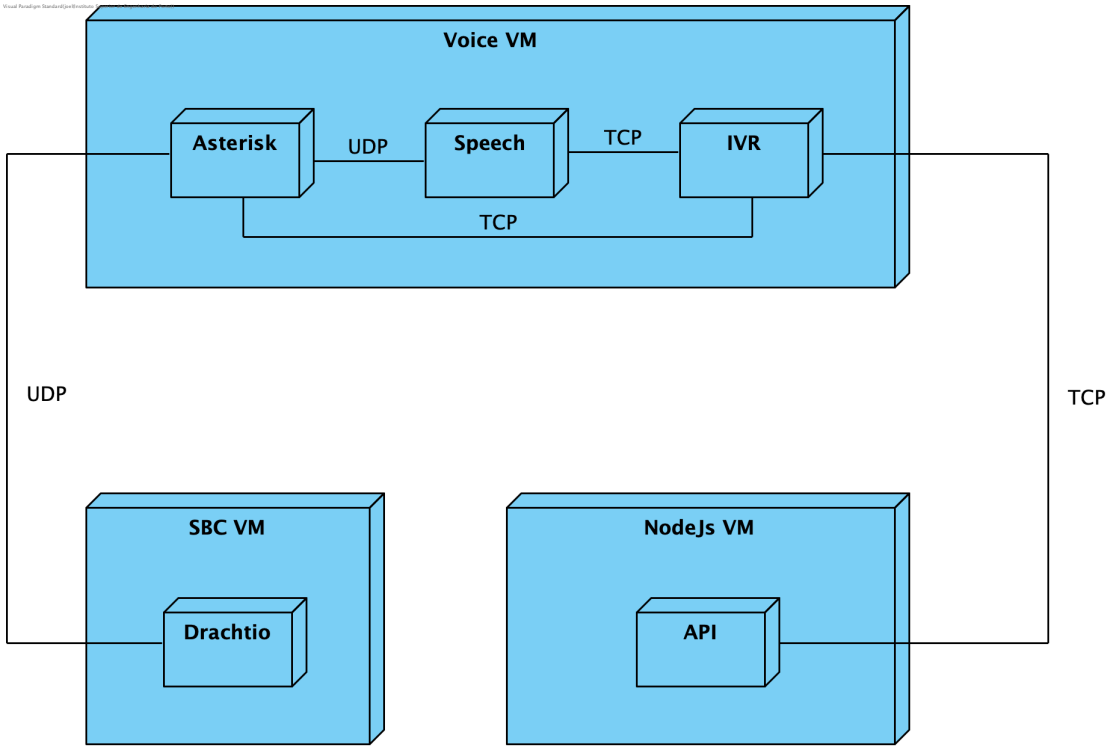

Figura 5.4: Diagrama de implantação da proposta 2

Prós

- Latência reduzida
- Resiliência
- Menor gestão de infraestrutura
- Menor complexidade do sistema

Contras

- Escalabilidade orientada a infraestrutura em vez de serviços
- Má otimização de recursos

Terceira proposta

A terceira proposta de implementação baseia se em um modelo de arquitetura orientada a eventos. O ponto de entrada no sistema de IVRs é feito através da SBC, que é responsável por gerir o tráfego de chamadas para as diversas VMs de voz presentes no sistema.

Nesta proposta é introduzido um *message broker* que disponibiliza várias filas para segregar por tipo os eventos emitidos pelos *publishers*. Em reverso, os consumidores recebem eventos de filas específicas. Seguindo esta lógica conseguimos aplicar separação de responsabilidades de forma eficiente.

Com a introdução de um *message broker* criamos uma camada de abstração que nos permite obter escalabilidade orientada a serviços, ou seja, cada componente essencial do sistema consegue escalar de forma independente.

A resiliência do sistema fica garantida se existir redundância nos serviços, ou seja, existir várias instâncias ativas no sistema de cada componente. No caso do *message broker* este estaria numa versão cluster, logo caso um dos nós do cluster ficar inativo os outros nós asseguram o processamento dos eventos das filas do nó que ficou inativo.

Apesar desta proposta em termos de infraestrutura conter menos elementos que a primeira proposta, existe na mesma um grau de latência elevado associado ao processamento dos eventos. Para o contexto do sistema, a latência é algo que deve ser considerado uma vez que é um sistema em tempo real. Caso uma das filas fique cheia não será possível executar ações sobre as chamadas instantaneamente.

A figura 5.5 apresenta o diagrama de componentes para a terceira proposta. Com este diagrama é possível visualizar todos os componentes e as suas interfaces de comunicação.

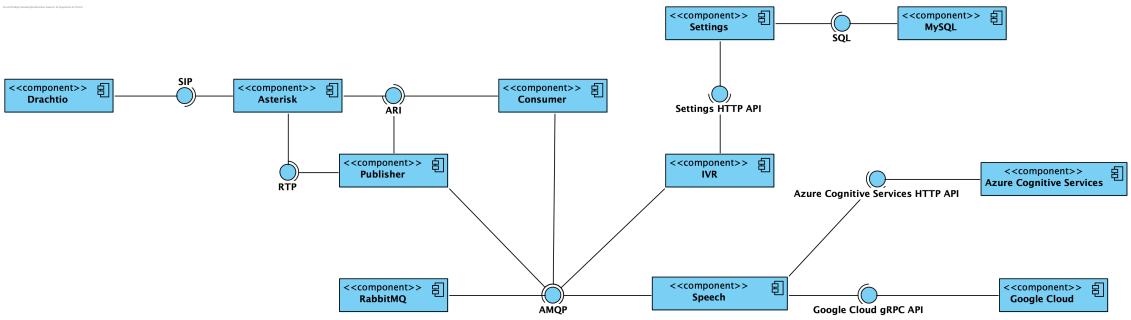

Figura 5.5: Diagrama de componentes da proposta 3

A figura 5.6 apresenta o diagrama de implantação da terceira proposta assim como todos os tipos de ligação presentes entre os várias instâncias do sistema.

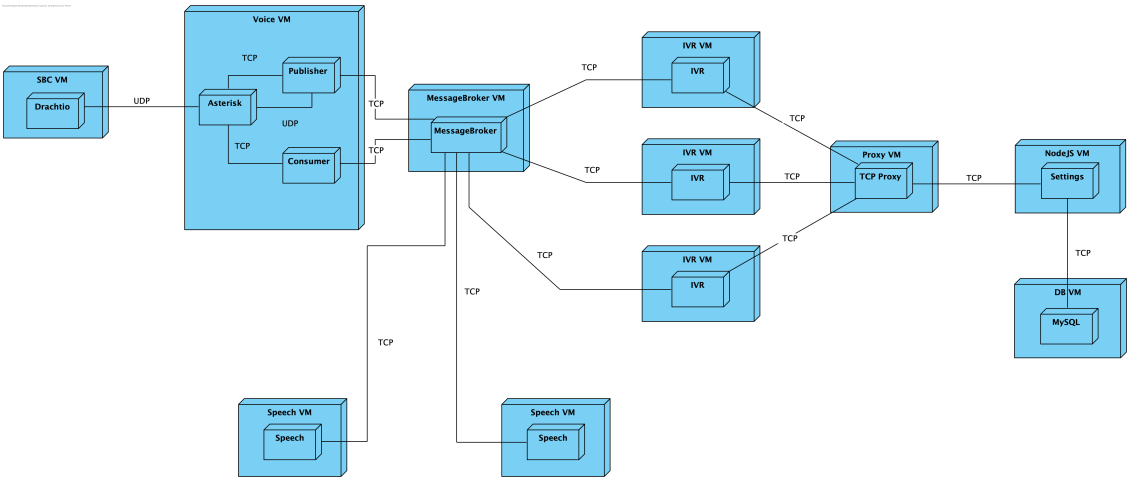

Figura 5.6: Diagrama de implantação da proposta 3

Prós

- Escalabilidade orientada a serviços
- Resiliência
- Menor gestão de infraestrutura

Contras

- Maior latência no processamento de ações
- Maior complexidade do sistema

Escolha da proposta

Decidir um modelo de arquitetura não é uma tarefa fácil. A arquitetura [89] é desenhada com base nos requisitos atuais do negócio sem ter em conta que os requisitos estão em constante mudança. Ralph Johnson [88] afirma que arquitetura é: *"the decisions you wish you could get right early in a project"*.

Tendo em conta os requisitos atuais do negócio e sabendo que o sistema tem características de um sistema de tempo real, o aspeto mais importante em consideração é a latência de como as ações são produzidas.

Posto isto, a proposta que mais se adequa e que cumpre de forma eficaz estes objetivos é a segunda proposta, logo o sistema será modelado e construído com base nessa arquitetura.

5.2 Desenho dos casos de uso

Com base na arquitetura escolhida na secção anterior e recorrendo ao uso de diagramas de *package*, diagramas de sequência e diagramas de classe, nesta secção é exposto o design dos casos de uso recolhidos no capítulo 4

5.2.1 Diagramas de sequência

Os diagramas de sequência a seguir apresentados tem como objetivo ilustrar o processo de funcionamento de cada caso de uso anteriormente exposto no capítulo 4.

Antes de iniciar um IVR, os casos de uso associados ao visitante devem carregar a configuração de execução dos mesmos. A configuração pode ser consultada através da API do componente Settings. A figura 5.7 representa o processo de obtenção da configuração do IVR no componente Settings.

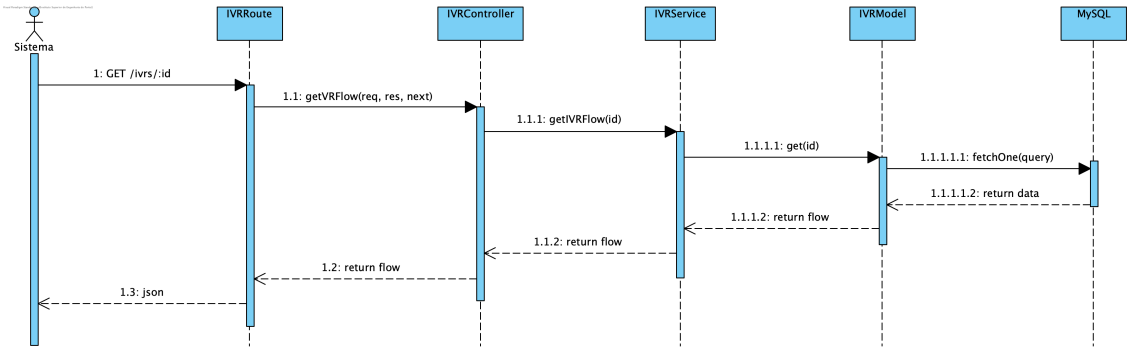

Figura 5.7: Diagrama de sequência obter a configuração de um IVR

UC1 - Ouvir um áudio

Neste caso de uso o objetivo é que o visitante consiga ouvir um áudio. O áudio pode ser gerado através de *text-to-speech* ou ser proveniente de um ficheiro. A figura 5.9 representa o processo de que leva à reprodução do áudio para o visitante.

No caso de o áudio ser gerado através de *text-to-speech*, é feito um pedido à API do componente Speech. No pedido é passado o texto e as configurações necessárias para obter o áudio gerado. A figura 5.9 ilustra o processo de obtenção do áudio gerado por *text-to-speech*.

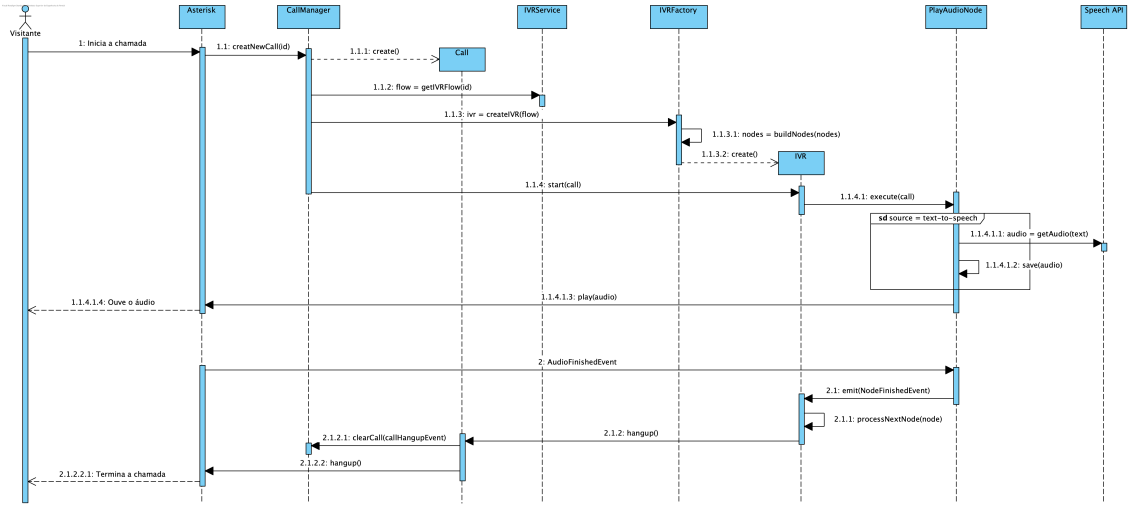

Figura 5.8: Diagrama de sequência de ouvir um áudio

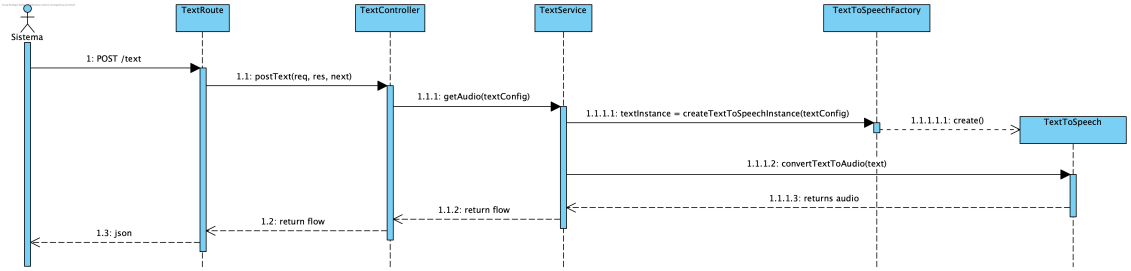

Figura 5.9: Diagrama de sequência para gerar um áudio

UC2 - Introduzir DTMFs

Neste caso de uso o objetivo é o visitante conseguir introduzir DTMFs para poder escolher opções ou passar informação numérica. A figura 5.10 representa o processo de introdução de DTMFs por parte do visitante.

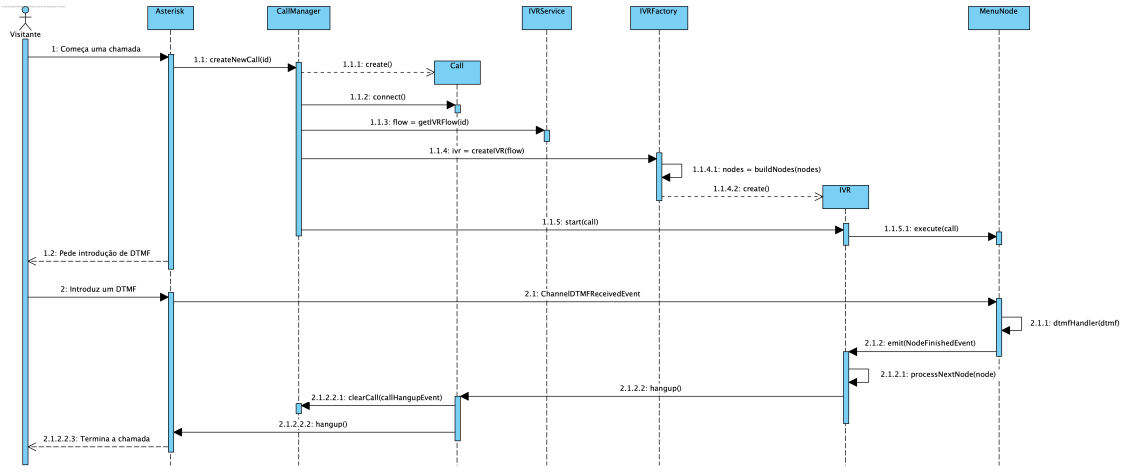

Figura 5.10: Diagrama de sequência para introduzir DTMFs

UC3 - Responder verbalmente

Neste caso de uso o objetivo é o visitante conseguir responder verbalmente, ou seja, com este caso de uso é possível o visitante escolher opções e responder a questões utilizando a sua própria voz. A figura 5.11 representa o processo de resposta verbal por parte do visitante.

Para ser possível fazer o reconhecimento de voz é necessário fazer um pedido ao projeto Speech para configurar o reconhecimento com os serviços externos de reconhecimento de voz. A figura 5.12 representa o processo para configurar o reconhecimento de voz.

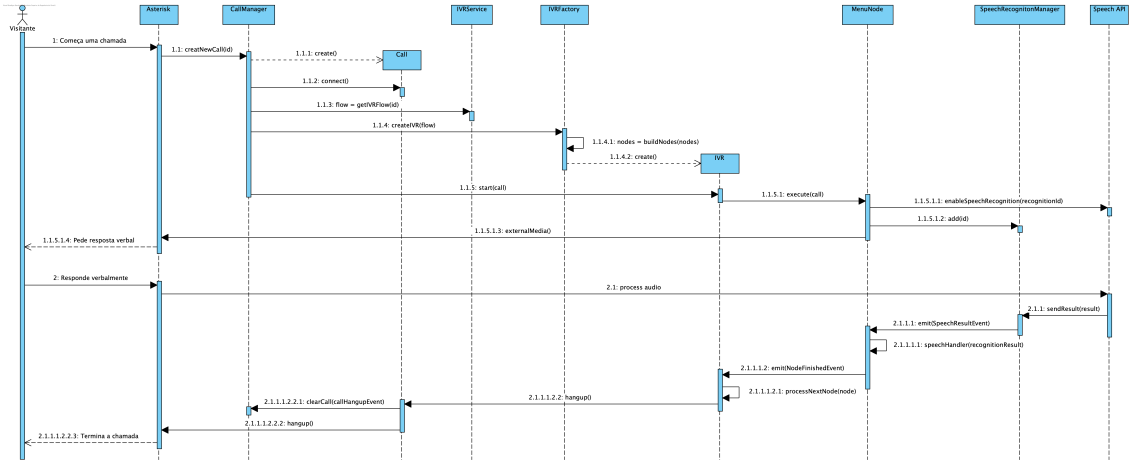

Figura 5.11: Diagrama de sequência para responder verbalmente

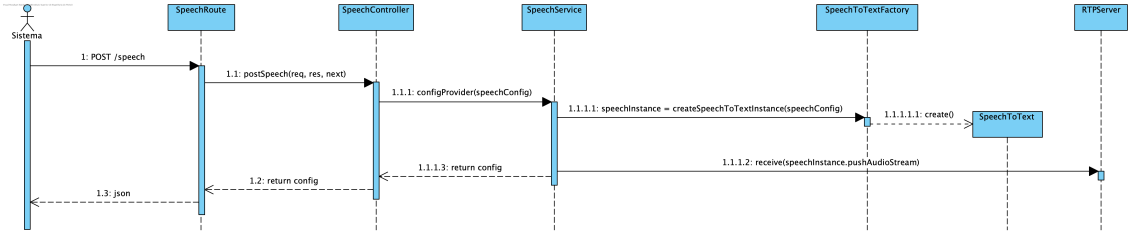

Figura 5.12: Diagrama de sequência para configurar o reconhecimento de voz

UC4 - Criar um fluxo de IVR

Neste caso de uso o objetivo é criar um fluxo de IVR e persistir a configuração na base de dados. A figura 5.13 representa o processo de criação e persistência do fluxo.

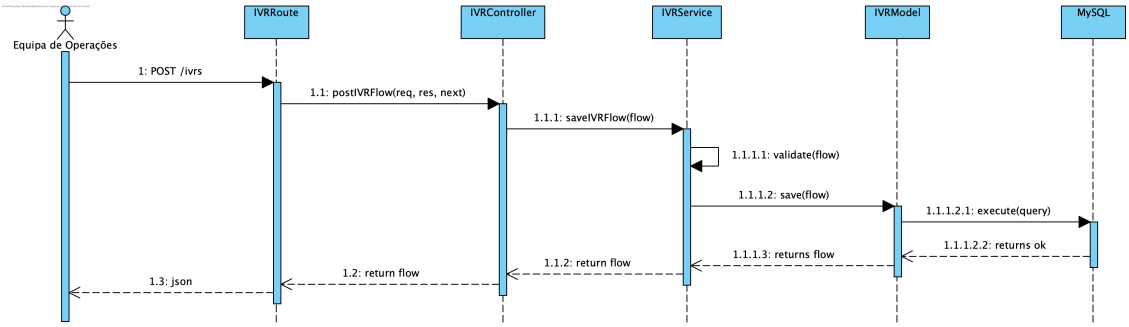

Figura 5.13: Diagrama de sequência para criar um fluxo de IVR

UC5 - Editar um fluxo de IVR

Neste caso de uso o objetivo é editar um fluxo de IVR previamente configurado e persistir a nova configuração na base de dados. A figura 5.14 representa o processo de atualização e persistência das novas configurações.

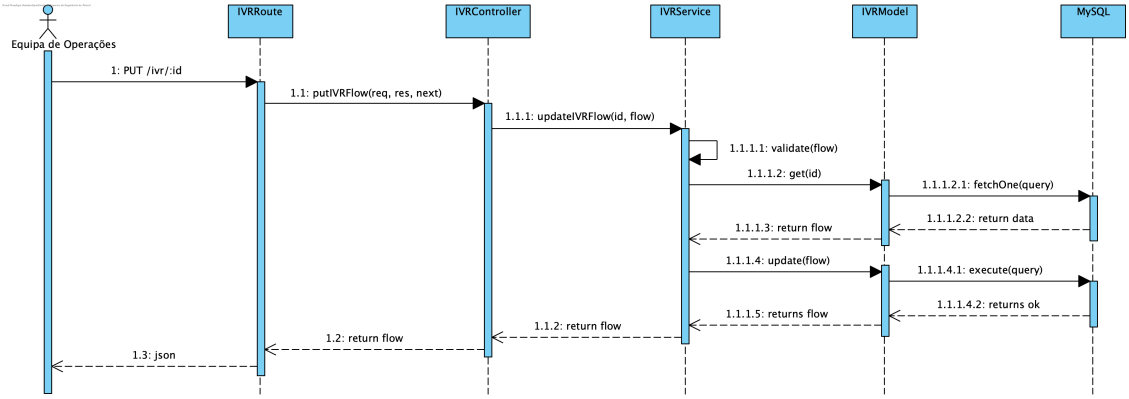

Figura 5.14: Diagrama de sequência para editar um fluxo de IVR

UC6 - Apagar o fluxo de IVR

Neste caso de uso o objetivo é apagar um fluxo de IVR previamente configurado e persistido. A figura 5.15 representa o processo de remoção do fluxo.

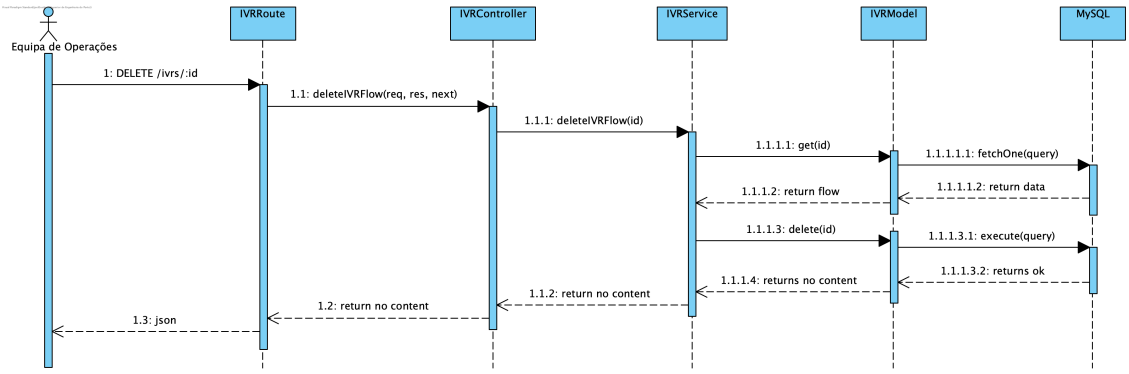

Figura 5.15: Diagrama de sequência para apagar um fluxo de IVR

UC7 - Ouvir gravações

Neste caso de uso o objetivo é obter uma gravação de uma chamada em que um fluxo de IVR foi executado. A figura 5.16 representa o processo de obtenção da gravação de uma chamada.

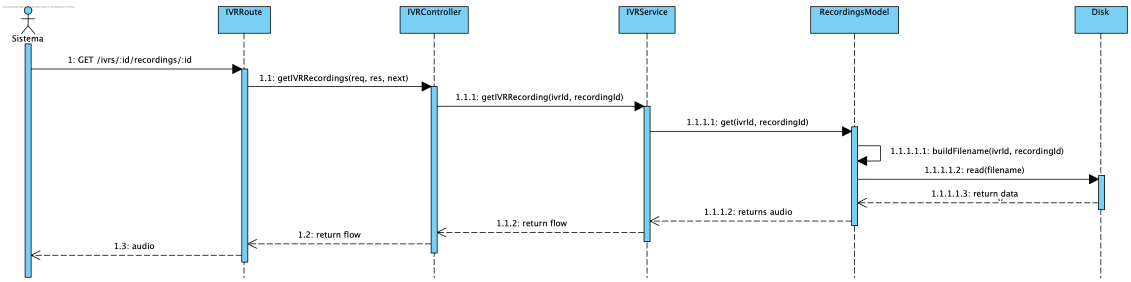

Figura 5.16: Diagrama de sequência para ouvir uma gravação de um IVR

UC8 - Visualizar as interações dos visitantes durante um IVR

Neste caso de uso o objetivo é conseguir visualizar as interações dos visitantes durante um IVR. A figura 5.17 representa o processo que permite visualizar essas interações.

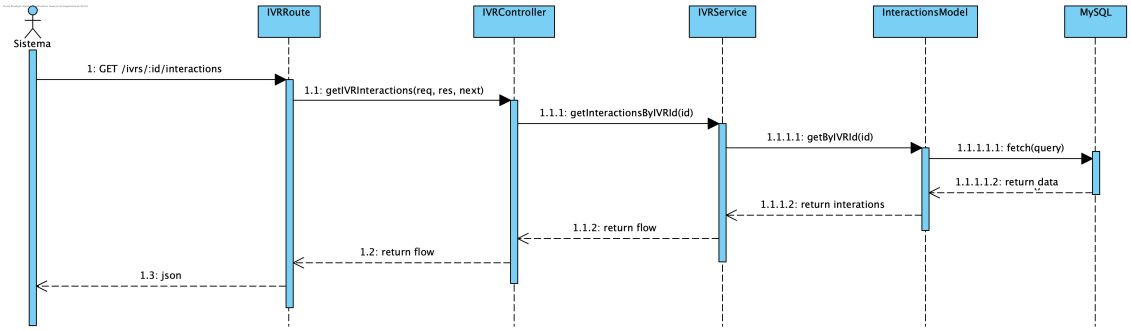

Figura 5.17: Diagrama de sequência para visualizar as interações dos visitantes durante um IVR

5.2.2 Diagramas de classe

Os diagramas de classe a seguir apresentados tem como principal objetivo especificar as funções apresentadas nos diagramas de sequência. Os diagramas estão divididos por componente e mostram as dependências entre cada classe.

IVR

A figura 5.18 apresenta o diagrama de classe para o componente IVR, assim como as ligações e dependências entre as diferentes classes.

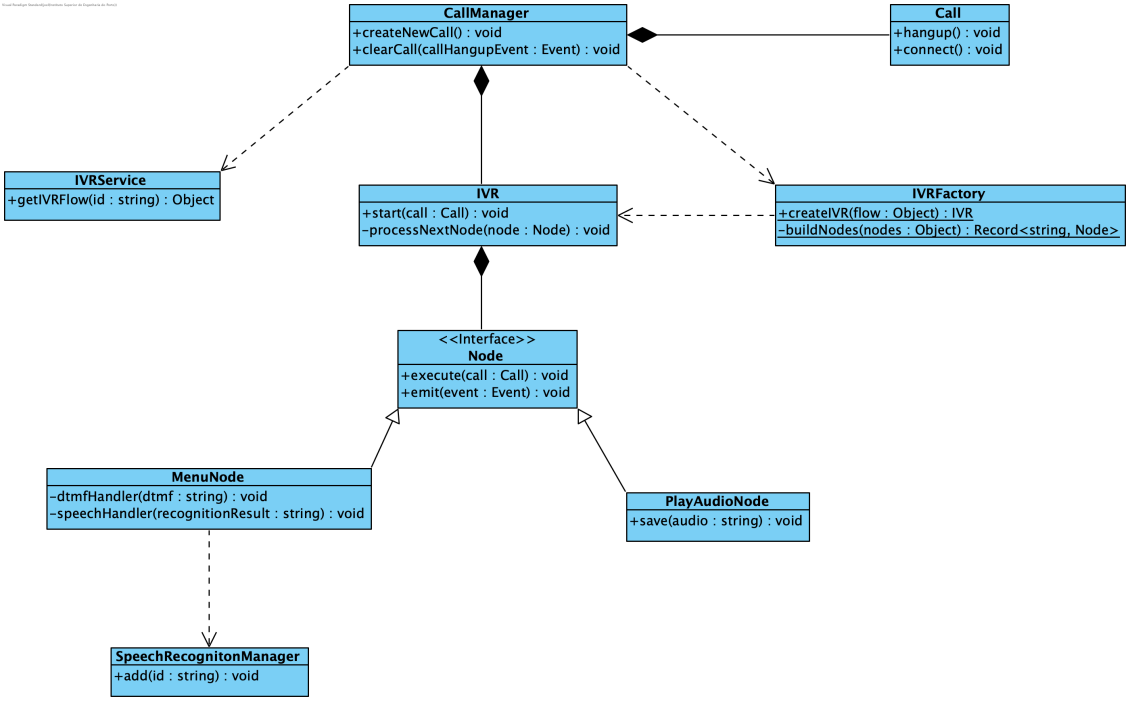

Figura 5.18: Diagrama de sequência apagar fluxo

Speech

A figura 5.19 apresenta o diagrama de classe para o componente Speech, assim como as ligações e dependências entre as diferentes classes.

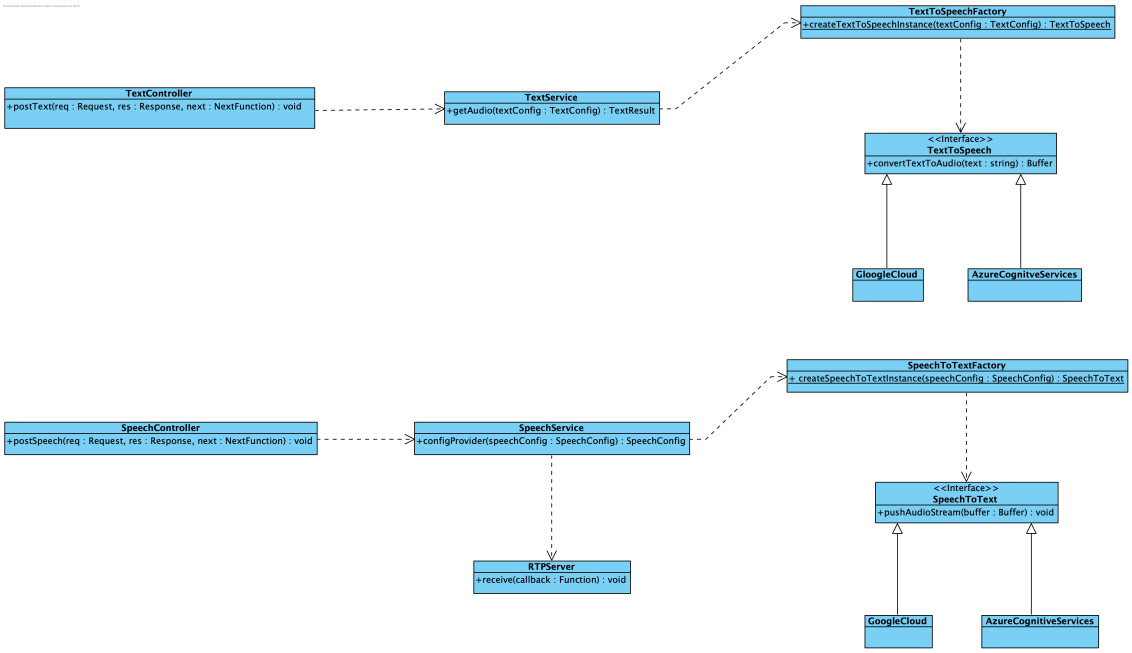

Figura 5.19: Diagrama de sequência apagar fluxo

Settings

A figura 5.20 apresenta o diagrama de classe para o componente Settings, assim como as ligações e dependências entre as diferentes classes.

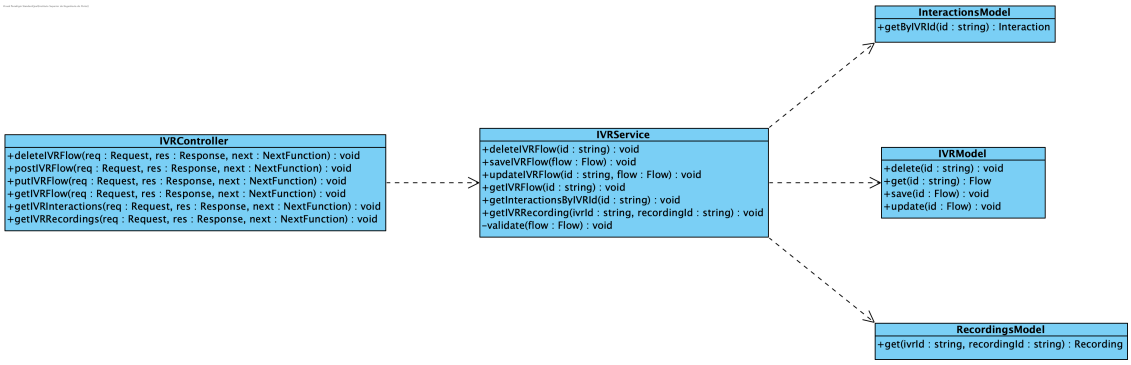

Figura 5.20: Diagrama de sequência apagar fluxo

5.2.3 Diagramas de packages

Os seguintes diagramas de *packages* ilustram a separação das classes de código pelas suas funções. Durante esta dissertação foram desenvolvidos três projetos (IVR, Speech, Settings). Existem *packages* que são equivalentes para todos os projetos. A seguir são enumerados esses *packages* equivalentes.

O *package* **api** agrega dois packages: routes e controllers. O *package* de **routes** agrega todos as classes que definem as rotas HTTP da API dos projetos. O *package* **controllers** agrega todas as classes de ligação da vista, que neste caso são as rotas HTTP, com a camada de serviço, que neste caso corresponde as todas as classes contidas no *package* **services**.

O *package* **middlewares** [90] agrega classes que tem como objetivo serem encadeadas no processamento de pedidos HTTP. Essas classes podem ser utilizadas para lidar com os erros lançados durante o processamento do pedido, fazer o registo dos pedidos para auditoria ou até mesmo fazer a validação de tokens de autenticação.

O *package* **services** [91] agrega todas as classes que coordenam as relações entre várias fontes de dados e a lógica de negócio. Tem como principal objetivo controlar transações entre várias classes de domínio. Pode ser utilizado para aceder aceder a serviços externos à aplicação (e.g. consultar informação numa API externa aplicação).

O *package* **interfaces** agrega todos os DTOs presentes na aplicação. Os DTOs [92] tem como principal objetivo agregar vários tipo de dados provenientes da camada de domínio para serem apresentados na camada de apresentação e vice-versa. Isto reduz o custo pois a informação passa a ser distribuída num pedido em vez de vários pedidos.

O *package* **errors** agrega todas as classes que são consideradas erros da camada de negócio. Estes erros serão lançados sempre que uma validação de lógica de negócio for infringida.

O *package* **utils** agrega todas a classes que disponibilizam funções utilitárias que são utilizadas por mais do que uma classe (e.g. *logger*, conversão de números, formatação de sequências alfanuméricas).

IVR

No caso do projeto IVR existem mais dois *packages*: **call** e **ivr**.

O *package* **call** agrega todas as classes que fazem o controlo da chamada, ou seja, todas as classes que conseguem executar ações perante a chamada (e.g. terminar chamada, reproduzir um áudio, conectar vários canais).

O *package* **ivr** agrega todas as classes responsáveis por gerir o fluxo de interação durante um IVR. O *package* **nodes**, contido dentro do *package* **ivr**, agrega todas as classes com a implementação dos vários tipos de interação.

A figura 5.21 representa o diagrama de *packages* para o projeto IVR e as ligações entre os vários *packages*.

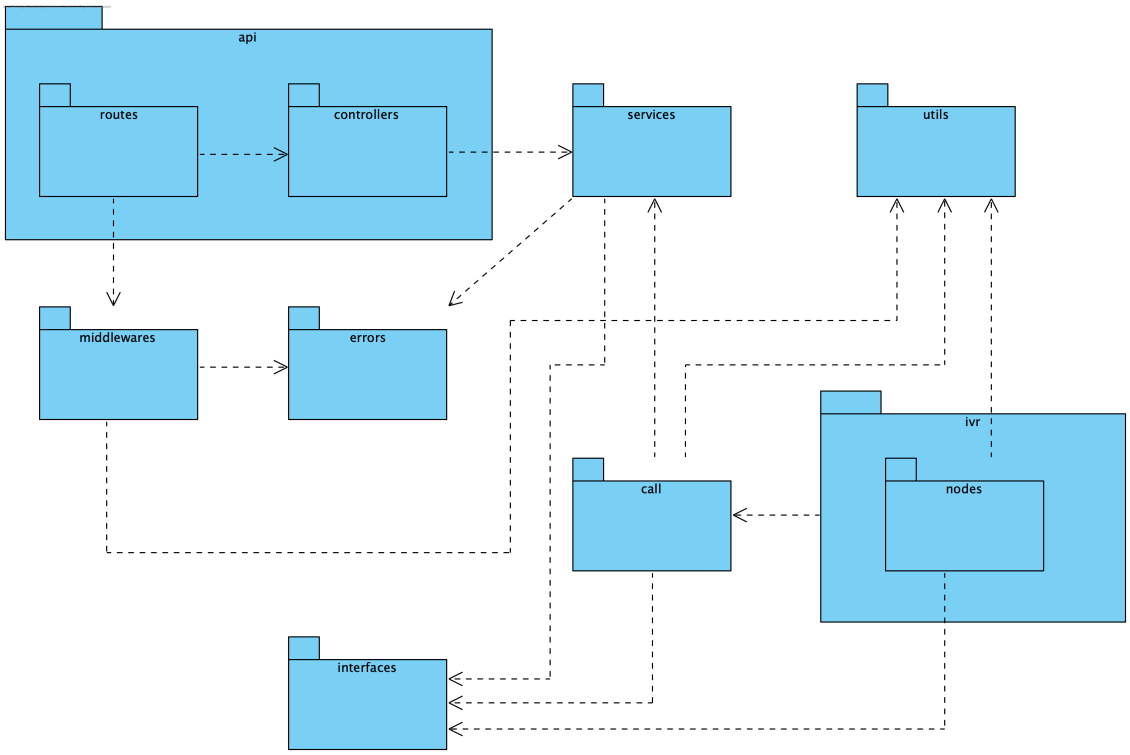

Figura 5.21: Diagrama de *packages* do componente IVR

Speech

No caso do projeto Speech existem mais três *packages*: **text-to-speech**, **speech-to-text** e **rtp-server**.

O *package* **rtp-server** agrega todas as classes responsáveis por receber e transformar os pacotes de áudio, com origem no Asterisk, e que serão usados para reconhecimento de voz.

O *package* **speech-to-text** agrega todas as classes responsáveis por fazer a integração com os vários fornecedores de reconhecimento de voz. No *package* **speech-to-text** está contido o *package* **providers** que agrega todas classes que implementam a integração com os fornecedores externos de reconhecimento de voz (e.g. Google Cloud e Azure Cognitve Services).

O *package* **text-to-speech** agrega todas as classes responsáveis por fazer a integração com os vários fornecedores de síntese de voz. No *package* **text-to-speech** está contido o *package* **providers** que agrega todas classes que implementam a integração com os fornecedores externos de síntese de voz (e.g. Google Cloud e Azure Cognitve Services).

A figura 5.22 representa o diagrama de *packages* para o projeto Speech e as ligações entre os vários *packages*.

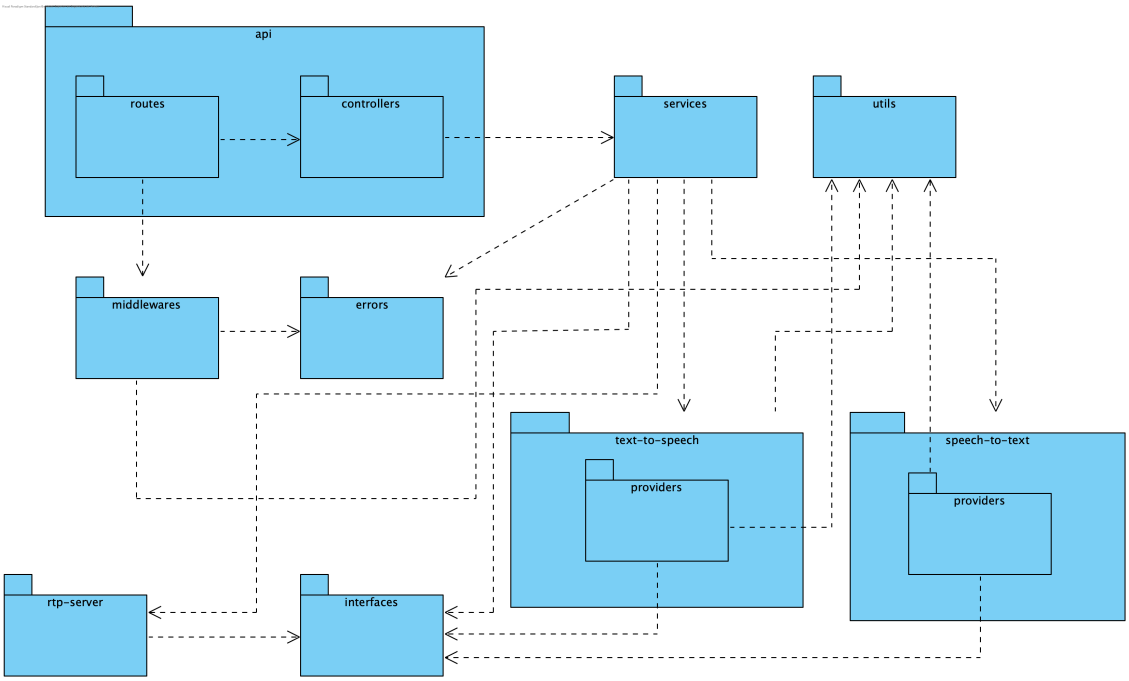

Figura 5.22: Diagrama de *packages* do componente Speech

Settings

No caso do projeto Speech existem mais um *package*: **mappers**.

O *package* **mappers** agrega todas as classes que interagem com os componentes de persistência, neste caso em concreto o componente de persistência é o MySQL.

A figura 5.23 representa o diagrama de *packages* para o projeto Settings e as ligações entre os vários *packages*.

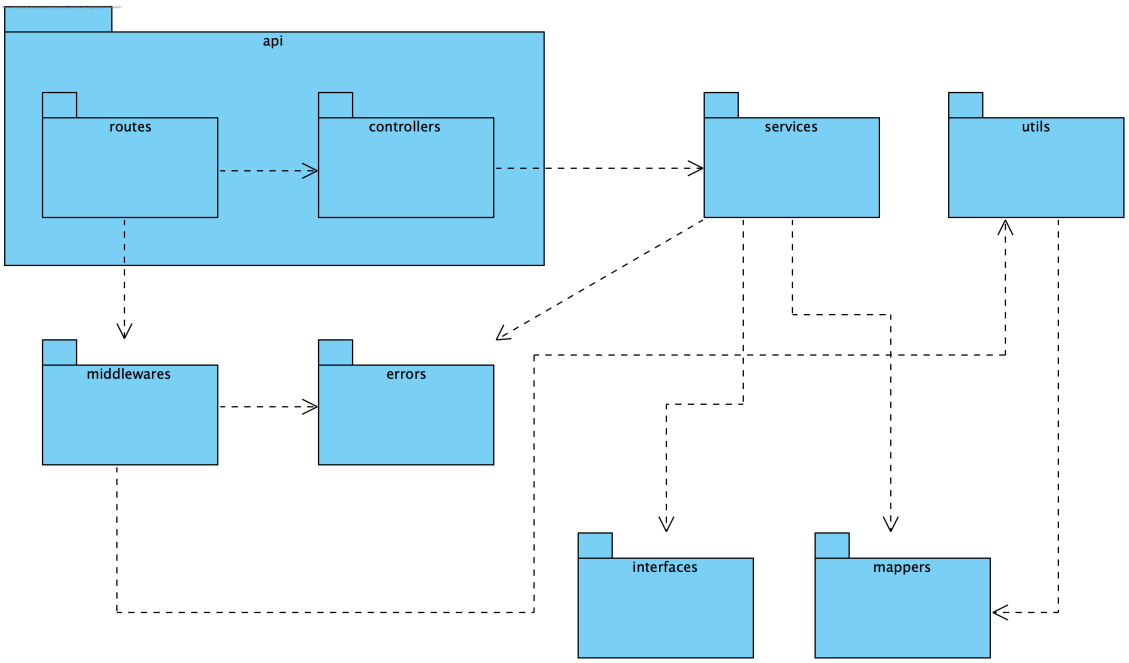

Figura 5.23: Diagrama de *packages* do componente Settings

Capítulo 6

Implementação da solução

Este capítulo tem como objetivo apresentar e explicar a implementação da solução. Além disso, é tido em conta as boas práticas de engenharia de software, sendo apresentados os padrões de software aplicados na solução. Por fim, é explorado o uso de *continuous integration/continuous delivery* e de que forma contribui para reforçar o uso das boas práticas de engenharia de software.

6.1 Motor de IVR

O motor de IVR é o principal componente do sistema de IVR, pois é ele que tem o conhecimento das regras de negócio. Por consequência ele é responsável por processar todos os *inputs* provenientes do visitante e garantir que toda a informação inserida pelo visitante é válida.

O motor de IVR está implementado em TypeScript/Node.js. O motor utiliza uma classe nativa do Node.js (EventEmitter) para propagar informação entre várias classes. Ao usar esta classe para a propagação de informação, está a ser posto em prática o uso do padrão *observer*.

Para impor as regras de negócio, o motor de IVR consulta uma API de definições que retorna as configurações de um IVR e o fluxo que o visitante irá percorrer durante a chamada. A configuração obtida pela API é transformada num DTO, para ser utilizada na construção do fluxo de IVR.

A figura 6.1 apresenta um excerto de código com definição de um DTO.

```
export interface Node {
    nodeType: NodeTypes;
    action: Record<string, any>;
    contextVariables: string[];
    nextNode: string;
}

export interface IVRFlow {
    initialNode: string;
    settings: Record<string, any>;
    nodes: Record<string, Node>;
}
```

Figura 6.1: Excerto de código utilizando o padrão DTO

Este componente recebe todos os inputs do visitante na forma de eventos que podem ter origem no Asterisk (caso o input seja do tipo DTMF) ou no Speech (caso o input seja do tipo reconhecimento de voz).

Com base no processamento dos eventos, o visitante vai percorrendo os vários nós da jornada configurada para a sua chamada.

Existem vários tipos de nós ou tipos de pontos na jornada do visitante durante a chamada:

- Nó de menu
- Nó de reprodução de áudio
- Nó de transferência da chamada
- Nó de pedido externo
- Nó de pergunta

Para construir cada nó do IVR, foi utilizado o padrão *factory* [93], que permite criar objetos sem expor a implementação dos mesmos, pois os objetos implementam uma interface define as funções para cada objeto.

A figura 6.2 apresenta um excerto de código em que o padrão é utilizado.

```
private static buildNodes(
    IVRNodes: Record<string, Node>,
    contextVariables: Record<string, string>,
    // eslint-disable-next-line @typescript-eslint/no-explicit-any
    IVRSettings: Record<string, any>,
    configAPI: ConfigAPI,
    logger: Logger,
    asteriskConfig: AsteriskConfig
) {
    return (nodes: Record<string, NodeInterface>, nodeId: string): Record<string, NodeInterface> => {
        const node: Node = IVRNodes[nodeId];

        if (node.nodeType === NodeTypes.PLAY_AUDIO) {
            nodes[nodeId] = new PlayAudioNode(
                nodeId,
                node,
                contextVariables,
                IVRSettings,
                logger,
                asteriskConfig.soundsPath
            );
            return nodes;
        }

        if (node.nodeType === NodeTypes.MENU) {
            nodes[nodeId] = new MenuNode(
                nodeId,
                node,
                contextVariables,
                IVRSettings,
                logger,
                asteriskConfig.soundsPath
            );
            return nodes;
        }

        if (node.nodeType === NodeTypes.CALL) {
            nodes[nodeId] = new CallNode(nodeId, node, contextVariables, IVRSettings, logger, configAPI);
            return nodes;
        }

        return nodes;
    };
}
```

Figura 6.2: Excerto de código utilizando o padrão *factory*

6.1.1 Nó de menu

O nó de menu permite ao visitante ouvir uma áudio que indica as opções que podem ser escolhidas para progredir no IVR. O áudio reproduzido pode ser lido de um ficheiro ou gerado utilizando *text-to-speech*. Para escolher uma opção o visitante pode introduzir um DTMF ou verbalizar a opção para ser reconhecida via *speech-to-text*.

6.1.2 Nó de reprodução de áudio

O nó de reprodução de áudio tem como objetivo fazer reprodução de um áudio para o visitante, usando um ficheiro ou gerando o áudio através de texto previamente configurado, recorrendo ao uso de *text-to-speech*.

6.1.3 Nó de transferência de chamada

O nó de transferência de chamada tem como objetivo transferir a chamada para uma fila ou para um número externo, isto permite ao visitante comunicar com um agente, caso o IVR não consiga fornecer a informação necessária ao que ele necessita. Neste nó é criado um novo canal para o destino. Assim que atendido o canal de destino é feita a junção dos dois canais numa *bridge*.

6.1.4 Nó de pedido externo

O nó de pedido externo tem como objetivo fazer um pedido a um URL que seja configurado para este nó. Neste momento, apenas é suportado o protocolo HTTP. A informação recolhida a partir da resposta do pedido pode ser utilizada para decidir o próximo caminho ou simplesmente ser armazenada para utilização nos próximos nós do IVR.

6.1.5 Nó de pergunta

O nó de pergunta tem como objetivo permitir ao visitante introduzir informação através do uso de reconhecimento de voz. Na execução deste nó é reproduzido um áudio que é lido de um ficheiro ou gerado por *text-to-speech*. Após a reprodução do áudio, o visitante responde à pergunta verbalmente. A informação recolhida da resposta do visitante pode ser usada para progredir no fluxo ou para ser armazenada para posterior utilização nos restantes caminhos do fluxo de IVR.

6.2 Integração com speech-to-text

A integração com *speech-to-text* é feita no componente Speech. O Speech é responsável por fazer integração com serviços externos de reconhecimento de voz. Os serviços externos utilizados para reconhecimento de voz são:

- Azure Cognitive Services Speech API
- Google Cloud Speech-To-Text API

A integração com os serviços externos foi executada utilizando o padrão *factory*. Com a aplicação deste padrão é possível fazer integrações com mais serviços externos de forma mais fácil. Em conjunto com uma interface comum para as duas implementações de reconhecimento de voz (Azure Cognitive Services Speech API e Google Cloud Speech-To-Text

API), a extensão para mais implementações é fácil e sem grandes alterações ao restante código do componente.

A figura 6.3 apresenta a implementação do padrão *factory* para instanciar objetos de reconhecimento de voz.

```
export default class SpeechToTextFactory {
    public static createSpeechToTextInstance(config: SpeechConfig, logger: Logger): SpeechToText {
        const provider: string = config.provider;

        if (provider === SpeechProviders.AZURE_COGNITIVE_SERVICES) {
            return AzureCognitiveServicesProvider.create(config, logger);
        }

        if (provider === SpeechProviders.GOOGLE_CLOUD) {
            return GoogleCloudProvider.create(config, logger);
        }

        return NullProvider.create(config, logger);
    }
}
```

Figura 6.3: Excerto de código utilizando o padrão *factory*

A figura 6.4 apresenta a definição da interface comum para as implementações de reconhecimento de voz.

```
export interface SpeechToText {
    readonly logger: Logger;
    readonly speechConfig: SpeechConfig;

    pushAudioStream(chunk: Buffer): void;
    finishedAudioStream(): void;
}
```

Figura 6.4: Excerto de código da interface de reconhecimento de voz

Além da utilização do padrão *factory* foi utilizado o padrão *null object*. O padrão *null object* [94] permite substituir a validação por *null* por uma implementação vazia, ou seja existe uma relação de execução vazia. Muitas vezes este padrão é utilizado quando não existem dados e é necessário definir um comportamento por defeito.

A figura 6.5 apresenta a implementação do padrão *null object* para definir um comportamento por defeito caso o reconhecimento de voz não seja suportado.

```
export default class NullProvider extends AbstractSpeechProvider {
    public constructor(readonly config: SpeechConfig, readonly logger: Logger) {
        super();
    }

    public pushAudioStream(chunk: Buffer): void {
        this.logger.error(`${NullProvider.name} :: Push audio stream was called`);
        return;
    }

    public finishedAudioStream(): void {
        this.logger.error(`${NullProvider.name} :: Finish audio stream was called`);
        return;
    }

    public static create(config: SpeechConfig, logger: Logger): SpeechToText {
        return new NullProvider(config, logger);
    }
}
```

Figura 6.5: Excerto de código utilizando o padrão *null object*

A implementação concreta dos serviços externos de reconhecimento de voz recorreu a uma biblioteca desenvolvida pelos serviços externos para a plataforma de Node.js.

As bibliotecas utilizadas foram: @google-cloud/speech e microsoft-cognitiveservices-speech-sdk

Para receber os pacotes de áudio provenientes do Asterisk, o componente Speech está à escuta num intervalo de portas de pacotes UDP no formato aplicacional RTP. Os pacotes RTP recebidos, são transformados num conjunto de pacotes de áudio, que posteriormente são enviados para os serviços de reconhecimento de voz.

Assim que exista um resultado para o áudio enviado para os serviços externos de reconhecimento de voz, é enviado o resultado do reconhecimento para o motor de IVR, que por sua vez, processa o resultado, avançando assim para a próxima etapa no fluxo.

6.3 Integração com text-to-speech

A integração com *text-to-speech* é feita no componente Speech. O Speech é responsável por fazer integração com os serviços externos de síntese de voz. Os serviço externos utilizados para síntese de voz são:

- Azure Cognitive Services Speech API
- Google Cloud Text-To-Speech API

A integração com os serviços externos foi executada utilizando o padrão *factory*. Com a aplicação deste padrão é possível fazer integrações com mais serviços externos de forma mais fácil. Em conjunto com uma interface comum para as duas implementações de síntese de voz (Azure Cognitive Services Speech API e Google Cloud Text-To-Speech API), a extensão para mais implementações é fácil e sem grandes alterações ao restante código do componente.

A figura 6.6 apresenta a implementação do padrão *factory* para instanciar objetos de síntese de voz.

```
export default class TextToSpeechFactory {
    public static createTextToSpeechInstance(config: TextInput, logger: Logger): TextToSpeech {
        const provider: string = config.provider;

        if (provider === TextProviders.AZURE_COGNITIVE_SERVICES) {
            return new AzureCognitiveServicesProvider(config, logger);
        }

        if (provider === TextProviders.GOOGLE_CLOUD) {
            return new GoogleCloudProvider(config, logger);
        }

        return new NullProvider(config, logger);
    }
}
```

Figura 6.6: Excerto de código utilizando o padrão *factory*

A figura 6.7 apresenta a definição da interface comum para as implementações de síntese de voz.

```
export interface TextToSpeech {
    convertTextToAudio(text: string): Promise<TextResult>;
}
```

Figura 6.7: Excerto de código da interface de síntese de voz

Além da utilização do padrão *factory* foi utilizado o padrão *null object*.

A figura 6.5 apresenta a implementação do padrão *null object* para definir um comportamento por defeito caso a síntese de voz não seja suportada.

```
export default class NullProvider implements TextToSpeech {
    public constructor(private readonly config: TextConfig, private readonly logger: Logger) {}

    public async convertTextToAudio(text: string): Promise<TextResult> {
        this.logger.error(`${NullProvider.name} :: config: ${JSON.stringify(this.config)} :: text: ${text}`);
        return {} as TextResult;
    }
}
```

Figura 6.8: Excerto de código utilizando o padrão *null object*

O pedido para efetuar a síntese de voz é proveniente do motor de IVR. O Speech recebe o pedido de síntese de voz com as configurações do serviço externo e o texto que é para ser sintetizado. O áudio obtido dos serviços externos é enviado para o motor de IVR, que reproduz o áudio para o visitante. Toda a comunicação é feita por HTTP.

A implementação concreta dos serviços externos de reconhecimento de voz recorreu a uma biblioteca desenvolvida pelos serviços externos para a plataforma de Node.js.

As bibliotecas utilizadas foram: @google-cloud/text-to-speech e microsoft-cognitiveservices-speech-sdk

6.4 Aplicação de configurações

A aplicação de configurações é um serviço que guarda as configurações de um IVR. As configurações incluem:

- Nós do fluxo
- Configuração de *speech-to-text*
- Configuração de *text-to-speech*

O serviço de configurações disponibiliza uma HTTP API para criar, editar, obter e apagar um fluxo de IVR. O serviço está implementado em TypeScript/Node.js.

A persistência das informações é efectuada recorrendo a uma base de dados relacional.

6.4.1 Modelo da dados

A modelação dos dados teve em conta todas as possibilidades de configuração dos nós assim como as configurações dos serviços externos.

A figura 6.9 apresenta o modelo de dados que permite guardar as configurações dos fluxos de IVR.

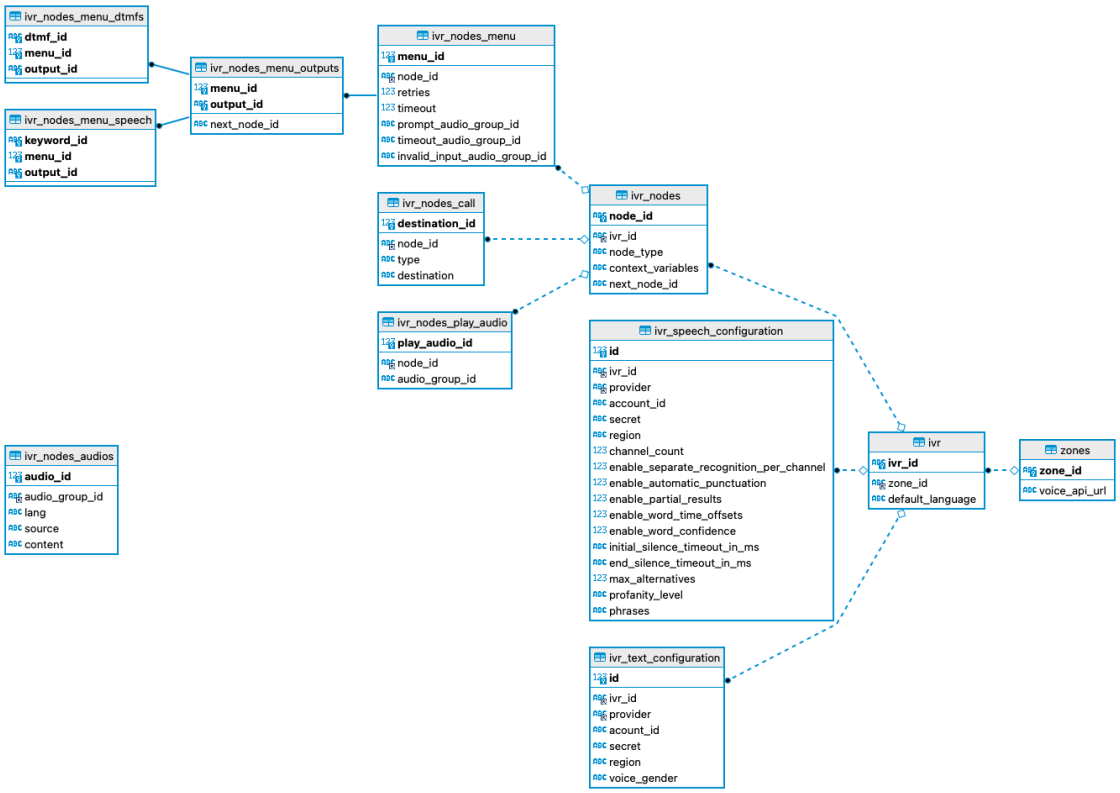

Figura 6.9: Modelo de dados de configurações

6.4.2 Implementação do serviço

O serviço é um simples servidor HTTP, que disponibiliza uma API REST. O serviço implementa o padrão MVC. O padrão MVC estabelece a separação de responsabilidades entre a camada de visualização e a camada de negócio. Com esse intuito é criada uma classe de *pure fabrication* denominada de *controller*.

O conceito de *controller* permite que a camada de visualização mude sem afectar a camada de negócio e vice versa.

Na figura 6.10 é apresentado um exemplo de uma vista.

```
export default (container: IContainer): Router => {
    const router: Router = Router();

    router.delete('/ivr/:id', ivrController.deleteIVR(container));
    router.get('/ivr/:id', ivrController.getIVR(container));
    router.post('/ivr', ivrController.postIVR(container));
    router.put('/ivr/:id', ivrController.putIVR(container));

    return router;
};
```

Figura 6.10: Excerto de código com uma vista do módulo de IVR

Na figura 6.11 é apresentado um exemplo de um *controller*.

```
export const getIVR = (container: IContainer) => async (
    req: Request,
    res: Response,
    next: NextFunction
): Promise<void> => {
    const { id } = req.params;

    const ivrService: IVRService = container.ivrService;

    const ivr: IVR = await ivrService.get(id);

    res.status(200);
    res.json(ivr);
    next();
};
```

Figura 6.11: Excerto de código com o *controller* para o módulo de IVR

Na figura 6.12 é apresentado um exemplo de uma classe da camada de negócio.

```
export default class IVRService {
    public constructor(private readonly ivrModel: IVRModel) {}

    public async get(id: string): Promise<IVR> {
        const ivr: IVR | undefined = await this.ivrModel.get(id);

        if (!ivr) {
            throw new NotFoundError('IVR not found', BysideOneErrorTypes.RESOURCE_NOT_FOUND);
        }

        return ivr;
    }
}
```

Figura 6.12: Excerto de código com a classe de serviço para o módulo de IVR

Para passar a informação entre as várias camadas foi utilizado o padrão DTO. A figura 6.13 apresenta as propriedades do DTO utilizado no módulo IVR.

```
export interface IVR {
    id: string;
    flow: Record<any, any>;
    setings: Record<any, any>;
}
```

Figura 6.13: Excerto de código com o DTO utilizado no módulo IVR

Para além dos padrões referidos em cima, foi utilizado o padrão *dependency injection*. Neste projeto a injeção de dependências é feita de forma simples utilizando um contentor de objetos.

A figura 6.11 representa um exemplo de como é conseguida a injeção de dependências através de um contentor.

O *data mapper* [95] é um padrão utilizado para transferir dados entre os objetos de domínio e os *schemas* presentes nas base de dados relacionais. Com esta abstração, cada entidade envolvida na transação é independente e existe uma separação de responsabilidades bem desenhada.

A figura 6.14 apresenta um excerto de código onde é possível verificar o uso do padrão *data mapper*.

```
export default class BysideOneTeams {
    public constructor(private readonly mySQL: MySQLWrapper) {}

    public async create(team: Team): Promise<Team | undefined> {
        const { teamId, zoneId, host } = team;

        const query: string = `
        INSERT INTO teams SET
            zone_id = :zoneId,
            team_id = :teamId,
            host_id = :hostId`;

        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const params: any = { zoneId, teamId: buildLocalId(zoneId, teamId), hostId: host.hostname };

        const result: OkPacket = await this.mySQL.execute(query, params);

        return result.affectedRows === 1 ? team : undefined;
    }
}
```

Figura 6.14: Excerto de código com um *data mapper*

A API deste serviço foi desenvolvida utilizando a biblioteca **express**. O **express** é uma *framework* para desenvolver aplicações HTTP. Uma das funcionalidades mais utilizadas da *framework* neste projeto foi o encadeamento de *middlewares*, que permitiu obter métricas de sobre o tempo de execução de cada pedido e ajudou na reutilização de código para lidar com erros.

Capítulo 7

Avaliação da solução

Este capítulo tem como objetivo avaliar a solução. Para fazer tal avaliação foram efetuados testes que garantiram a qualidade da solução. Além disso foram aplicadas práticas de CI/CD para tornar os processos de validação automáticos. Por fim, foram aplicadas algumas métricas de monitorização, para avaliar a saúde global do sistema e definidos os limites para alertas e notificações.

7.1 Testes unitários

Como forma de garantir a suportabilidade do sistema, os testes unitários são um mecanismo simples de validação de código. Com os testes unitários conseguimos avaliar que todos os pedaços de código produzem um resultado correto. A métrica de cobertura garante que todo o código foi testado, logo que todos os pedaços de código produzem um resultado correto.

Para executar os testes unitários nos vários serviços que constituem o serviço de IVR, utilizou-se a biblioteca **mocha** como *framework* de teste. Em conjunto com o **mocha**, foi utilizada a biblioteca **nyc**, que permite obter o **relatório de cobertura** dos testes unitário.

O relatório de cobertura gerado pelo **nyc** sobre os testes unitários fornece a cobertura para cada ficheiro constituinte do projeto. A figura 7.1 mostra um exemplo de relatório de cobertura gerado pelo **nyc** no serviço Speech.

```
  130 passing (909ms)

-------------------------------|---------|----------|---------|---------|-------------------
File                           | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------------------------|---------|----------|---------|---------|-------------------
All files                      |     100 |      100 |     100 |     100 |
 api/controllers               |     100 |      100 |     100 |     100 |
  HealthcheckController.ts     |     100 |      100 |     100 |     100 |
  SpeechController.ts          |     100 |      100 |     100 |     100 |
  TextController.ts            |     100 |      100 |     100 |     100 |
 errors                        |     100 |      100 |     100 |     100 |
  APIError.ts                  |     100 |      100 |     100 |     100 |
  APIErrorTypes.ts             |     100 |      100 |     100 |     100 |
  BadRequestError.ts           |     100 |      100 |     100 |     100 |
  InternalServerError.ts       |     100 |      100 |     100 |     100 |
  MethodNotAllowedError.ts     |     100 |      100 |     100 |     100 |
  MySQLError.ts                |     100 |      100 |     100 |     100 |
  NotFoundError.ts             |     100 |      100 |     100 |     100 |
  index.ts                     |     100 |      100 |     100 |     100 |
 middlewares                   |     100 |      100 |     100 |     100 |
  BodyInterceptor.ts           |     100 |      100 |     100 |     100 |
  BodyValidate.ts              |     100 |      100 |     100 |     100 |
  ErrorHandler.ts              |     100 |      100 |     100 |     100 |
  HTTPLogger.ts                |     100 |      100 |     100 |     100 |
  RouteHandler.ts              |     100 |      100 |     100 |     100 |
 rtp-server                    |     100 |      100 |     100 |     100 |
  RTPServer.ts                 |     100 |      100 |     100 |     100 |
 services                      |     100 |      100 |     100 |     100 |
  HealthcheckService.ts        |     100 |      100 |     100 |     100 |
  SpeechService.ts             |     100 |      100 |     100 |     100 |
  TextService.ts               |     100 |      100 |     100 |     100 |
 speech-to-text                |     100 |      100 |     100 |     100 |
  SpeechToText.ts              |       0 |        0 |       0 |       0 |
  SpeechToTextFactory.ts       |     100 |      100 |     100 |     100 |
  SpeechToTextInstances.ts     |     100 |      100 |     100 |     100 |
 text-to-speech                |     100 |      100 |     100 |     100 |
  TextToSpeech.ts              |       0 |        0 |       0 |       0 |
  TextToSpeechFactory.ts       |     100 |      100 |     100 |     100 |
-------------------------------|---------|----------|---------|---------|-------------------
```

Figura 7.1: Relatório de cobertura do serviço Speech

Para automatizar o processo de obtenção da cobertura de teste durante o processo de desenvolvimento, integrou-se a *pipeline* de CI/CD com o relatório de cobertura do **nyc**.

Continuous Integration/Continuous Delivery

Em todos os serviços desenvolvidos foi utilizado a metodologia de CI/CD como forma de automatizar o processo de desenvolvimento dos projetos. A automatização deste processo é obtido através do uso de *pipelines*.

Continuous Integration [96] é o processo automatizado de convergir código para *branches* partilhados entre vários programadores. Como forma de validar as alterações efectuadas ao código são executados teste unitários e de integração.

Continuos Delivery [96] é o processo automatizado de exportação do código para um ambiente de produção. Para ter um processo de CD é necessário que o processo de CI esteja bem estruturado e que disponibilize um executável do código. O executável é a compilação de código que é executado em produção, logo este artefacto gerado durante a *pipeline* é exportado para produção.

No desenvolvimento dos serviços integrantes deste projeto, é utilizado o Gitlab como repositório de código e gestor de versões. O Gitlab além das funcionalidades de que permitem gerir código, disponibiliza na sua plataforma a possibilidade de criar um processo de CI/CD.

Nos projetos (IVR, Speech e Settings) foram criados os processos de CI/CD. Nas figura 7.2 é apresentado um exemplo de uma *pipeline* de CI/CD.

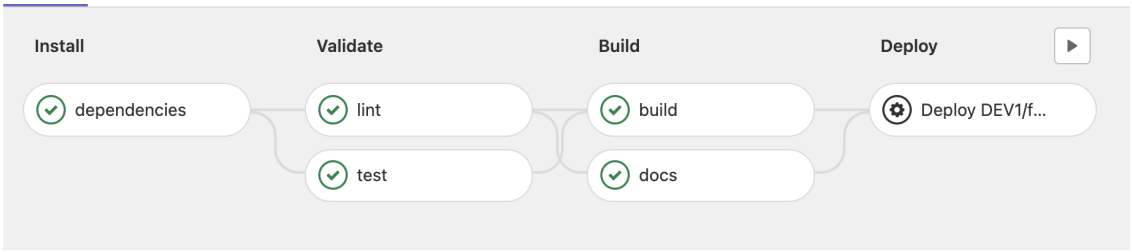

Figura 7.2: *Pipeline* do serviço motor IVR

Recolha dos resultados

Com a automatização do processo de obtenção do relatório de cobertura é possível analisar a variação do cobertura do projeto ao longo do desenvolvimento. A cobertura é analisada *commit* a *commit*, logo se um *commit* tiver uma descida na cobertura significa que: o *commit* não possui teste unitários ou o *commit* não tem casos de teste insuficientes para cobrir todas as condições do código.

Isso pode ser observado na *pipeline*, a figura 7.3 apresenta um exemplo de como se pode obter o relatório de cobertura usando uma *pipeline* de CI/CD.

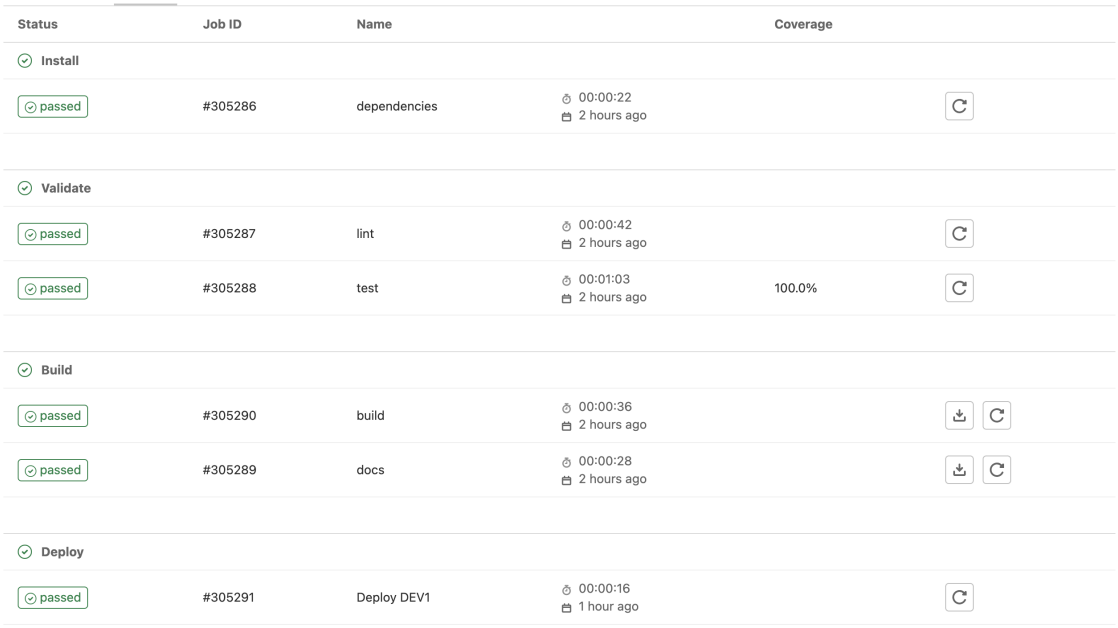

Figura 7.3: Tarefas da *pipeline* do serviço de Settings

7.1.1 Resultados

Como já referido anteriormente, o resultado dos testes unitários podem ser obtidos através do uso das *pipelines* de CI/CD, construídas para os serviços desenvolvidos neste projeto. A seguir são mostrados os resultados obtidos de cobertura obtido em cada serviço.

IVR

Usando a *pipeline* de CI/CD para o serviço IVR, observa-se na figura 7.4 que o mesmo possui uma cobertura de 100% de testes unitários.

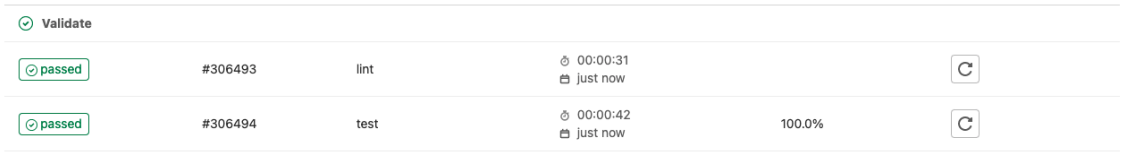

Figura 7.4: Cobertura de testes unitário no serviço IVR

Speech

Usando a *pipeline* de CI/CD para o serviço Speech, observa-se na figura 7.5 que o mesmo possui uma cobertura de 100% de testes unitários.

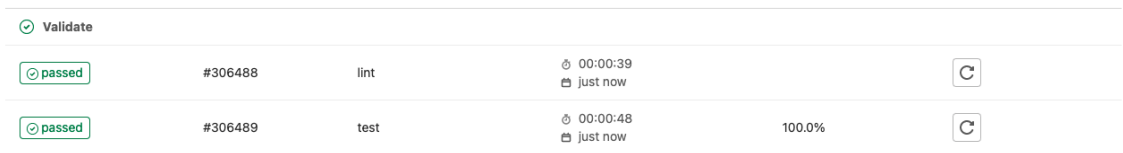

Figura 7.5: Cobertura de testes unitário no serviço Speech

Settings

Usando a *pipeline* de CI/CD para o serviço Settings, observa-se na figura 7.5 que o mesmo possui uma cobertura de 100% de testes unitários.

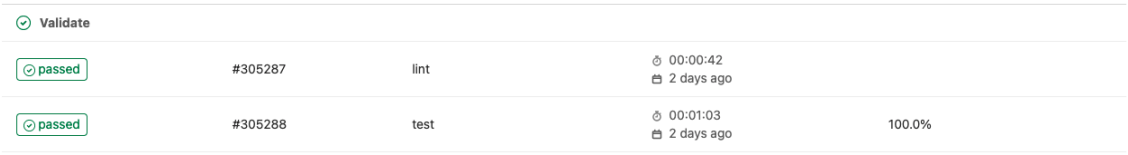

Figura 7.6: Cobertura de testes unitário no serviço Settings

Tabela com agregação dos dados

Serviço	Cobertura de testes unitários (%)
IVR	100
Speech	100
Settings	100

Tabela 7.1: Tabela com os resultados de cobertura de testes unitários para os serviços do sistema de IVR

7.2 Integração do motor de IVR com o sistema de monitorização BySide

Para o novo sistema ser integrado com a monitorização da plataforma BySide foram definidos alguns indicadores para avaliar a saúde da solução. Os indicadores definidos foram:

- Utilização de CPU;
- Memória disponível;
- Se o motor de IVR está ou não responsivo.

Utilização de CPU

Nesta métrica é analisada a percentagem de CPU utilizada e a carga média do CPU durante a execução de todos os serviços integrantes do motor de IVR.Caso este atinja um limite definido como crítico, isto pode ser um indício que o sistema precisa de ser escalado horizontalmente para lidar com o tráfego.

Memória disponível

Nesta métrica é analisado os valores de memória disponível, com esta métrica é possível verificar o nível de consumo de memória do motor de IVR. Caso este atinja um limite definido como crítico, isto pode ser um indício que o sistema precisa de ser escalado horizontalmente para lidar com o tráfego.

Se o motor de IVR está ou não responsivo

Nesta métrica é analisado se o motor de IVR é responsivo, isto é, se o motor está ativo e consegue processar pedidos. Caso o motor não responda ao pedido de *keep alive* pode indicar uma diminuição na performance do motor de IVR ou que existe um problema de rede.

7.2.1 Softwares de monitorização utilizados na BySide

Para monitorizar a sua plataforma e os vários componentes constituintes da mesma, a BySide utiliza dois softwares de monitorização: o Atlas e o Zabbix.

Zabbix

O Zabbix [97] é um software de monitorização *open source* para servidores, máquinas virtuais e dispositivos de rede. O Zabbix recolhe métricas em tempo real sobre o estado dos diferentes dispositivos monitorizados. Além disso é possível criar alertas, visualizar as métricas em painéis e criar períodos de manutenção.

A seguir são apresentadas algumas métricas [98] recolhidas pelo Zabbix:

- Percentagem de uso de CPU;
- Memória livre/ocupada;
- Pacotes/bytes transferidos pelas interfaces de rede;
- Espaço em disco.

Atlas

O Atlas é um projeto desenvolvido internamente e tem como objetivo monitorizar os serviços aplicacionais. A plataforma BySide tem várias zonas geográficas, cada zona geográfica possui serviços aplicacionais independentes. Os serviços aplicacionais são monitorizados usando a técnica de *keep alive*, ou seja, o Atlas inicia um pedido e o serviço aplicacional dá uma resposta.

O Atlas possui um painel que permite analisar os serviços que se encontram ativos. A figura 7.7 apresenta o painel de monitorização do Atlas.

Figura 7.7: Painel do Atlas com a monitorização dos serviços aplicacionais

Integração do motor de IVR com o Zabbix

No contexto deste projeto o Zabbix é utilizado para monitorizar as várias VMs onde o motor de IVR está alojado. O Zabbix está a recolher as métricas de utilização de CPU assim como as métricas sobre a memória disponível.

A recolha das métricas é feita por um processo do Zabbix, denominado de Zabbix Agent. O Zabbix Agent [98] é um processo, disponível para várias plataformas e sistemas operativos, que é responsável por enviar/obter as várias métricas suportadas pelo Zabbix.

A figura 7.8 apresenta um exemplo de um gráfico onde é possível observar a recolha das métricas de uma das VMs onde o motor de IVR está alojado.

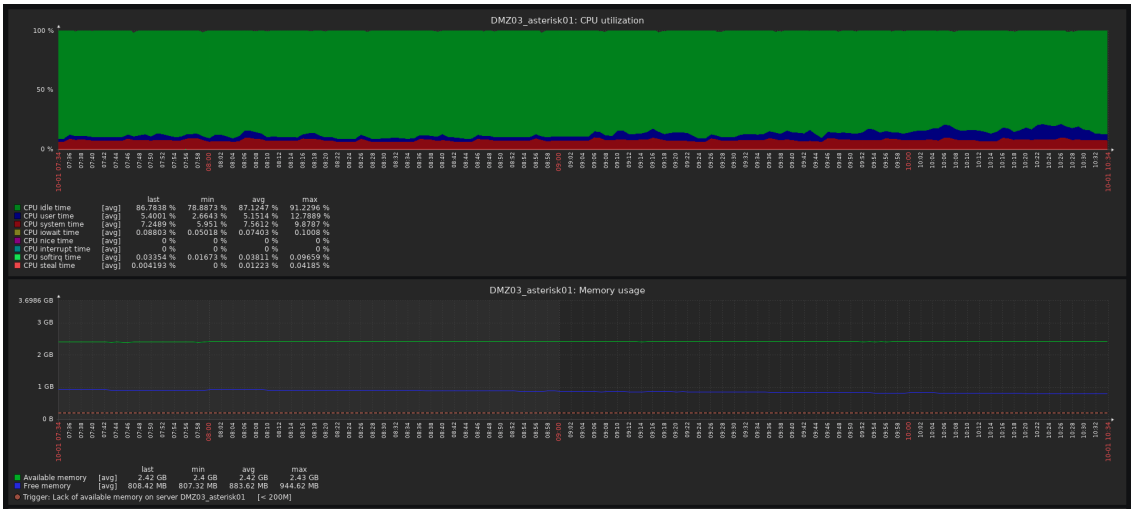

Figura 7.8: Painel do Zabbix com a monitorização de uma VM onde o motor está alojado

Com base nas métricas recolhidas foi possível criar alguns alertas. Para a utilização do CPU, quando este atingi um valor de utilização acima dos 90%, é disparada uma notificação para um canal de *chat* dedicado para este tipo de ocorrências. Para o caso de memória disponível, quando esta atinge o valor de menos 200 MB, é disparada uma notificação, assim como é feito para a utilização do CPU.

Integração do motor IVR com o Atlas

Para o Atlas conseguir fazer um *keep alive* ao motor de IVR, os vários serviços que o constituem disponibilizam uma rota HTTP destinada a esse propósito. Se o Atlas receber resposta ao pedido enviado considera que o serviço está disponível e que não existe nenhum problema com esse mesmo serviço.

Caso o serviço não responda ao pedido dentro de um determinado intervalo de tempo, isto pode indicar que o serviço está degradado. Com base nesta informação, pode-se tirar várias conclusões:

A primeira é que o sistema não está a conseguir lidar com o nível de tráfego presente naquele momento, e então é necessário escalar o sistema horizontalmente para de forma suave lidar com esse pico.

Outra alternativa possível é a ocorrência de um problema de rede entre os serviços constituintes do motor de IVR e o Atlas que pode provocar inconsistência dos dados do painel, no momento em que a interrupção na ligação entre os dois sistemas está a decorrer.

Capítulo 8

Conclusão

Este capítulo tem como objetivo fornecer uma apreciação global sobre o trabalho desenvolvido ao logo do projeto. Primeiramente é feita uma reflexão sobre os objetivos alcançados e os objetivos que ficaram por cumprir. De seguida é apresentado as limitações do projeto e os desenvolvimentos futuros para cumprir os objetivos que ficaram por alcançar. Por último é feita uma avaliação sobre os resultados obtidos e de que forma eles contribuíram para o percurso pessoal e profissional.

8.1 Objetivos alcançados

Nesta secção são apresentados os objetivos alcançados durante o desenvolvimento do projeto, quer a nível pessoal, quer a nível académico. Com base nos objetivos estabelecidos no capítulo 1, fez-se uma reflexão individual sobre cada um.

8.1.1 Nível pessoal

Os objetivos a nível pessoal foram enumerados no capítulo 1. Nesta sub-secção é feita uma reflexão sobre a forma como os objetivos foram concretizados e uma apreciação crítica sobre os seus resultados no crescimento pessoal.

Integração com tecnologias de reconhecimento e síntese de voz

A investigação sobre tecnologias de reconhecimento e síntese de voz permitiu entender os algoritmos e de que forma estas tecnologias foram evoluindo ao longo do tempo. Além da evolução tecnológica, é interessante constatar que todos os modelos teóricos sobre estas tecnologias foram desenvolvidos durante grande parte do século XX e que ainda hoje estão na base do reconhecimento e síntese de voz. Contudo estas tecnologias não eram viáveis nesse momento, pois os restantes componentes tecnológicos não tinham capacidade para suportar estas tecnologias.

No contexto do sistema desenvolvido a integração destas tecnologias tendo em conta a arquitetura desenhada pode ser considerado como cumprido, pois o motor de IVR consegue utilizar qualquer tipo de tecnologia de forma uniforme independentemente do serviço externo utilizado.

Investigação sobre workflows

A investigação sobre os *workflows* permitiu perceber que um IVR é um *workflow* que tanto é automático como manual. Com esta visão, o sistema de IVR teve em conta que o IVR é um fluxo. Um fluxo é um conjunto de nós que possuem ligação entre si.

Com base nisto, foram criadas abstrações que traduzem nós de um fluxo (e.g. Nó de menu, Nó de pergunta). Cada nó possui uma ligação, ou seja, um nó aponta sempre para outro nó, criando uma sequência de ações.

Existem nós que desencadeiam processos manuais (e.g. Nó de menu, Nó de pergunta) e outros que são puramente automáticos (e.g. Nó de reprodução de áudio, Nó de pedido externo).

Este objetivo foi cumprido, pois contribui para o desenho da solução, na medida em que as configurações de um IVR são baseadas em sequência de tarefas, automáticas ou manuais.

Estabelecer uma configuração uniforme para um IVR

A configuração foi atingida através da criação de abstrações (e.g. Nó de menu, Nó de pergunta). Com a criação destas abstrações conseguiu-se modelar a configurações necessárias para a criação de um fluxo de IVR. As abstrações dão espaço para a criação de uma interface gráfica intuitiva e robusta, pois cada parâmetro da configuração consegue ser validado mesmo antes de ser gravado.

8.1.2 Nível académico

A nível académico, esta dissertação contribuiu para por em prática conhecimentos adquiridos ao longo do mestrado. As unidades curriculares que contribuíram para a implementação deste projeto foram organização e desenvolvimento de software, qualidade em engenharia de software e integração de sistemas.

Em organização e desenvolvimento de software, foram leccionados os conceitos fundamentais de CI/CD. Além disso, foram exploradas algumas ferramentas que possibilitam a análise de código (e.g. *linters*, ferramentas de análise estática de código), e como é possível integrar estas tecnologias numa pipeline de CI/CD.

Em qualidade em engenharia de software, foram leccionadas metodologias de teste para assegurar a qualidade do software. Entre as metodologias leccionadas encontra-se os teste unitários, que foram diretamente aplicados neste projeto.

Em integração de sistemas, foram leccionadas técnicas de integração entre vários sistemas de informação. Além disso, foi leccionado os principais conceitos de uma arquitetura de micro-serviços. Neste projeto alguns dos conceitos dos micro-serviços foram aplicados (e.g. segregação de responsabilidades por cada componente).

Conforme apresentado nos vários tópicos apresentados em cima, conclui-se que a experiência académica teve um grande impacto na realização desta dissertação, sendo um guia excelente para aplicação das boas práticas de engenharia.

8.2 Limitações e trabalho futuro

Nesta secção são apresentadas as limitações do projeto assim como o plano de trabalho futuro a ser desenvolvido. Nas limitações é mostrado todos os tópicos que o motor de IVR não contempla e que podem ser implementados ou suportados. Na parte de trabalho futuro é apresentado um plano para proceder à implementação das funcionalidades que identificadas nas limitações.

8.2.1 Limitações

Neste momento as limitações presentes no motor de IVR são:

- A falta de registo das interações do visitante com o IVR;
- A falta de estatísticas sobre cada nó (e.g. tempo em processamento no nó, número de tentativas falhadas);
- Falta de suporte para fluxos por contexto, ou seja, baseado no processamento de linguagem natural.

8.2.2 Trabalho futuro

O planeamento do trabalho futuro foi priorizado tendo em conta as necessidades que a equipa de produto considera serem chaves para que o sistema possa ser lançado comercialmente. Posto isto o trabalho futuro foi dividido em duas fases: curto prazo (próximos 3 meses) e médio/longo prazo (após conclusão do plano a curto prazo).

A curto prazo

A curto prazo os objetivos são:

- Registar as interações do visitante com o IVR;
- Fornecer estatísticas sobre cada nó de um IVR.

Registar as interações do visitante com o IVR

Para registar as interações do visitante com o IVR é necessário adicionar aos nós, a publicação de eventos para uma fila com os dados introduzidos pelo visitante. O eventos são, por sua vez, consumidos por um serviço, que persiste os dados na base dados. Os dados podem ser consultados recorrendo a uma rota HTTP definida na API do sistema de IVR.

Fornecer estatísticas sobre cada nó de um IVR

Para fornecer as estatísticas sobre cada nó de um IVR a implementação dos nós deve ser alterada para guarda a informação dentro do contexto do IVR. No final do IVR, os dados guardados são publicados para uma fila. Os eventos são consumidos por um serviço, que persiste a informação na base de dados. Os dados podem ser consultados através do uso de uma rota HTTP definida na API do sistema de IVR.

A médio/longo prazo

A médio e a longo prazo o objetivo é integrar serviços de interpretação de linguagem natural (e.g. Microsoft LUIS, Google Dialogflow), para que seja possível ter suporte a fluxos por contexto.

Os fluxos por contexto são fluxos que com base no sentido frásico do texto reconhecido, conseguem escolher de forma inteligente, o caminho especializado para responder de forma assertiva ao problema do visitante.

Para suportar este tipo de fluxo é necessário adicionar uma nova implementação ao motor de IVR. Em alguns casos, pode ser necessário refazer algumas implementações existentes.

O objetivo para a integração passa por analisar a documentação fornecida pelos vários serviços de processamento de linguagem natural ou por analisar software *open source*, em que seja possível hospedar o software nos servidores da BySide.

O próximo passo é desenvolver uma API genérica para fazer a comunicação com os vários serviços de processamento de linguagem natural.

Finalmente fazer a integração da API genérica de NLP com a nova implementação dos fluxos por contexto.

8.3 Apreciação final

A presente dissertação foi um grande desafio quer a nível pessoal, quer a nível profissional e académico. O contacto com novas tecnologias abriu os horizontes para novos desafios e novas ideias, para o trabalho futuro no novo sistema de IVR da BySide.

O trabalho desenvolvido durante a dissertação contribui para fazer a ponte entre o mundo académico e o mundo empresarial. Alguns dos conceitos aprendidos em contexto académico contribuiram de forma positiva para o mundo empresarial, assim como alguns dos conhecimentos do mundo empresarial contribuíram para a realização e produção desta dissertação.

Em nota final, o projeto foi desafiante e inspirador. Agora é hora de colher os frutos e construir iterativamente uma solução cada vez melhor.

Bibliografia

[1] *BySide Lead Activation Journey*. url: https://www.byside.com/lead-activation.

[2] *What is Interactive Voice Response (IVR)? Definition and Benefits*. url: https://www.ttec.com/glossary/interactive-voice-response.

[3] *What is IVR and 6 Benefits of Using One*. Fev. de 2021. url: https://www.talkdesk.com/blog/what-is-an-ivr-and-6-benefits-of-using-one/.

[4] *Interactive voice response*. Jul. de 2021. url: https://en.wikipedia.org/wiki/Interactive_voice_response.

[5] Kissflow. *What is a Workflow?: Definition & Guide of Workflows*. Set. de 2021. url: https://kissflow.com/workflow/what-is-a-workflow/#how-can-spot-workflows.

[6] Clifford Chi. *Workflow Automation Explained & 6 Best Workflow Software for 2021*. Ago. de 2021. url: https://blog.hubspot.com/marketing/workflow-automation.

[7] *Voice Over Internet Protocol (VoIP)*. Nov. de 2015. url: https://www.fcc.gov/general/voice-over-internet-protocol-voip.

[8] *Voice over IP*. Set. de 2021. url: https://en.wikipedia.org/wiki/Voice_over_IP#Protocols.

[9] *Real-time Transport Protocol*. Abr. de 2021. url: https://en.wikipedia.org/wiki/Real-time_Transport_Protocol.

[10] url: https://ptolemy.berkeley.edu/eecs20/week2/dtmf.html.

[11] Dave Horton. *Contributors*. url: https://drachtio.org/about.

[12] Sangoma Technologies. url: https://wiki.asterisk.org/wiki/display/AST/Asterisk%20as%20a%20Swiss%20Army%20Knife%20of%20Telephony.

[13] Sangoma Technologies. url: https://wiki.asterisk.org/wiki/display/AST/Channel%20Drivers.

[14] Sangoma Technologies. url: https://wiki.asterisk.org/wiki/display/AST/Dialplan.

[15] VoIP Info. *Asterisk AGI: Asterisk Gateway Interface - VoIP-Info*. Set. de 2021. url: https://www.voip-info.org/asterisk-agi/.

[16] Sangoma Technologies. url: https://wiki.asterisk.org/wiki/display/AST/The%20Asterisk%20Manager%20TCP%20IP%20API.

[17] Sangoma Technologies. url: https://wiki.asterisk.org/wiki/display/AST/AMI%20Libraries%20and%20Frameworks.

[18] Sangoma Technologies. url: https://wiki.asterisk.org/wiki/pages/viewpage.action?pageId=29395573.

[19] Sangoma Technologies. url: https://wiki.asterisk.org/wiki/display/AST/ARI%20Libraries.

[20] B. H. Juang e Lawrence R. Rabiner. *Automatic Speech Recognition – A Brief History of the Technology Development Abstract*. 2004.

[21] *Speech recognition*. Set. de 2021. url: https://en.wikipedia.org/wiki/Speech_recognition.

[22] Google. *Speech-to-Text: Automatic Speech Recognition | Google Cloud*. url: https://cloud.google.com/speech-to-text.
[23] Google. *Quotas & limits | Cloud Speech-to-Text Documentation*. url: https://cloud.google.com/speech-to-text/quotas.
[24] Google. *Detecting language spoken automatically*. url: https://cloud.google.com/speech-to-text/docs/multiple-languages.
[25] Google. *Enabling the profanity filter | Cloud Speech-to-Text Documentation*. url: https://cloud.google.com/speech-to-text/docs/profanity-filter.
[26] Google. *Getting automatic punctuation | Cloud Speech-to-Text Documentation*. url: https://cloud.google.com/speech-to-text/docs/automatic-punctuation.
[27] Google. *Send a recognition request with model adaptation*. url: https://cloud.google.com/speech-to-text/docs/adaptation.
[28] Google. *Separating different speakers in an audio recording*. url: https://cloud.google.com/speech-to-text/docs/multiple-voices.
[29] Google. *Cloud Speech-to-Text API | Cloud Speech-to-Text Documentation*. url: https://cloud.google.com/speech-to-text/docs/reference/rest.
[30] Google. *Quickstart: Using client libraries*. url: https://cloud.google.com/speech-to-text/docs/quickstart-client-libraries.
[31] Microsoft. *Speech to Text – Audio to Text Translation: Microsoft Azure*. url: https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text.
[32] Microsoft. *Speech-to-text quickstart - Speech service - Azure Cognitive Services*. url: https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/get-started-speech-to-text?tabs=windowsinstall&pivots=programming-language-nodejs.
[33] Microsoft. *How to use language identification - Azure Cognitive Services*. url: https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/how-to-automatic-language-detection?pivots=programming-language-csharp.
[34] Microsoft. *PropertyId enum*. url: https://docs.microsoft.com/en-us/javascript/api/microsoft-cognitiveservices-speech-sdk/propertyid?view=azure-node-latest.
[35] Microsoft. *Custom Speech overview - Speech service - Azure Cognitive Services*. url: https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/custom-speech-overview.
[36] IBM. *Watson Speech to Text - Features*. url: https://www.ibm.com/cloud/watson-speech-to-text/features.
[37] IBM. *IBM Cloud Docs*. url: https://cloud.ibm.com/docs/speech-to-text?topic=speech-to-text-service-features (acedido em 20/09/2021).
[38] IBM. *IBM Cloud Docs*. url: https://cloud.ibm.com/docs/speech-to-text?topic=speech-to-text-languageCreate (acedido em 20/09/2021).
[39] IBM. *IBM Cloud Docs*. url: https://cloud.ibm.com/docs/speech-to-text?topic=speech-to-text-acoustic (acedido em 20/09/2021).
[40] Amazon. *Amazon Transcribe Features – Amazon Web Services*. en-US. url: https://aws.amazon.com/transcribe/features/ (acedido em 20/09/2021).
[41] Amazon. *Speech input - Amazon Transcribe*. url: https://docs.aws.amazon.com/transcribe/latest/dg/input.html (acedido em 20/09/2021).

[42] Amazon. *What is Amazon Transcribe? - Amazon Transcribe*. url: https://docs.aws.amazon.com/transcribe/latest/dg/transcribe-whatis.html (acedido em 20/09/2021).

[43] Amazon. *Redacting or identifying personally identifiable information - Amazon Transcribe*. url: https://docs.aws.amazon.com/transcribe/latest/dg/pii-redaction.html (acedido em 20/09/2021).

[44] Amazon. *Transcribing streaming audio - Amazon Transcribe*. url: https://docs.aws.amazon.com/transcribe/latest/dg/streaming.html (acedido em 20/09/2021).

[45] Karolina Kuligowska, Pawel Kisielewicz e Aleksandra Włodarz. «Speech synthesis systems: disadvantages and limitations». Em: *International journal of engineering and technology* 7 (2018), p. 234.

[46] Sciforce. *Text-to-Speech Synthesis: an Overview*. Fev. de 2020. url: https://medium.com/sciforce/text-to-speech-synthesis-an-overview-641c18fcd35f.

[47] *Speech synthesis*. Ago. de 2021. url: https://en.wikipedia.org/wiki/Speech_synthesis#Applications.

[48] Google. *Text-to-Speech: Lifelike Speech Synthesis | Google Cloud*. url: https://cloud.google.com/text-to-speech.

[49] Google. *Supported voices and languages | Cloud Text-to-Speech Documentation*. url: https://cloud.google.com/text-to-speech/docs/voices.

[50] Google. *Method: text.synthesize | Cloud Text-to-Speech Documentation*. url: https://cloud.google.com/text-to-speech/docs/reference/rest/v1/text/synthesize.

[51] Google. *Text-to-Speech client libraries | Cloud Text-to-Speech Documentation*. url: https://cloud.google.com/text-to-speech/docs/libraries.

[52] Microsoft. *Text to Speech*. url: https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech/#overview.

[53] Microsoft. *Text-to-speech quickstart - Speech service - Azure Cognitive Services*. url: https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/get-started-text-to-speech?tabs=script,windowsinstall&pivots=programming-language-javascript.

[54] IBM. *IBM Cloud Docs*. url: https://cloud.ibm.com/docs/text-to-speech?topic=text-to-speech-service-features (acedido em 20/09/2021).

[55] IBM. *IBM Cloud Docs*. url: https://cloud.ibm.com/docs/text-to-speech?topic=text-to-speech-voices (acedido em 20/09/2021).

[56] IBM. *IBM Cloud Docs*. url: https://cloud.ibm.com/docs/text-to-speech?topic=text-to-speech-ssml (acedido em 20/09/2021).

[57] Amazon. *Amazon Polly Features*. en-US. url: https://aws.amazon.com/polly/features/ (acedido em 20/09/2021).

[58] Amazon. *awsdocs/aws-doc-sdk-examples: Welcome to the AWS Code Examples Repository. This repo contains code examples used in the AWS documentation, AWS SDK Developer Guides, and more. For more information, see the Readme.rst file below.* url: https://github.com/awsdocs/aws-doc-sdk-examples.

[59] Genesys. *Contact Center Solutions: Omnichannel Customer Experience*. url: https://www.genesys.com/.

[60] Genesys. *Interactive voice response (IVR): Call centre software*. url: https://www.genesys.com/en-gb/capabilities/interactive-voice-response-ivr.

[61] Genesys. *Personalised Self-Service Customer Experience Solutions*. url: https://www.genesys.com/en-gb/capabilities/customer-self-service.

[62] Genesys. *Genesys Secure IVR Payments*. url: https://www.genesys.com/en-gb/collateral/genesys-secure-ivr-payments.

[63] Zendesk. *Customer Service Software & Sales CRM: Best in 2021*. url: https://www.zendesk.com/.

[64] Nora Mullen Edited September 14. *Routing incoming calls with IVR*. url: https://support.zendesk.com/hc/en-us/articles/214630317-Routing-incoming-calls-with-IVR.

[65] Talkdesk. *About | Talkdesk*. url: https://www.talkdesk.com/about/.

[66] Talkdesk. *Talkdesk Studio - Cloud IVR Software*. url: https://www.talkdesk.com/cloud-contact-center/customer-engagement/interactive-voice-response-ivr-system/.

[67] E-goi. *goi - Email Marketing e Marketing Automation para todos!* Set. de 2021. url: https://www.e-goi.com/pt/.

[68] E-goi. *Campanhas de Voz: Envie Mensagens e Chamadas de Voz Interativas*. Jul. de 2020. url: https://www.e-goi.com/pt/campanhas-de-voz/.

[69] Twilio. *What is Cloud Communications*. url: https://www.twilio.com/what-is-cloud-communications.

[70] Twilio. *How To Build an IVR System with Twilio Studio*. Jan. de 2018. url: https://www.youtube.com/watch?v=GifYSpB-EU4.

[71] Twilio. *IVR*. url: https://www.twilio.com/solutions/ivr.

[72] Nick Rich e Matthias Holweg. Em: *Value Analysis, Value Engineering* (jan. de 2000). url: https://www.urenio.org/tools/en/value_analysis.pdf.

[73] Peter A. Koen et al. «1 Fuzzy Front End : Effective Methods , Tools , and Techniques». Em: 2002.

[74] Twilio. *The Seven Benefits of an IVR System - Voice & Video*. url: https://www.twilio.com/learn/voice-and-video/the-seven-benefits-of-an-ivr-system.

[75] Thomas L. Saaty. «How to make a decision: The analytic hierarchy process». Em: *European Journal of Operational Research* 48 (1990), pp. 9–26.

[76] Enrique Mu e Milagros Pereyra-Rojas. «Understanding the Analytic Hierarchy Process». Em: *Practical Decision Making: An Introduction to the Analytic Hierarchy Process (AHP) Using Super Decisions V2*. Cham: Springer International Publishing, 2017, pp. 7–22. isbn: 978-3-319-33861-3. doi: 10.1007/978-3-319-33861-3_2. url: https://doi.org/10.1007/978-3-319-33861-3_2.

[77] Susana Nicola, Eduarda Pinto Ferreira e João Pinto Ferreira. «A novel framework for modeling value for the customer, an essay on negotiation». Em: (2012). doi: https://www.worldscientific.com/doi/epdf/10.1142/S0219622012500162.

[78] Philip Kotler e Kenvin Lane Keller. *Creating Customer Value, Satisfaction, and Loyalty*. Pearson Education, Inc, 2009.

[79] Tony Woodall. «Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis». Em: *Academy of Marketing Science Review* 12 (jan. de 2003).

[80] Cambridge Dictionary. *Value Proposition*. url: https://dictionary.cambridge.org/dictionary/english/value-proposition.

[81] Ana Barquet et al. «Business Model Elements for Product-Service System». Em: mar. de 2011, pp. 332–337. isbn: 978-3-642-19688-1. doi: 10.1007/978-3-642-19689-8_58.

[82] Alexander Osterwalder e Yves Pigneur. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Jan. de 2010, 1 online resource (278 s.)

[83] Mike Ebinum. *How To: Business Model Canvas Explained*. Out. de 2019. url: https://medium.com/seed-digital/how-to-business-model-canvas-explained-ad3676b6fe4a.

[84] F Silvestre. *TCN-SRS-01_00-EN*. Fev. de 2003. url: https://ec.europa.eu/transport/sites/default/files/modes/road/social_provisions/doc/tcn_srs_01-00.pdf.

[85] Tim Ottinger e Jeff Langr. url: https://agileinaflash.blogspot.com/2009/04/furps.html.

[86] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004.

[87] url: https://www.tutorialspoint.com/software_architecture_design/introduction.htm.

[88] Martin Fowler. *Software Architecture Guide*. url: https://martinfowler.com/architecture/.

[89] Hans van Vliet e Antony Tang. «Decision making in software architecture». Em: *Journal of Systems and Software* 117 (2016), pp. 638–644. issn: 0164-1212. doi: https://doi.org/10.1016/j.jss.2016.01.017. url: https://www.sciencedirect.com/science/article/pii/S0164121216000157.

[90] Xing Wang. *What is HTTP middleware? Best practices for building, designing and using middleware.* Ago. de 2018. url: https://www.moesif.com/blog/engineering/middleware/What-Is-HTTP-Middleware/.

[91] Martin Fowler. *Service Layer*. url: https://martinfowler.com/eaaCatalog/serviceLayer.html.

[92] Martin Fowler. *Data Transfer Object*. url: https://martinfowler.com/eaaCatalog/dataTransferObject.html.

[93] url: https://www.tutorialspoint.com/design_pattern/factory_pattern.htm.

[94] url: https://www.tutorialspoint.com/design_pattern/null_object_pattern.htm.

[95] Martin Fowler. *Data Mapper*. url: https://martinfowler.com/eaaCatalog/dataMapper.html.

[96] *What is CI/CD?* url: https://www.redhat.com/en/topics/devops/what-is-ci-cd.

[97] Zabbix. *What is Zabbix*. url: https://www.zabbix.com/features.

[98] Zabbix. url: https://www.zabbix.com/zabbix_agent.