UNIVERSIDADE D
COIMBRA

José Diogo Monteiro Reis

REGISTO E CODIFICAÇÃO DA ANAMNESE NA CONSULTA EXTERNA COM RECURSO A NLP

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pela Professora Doutora Maria José Marcelino e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho de 2023

1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

José Diogo Monteiro Reis

Registo e codificação da anamnese na consulta externa com recurso a NLP

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pela Professora Doutora Maria José Marcelino e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho de 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

José Diogo Monteiro Reis

Registering and coding of anamnesis during external consultation using NLP

Dissertation in the context of the Master in Informatics Engineering, specialization in Software Engineering, advised by Professor Maria José Marcelino and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July, 2023

Agradecimentos

Primeiramente, gostaria de agradecer aos meus orientadores, por disponibilizarem o seu tempo, conhecimento e experiência durante a elaboração este trabalho.

Agradeço também a todos os meus colegas de estudo e amigos pelo apoio e incentivo constante.

Finalmente, gostaria de expressar a minha gratidão à minha família pelo seu apoio e compreensão incondicionais. Este trabalho é-lhes dedicado.

Resumo

O principal objetivo deste trabalho é desenvolver uma prova de conceito na forma de uma aplicação móvel que auxilie o médico no seu trabalho de registo de informação durante uma consulta. Este registo, designado por anamnese, é das fases mais importantes de uma consulta médica, pois permite ao médico obter informações valiosas sobre o histórico de saúde do paciente, tal como as suas queixas, sintomas, doenças prévias, entre outros fatores que o ajudam a decidir quais as melhores decisões de tratamento. Para recolher estas informações é usado o método SOAP, acrónimo para subjetivo, objetivo, avaliação e plano, que é uma estrutura usual de registo clínico na qual cada letra do acrónimo corresponde a um campo de anotação com um conjunto de informações inerentes ao mesmo. No âmbito deste trabalho apenas os campos "subjetivo" e "objetivo" deste método serão equacionados devido à limitação de tempo, visto o trabalho se inserir num estágio. Com a intenção de agilizar o processo de consulta, pretende-se que o preenchimento destes campos seja feito usando ditado e que seja possível identificar as queixas/sintomas presentes no campo "subjetivo" e traduzi-las no respetivo código SNOMED (coleção de códigos que representa conceitos clínicos) e as medições/biometrias presentes no campo "objetivo", ambos recorrendo a processamento de linguagem natural (NLP). A lista destas anotações facilita, não só a sua interpretação, como ainda a troca de dados entre sistemas médicos de registo eletrónico.

Palavras-Chave

Anamnese, API, *Cloud*, NLP, Textos clínicos, Redes neuronais, SNOMED, *Speech-to-text*, SOAP

Abstract

The main goal of this work is to develop a mobile application as a proof of concept that helps the doctor during the recording of patient's information during a consultation. This medical recording, known as anamnesis, is one of the most important phases of a consultation, allowing the doctor to obtain valuable information about the patient's health history, such as his complaints, symptoms, previous illnesses, among other factors that help him to take the best treatment decisions. To collect this information the SOAP method is used, an acronym for subjetive, objective, assessment and plan, a widely used method of documentation for healthcare in which each letter of the acronym corresponds to an annotation field with a set of informations related to it. The objective of this work only covers the "subjective" and "objective" fields of this method, due to time constraints since the study is part of an internship. With the intention of speeding up the consultation process, it is intended that these fields be filled out using dictation and that it is possible to identify the complaints/symptoms present in the "subjective" field and translate them into the respective SNOMED code (a collection of codes that represents clinical concepts) and the measurements/biometrics present in the "objective" field, both using natural language processing (NLP). The list of these annotations facilitates, not only their interpretation, but also data exchange between electronic health record systems.

Keywords

Anamnesis, API, Clinical Notes, Cloud, Neural networks, NLP, SNOMED, *Speech-to-text*, SOAP

Índice

Acrónimos

AI Artificial Intelligence
AWS Amazon Web Services
BERT Bidirectional Encoder Representations from Transformers
CER Character Error Rate
EHR Electronic Health Record
EUA Estados Unidos da América
GDPR General Data Protection Regulation
GPU Graphics Processing Unit
IDE Integrated Development Environment
IoT Internet of Things
JWT Json Web Token
LSTM Long Short-Term Memory
MVC Model-View-Controller
MVP Model–View–Presenter
MVVM Model-View-ViewModel
NLP Natural Language Processing
NER Named Entity Recognition
PaaS Platform as a Service
RNN Recurrent Neural Network
SDK Software Development Kit
SNOMED Systematized Nomenclature of Medicine
SNS Serviço Nacional de Saúde
SOAP Subjetivo, Objetivo, Avaliação, Plano
SPMS Serviços Partilhados do Ministério da Saúde
UI User Interface
USD United States Dollar
WER Word Error Rate

Lista de Figuras

Lista de Tabelas

Capítulo 1
Introdução

O presente documento descreve o trabalho resultante do estágio realizado na empresa **MedicineOne**, sob orientação do colaborador **Pedro Faustino,** destacado da equipa de desenvolvimento da mesma empresa, e pela Professora Doutora do Departamento de Engenharia Informática, **Maria José Marcelino**.

No primeiro capítulo são abordados os objetivos do estágio, bem como a razão pelo qual foi proposto pela empresa. Este capítulo está dividido em 4 secções. A primeira introduz a empresa, a sua posição no mercado e a sua missão. A segunda pretende explicar o racional que levou à proposta. A terceira enuncia a proposta de estágio, designadamente os seus objetivos. Finalmente a quarta explica uma restrição tecnológica.

1.1 Empresa

A MedicineOne, Life Sciences Computing é uma empresa que desenvolve *software* para a área da saúde fundada em 1988 [19].

Nessa altura, a sua primeira aplicação "Consultórios" tinha um objetivo claro: auxiliar e facilitar as tarefas dos médicos. Mais tarde renomeada para M1, foi em 1990 introduzida no Centro de Saúde de Celas de Coimbra e permitia que os utentes tivessem um processo clínico eletrónico. Nos anos seguintes a equipa cresceu e aplicação evoluiu, passando a suportar mais módulos, tal como o de enfermagem.

Recentemente, o *software* M1 já chegou a mercados de vários países como Angola, Cabo-Verde e Brasil, mas é no setor privado do mercado nacional que atualmente se encontra a sua maior fatia de clientes. Alguns deles são: Grupo Luz Saúde, Grupo Lusíadas e Grupo Sanfil. Conta agora com vários módulos (Figura 1) que respondem às necessidades, não só dos profissionais e auxiliares de saúde, como também do pessoal administrativo.

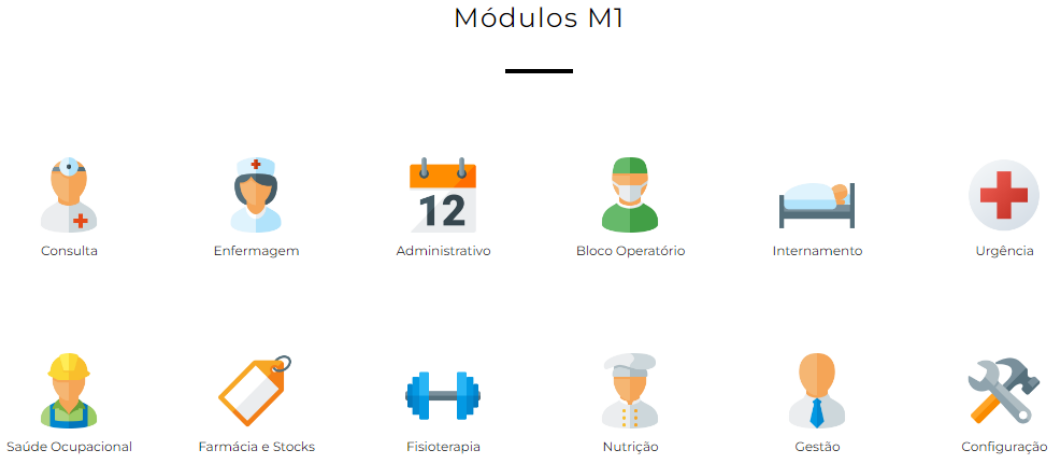


Figura 1 - Módulos do M1, MedicineOne

O M1 faculta ainda uma aplicação "M1 Handy" disponível para Android e iOS.

Existem alguns competidores tais como: SPMS com o *software* SClínico, mas que tem a sua operação no Serviço Nacional de Saúde; Glintt também com *software* para hospitais e clínicas [44].

A missão da empresa é "ajudar os profissionais de saúde a atingirem o seu máximo potencial na prestação de cuidados e apoiar as pessoas na melhoria da sua qualidade de vida" [19] e visa ser uma referência no desenvolvimento de software inteligente para a área da saúde.

1.2 Racionalização da proposta

A proposta de estágio foi feita no sentido de melhorar um processo realizado pelos profissionais de saúde. Devido ao crescente contacto que estes profissionais têm com os *softwares* de apoio à sua ação, são variados os pontos que podem ser melhorados à medida que avança a tecnologia.

Assim, no caso da MedicineOne, foi inicialmente proposto um desenvolvimento visando um dos processos realizados aquando do internamento do doente: registo de escalas de avaliação do estado de saúde.

O estado de saúde de um doente que se encontra internado é monitorizado pelas equipas de enfermagem constantemente e registadas algumas avaliações, são elas: risco de queda; risco de formação de úlcera por pressão; avaliação da dor em doentes não comunicantes; avaliação do nível de dependência do doente. Era objetivo simplificar a ação de registo das avaliações de monitorização clínica, permitindo que o enfermeiro o fizesse recorrendo apenas a um ditado em linguagem natural.

No entanto, numa perspetiva de negócio, a empresa decidiu focar-se num processo diferente, mais prioritário, o que resultou numa pequena alteração da área de ação em relação ao inicialmente previsto (Secção 3.4).

Assim, a proposta final deste estágio evoluiu para o seguimento de uma ação médica realizada aquando de uma consulta: o **registo da anamnese.**

A anamnese é a primeira etapa de uma entrevista médica, que visa o registo de informações sobre o doente, as suas queixas e o seu histórico de saúde. Este registo é fundamental para guiar o diagnóstico médico e, posteriormente, o seu plano de tratamento [36].

Estas anotações clínicas são feitas utilizando um método designado pelo acrónimo SOAP [25], que denota as seguintes categorias:

- **S**ubjetivo – inclui os motivos de consulta, os sintomas, queixas e sentimentos;
- **O**bjetivo – contém informações factuais como os resultados de exames e medições;
- **A**valiação – deve integrar a avaliação que o médico faz dos problemas da pessoa identificados na consulta;
- **P**lano – anexa o plano de intervenção acordado entre médico e doente, os procedimentos realizados/propostos e a terapêutica.

Este ato é efetuado sempre que ocorre uma consulta e os registos efetuados são guardados no *software* designado. No entanto, principalmente no campo "Subjetivo", estes registos

são feitos em texto livre tornando difícil e moroso, por exemplo, fazer um sumário dos sintomas e queixas do doente. Além disto, é uma ação que consome tempo de consulta, reduzindo o tempo de interação com o doente [33]. É apresentado um exemplo de um registo SOAP feito usando o *software* M1 na Figura 2.

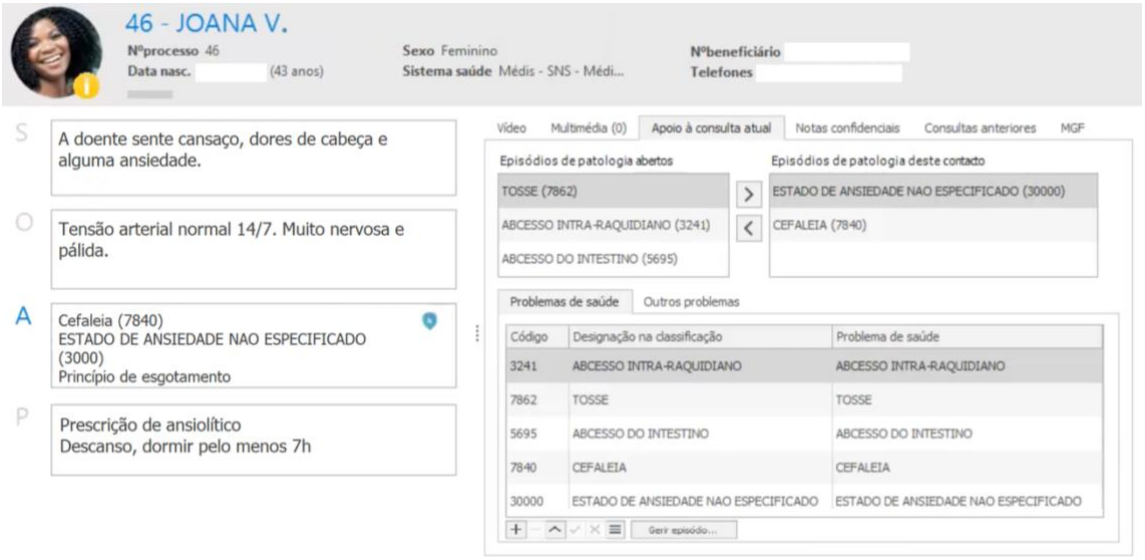


Figura 2 - Exemplo de um registo SOAP no *software* M1

1.3 Proposta de estágio e objetivos

O objetivo deste estágio passa por anotar os textos clínicos registados durante uma consulta por um médico usando o método SOAP, em específico as componentes:

- Subjetivo – identificando códigos SNOMED CT referentes às patologias do doente;
- Objetivo – identificando resultados de medições e biometrias (ver lista completa no Apêndice B).

Apenas estas componentes "S" e "O" são alvo do trabalho, visto que o processamento de todos os campos iria implicar uma complexidade e tempo de desenvolvimento demasiado elevados, tendo em conta a duração do estágio.

Também como objetivo é previsto o desenvolvimento de uma aplicação para dispositivos móveis iOS (mantendo o seguimento da proposta inicial), como prova de conceito e que permita integrar os registos SOAP e o serviço de anotação.

Esta aplicação deve possibilitar aos médicos o preenchimento dos registos através de um **ditado**, reconhecendo o que foi dito recorrendo a mecanismos de *speech-to-text*. Depois de registados os textos clínicos, serão anotados segundo a tipologia do seu campo (ver acima).

SNOMED CT, por extenso *Systematized Nomenclature of Medicine Clinical Terms*, (ou apenas SNOMED) é uma coleção de terminologias clínicas representadas por códigos, que permite traduzir não só patologias e procedimentos (entre outros), mas também relações entre os conceitos [53].

A Figura 3 mostra o exemplo de um código referente a uma patologia, decomposto em componentes relacionados, também eles representados por códigos.

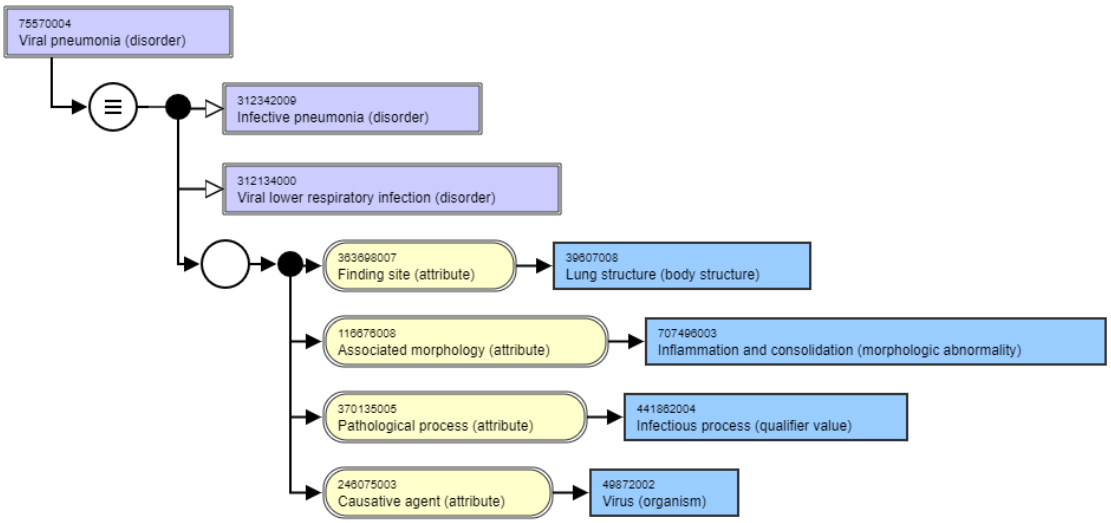


Figura 3 - Exemplo dos códigos relacionados a uma patologia gerado a partir da ferramenta *online* SNOMED CT Browser

A introdução destes códigos nos registos eletrónicos "facilita a estruturação e a interoperabilidade entre sistemas de informação e permite a codificação, armazenamento, troca e agregação de dados" (SNOMED CT E Interoperabilidade Semântica Nos Sistemas de Informação Da Saúde, 2014 [49]), daí os motivos da sua introdução.

Como as áreas do SOAP são registos em texto livre, é expectável que a sua análise e anotação seja feita recorrendo a mecanismos de processamento de linguagem natural ou (usando a designação mais comum) em inglês – ***Natural Language Processing***, termo referenciado pela sigla **NLP**.

Em suma, como resultado do estágio deve ser desenvolvida uma *app* para iOS que permita aos médicos, durante a anamnese, fazer registos clínicos através de ditado usando o método SOAP, para que depois sejam processados e anotados.

1.4 Aplicação iOS

Dado que foi objetivo desenvolver uma aplicação para iOS faz sentido entender a razão desta escolha por parte da empresa.

Os motivos apresentados são simples, dado que o desenvolvimento está integrado no contexto de estágio, não iria haver tempo para desenvolver uma aplicação para ambas as plataformas usadas pela empresa: Android e iOS. Assim, o objetivo segue a tecnologia priorizada internamente na empresa: iOS.

Também o motivo de escolher um dispositivo móvel tem a ver com o facto de, ao contrário de alguns computadores, terem incluído um microfone e por isso possibilitarem o uso do áudio captado para processamento de ditado.

1.5 Estrutura do documento

Tendo em conta a introdução inicial, contextualização da proposta de estágio e objetivos, seguem-se os restantes capítulos, estruturados da seguinte forma:

- **Capítulo 2** apresenta os *softwares* existentes no mercado que vão de encontro aos objetivos e justifica o desenvolvimento de um novo produto. Tendo como partida os mecanismos de NLP que estão na base dos *softwares* anteriormente descritos, é analisado o funcionamento desta tecnologia e comparados serviços/bibliotecas que podem ser utilizados para integrar a solução a desenvolver;
- **Capítulo 3** descreve a metodologia usada durante o desenvolvimento, bem como as ferramentas e linguagens de programação usadas. Também nele é inserida a calendarização expectável para o decorrer do estágio;
- **Capítulo 4** especifica as *user stories* e requisitos funcionais/não funcionais;
- **Capítulo 5** contém uma análise de riscos inerentes ao trabalho, bem como as medidas de contingência caso um dos riscos identificados ocorra. Inclui também um ensaio sobre a dimensão crítica do sistema visto ser um sistema de uso em ambiente clínico;
- **Capítulo 6** tem a definição da arquitetura inicial, descrevendo os componentes do sistema a desenvolver. Ainda neste capítulo é feita um estudo sobre qual dos serviços *cloud* será usado para fazer o reconhecimento de códigos SNOMED e também para tradução dos textos do campo "subjetivo";
- **Capítulo 7** descreve o processo de desenvolvimento da *backend*, dos vários componentes de processamento incluindo o treino dos modelos usados para reconhecimento de entidades, mecanismos de autenticação, disponibilização da API e testes à mesma;
- **Capítulo 8** mostra o desenvolvimento da *frontend* composta pela aplicação. São introduzidos os *mockups* elaborados pela empresa, os detalhes da implementação destacando a componente de ditado e os ecrãs resultantes;
- **Capítulo 9** tem novamente a definição da arquitetura mas desta vez refletindo as mudanças implementadas durante o desenvolvimento. É feita ainda a validação dos requisitos e abordados objetivos no caso de uma futura continuação do trabalho;
- **Capítulo 10** conclui o documento e conta com as reflexões sobre a calendarização prevista, o trabalho realizado e a aprendizagem adquirida.

Capítulo 2
Estado da arte

Neste capítulo são apresentadas algumas das tecnologias que podem ser usadas no âmbito da proposta de estágio.

Numa primeira secção vão ser analisadas aplicações de NLP na área da saúde, mais especificamente as que na atualidade se aproximam do objetivo da proposta. Na segunda secção explica-se brevemente o que é e qual a tecnologia que permite o processamento de linguagem natural. Nas três secções seguintes são enunciados serviços e recursos de programação que permitem fazer NLP, bem como as suas funcionalidades, vantagens e preços: a terceira secção aborda serviços *cloud* genéricos, a quarta secção as principais bibliotecas e a quinta soluções que estejam direcionadas para a área da saúde. As restantes secções abordam os serviços de suporte que poderão vir a ser usados: serviços de *speech-to-text*, tradução e *backend*.

2.1 NLP na área da saúde

Cada vez mais os profissionais de saúde passam uma grande parte do seu tempo a fazer registos sobre os doentes. Com isto advém uma acrescida dificuldade de fazê-lo em tempo útil (o tempo de consulta é um exemplo) e cada vez é necessária mais informação e maior detalhe, para que resulte um melhor diagnóstico.

Para tentar colmatar estes aspetos e ajudar os profissionais, progressivamente são usadas mais tecnologias, como a Inteligência Artificial (AI), em vários campos da saúde. Seja para melhorar a rapidez e precisão de diagnóstico – com aplicações tais como a análise de estudos clínicos, imagens médicas para a deteção de anomalias como cancros ou outras patologias [29] – mas também para facilitar os processos de registo.

Com o foco de melhorar os sistemas de registos eletrónicos dos pacientes (EHR) cada vez mais são aplicadas as técnicas de *Machine Learning* (ML) e *Natural Language Processing* (NLP), áreas da Inteligência Artificial, nomeadamente para auxiliar na escrita e análise de textos clínicos.

Atualmente já existem vários *softwares* ou serviços usados em ambiente clínico que aplicam estas tecnologias para auxílio nesta tarefa, na sua maioria noutros países como os Estados Unidos da América. Já "em Portugal é um tema relevante e que assume uma importância, cada vez maior, no setor da saúde" (SPMS, Inteligência Artificial Na Saúde, 2019 [27]). São exemplos desses *softwares*:

- 3M M*Modal [11], G2 Speech [13], Nextgen Healthcare [18], Nuance Dragon Medical [32] têm funções de preenchimento de registos médicos usando a voz;
- Suki AI [51] que é um assistente virtual ao qual o médico pode pedir a criação de notas clínicas e ditá-las, com a capacidade de identificar códigos ICD-10;

- S10 AI [46] e DeepScribe [15] que traduzem a conversa entre médico e paciente nos vários termos usados (seja sintomas, patologias ou tratamentos) também permitindo identificar códigos clínicos como SNOMED ou ICD-10.

No entanto, estas soluções estão pensadas para o mercado americano, não integram com os sistemas de registo eletrónico portugueses, nem suportam a língua portuguesa. Há, por isso, interesse em desenvolver sistemas adaptados às nossas necessidades.

2.2 NLP – Breve explicação

Na secção anterior foram dados exemplos de várias aplicações no setor da saúde que permitem análise de notas clínicas (tal como o que se pretende fazer no decorrer do estágio) e todas estas têm na sua base NLP. Deste modo, passa-se a introduzir esta tecnologia.

NLP é a tentativa de, através de texto, inferir algo sobre este, tal como classificar os assuntos abordados, reconhecer as entidades presentes ou o sentimento implícito (tom positivo ou negativo). Ou seja, é a possibilidade de um sistema entender o significado das palavras no texto e relacioná-las, tal como um humano faz. Para encontrarmos alguma relação entre as palavras temos de conseguir fazer uma análise semântica, baseada no significado e contexto em que estão inseridas. No contexto das formações efetuadas na Pluralsight, a Figura 4 representa tipos de operações que podem ser efetuadas usando NLP e a Figura 5 representa anotações resultado da operação de extração de entidades.

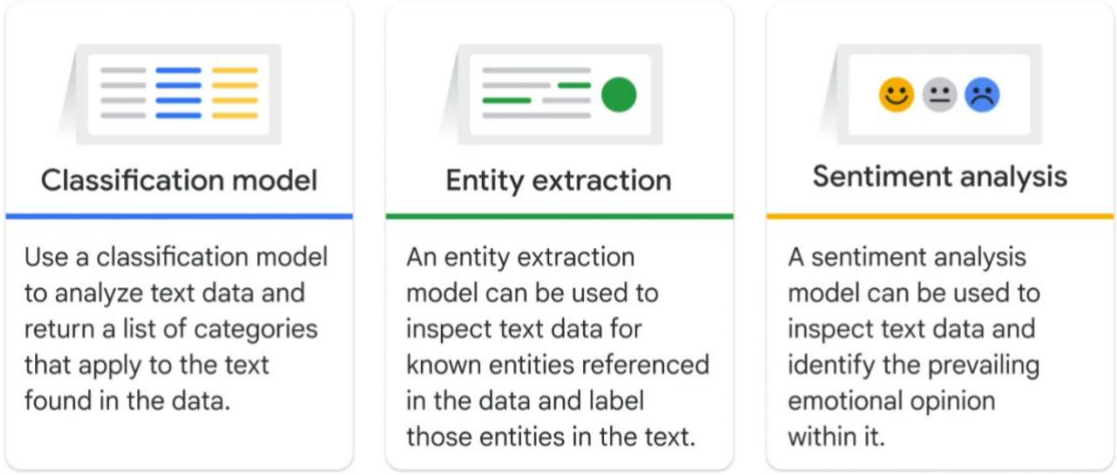


Figura 4 – NLP on Google Cloud, Pluralsight [24]

Figura 5 - Analysing text with Amazon Comprehend, Pluralsight [39]

NLP é uma componente da Inteligência Artificial que terá começado a ser estudada depois da segunda guerra mundial, visto que se sentiu uma necessidade de automatizar o processo de tradução [38]. No entanto, esta tarefa provou-se mais complicada do que parecia, pois as traduções literais não faziam sentido gramaticamente. Até aos anos 90, a área teve um crescimento moderado e eram usados principalmente modelos estatísticos para representar as ocorrências de palavras num contexto.

Em 2001 foi proposto o primeiro modelo que recorria a *deep learning* (área de ML que se foca na aprendizagem autónoma através de grandes quantidades de dados) e que usava **redes neuronais** e a área começou a evoluir rapidamente [16]. Isto porque deixou de ser necessário indicar ao computador quais as caraterísticas dos dados a ter em conta (*features*), pois este descobre-as a partir da rede neuronal.

Ao longo da última década, estes modelos foram-se aperfeiçoando e surgiram aplicações que usamos no dia a dia como o Google Translate ou a Apple Siri. É também esta evolução que permite o seu uso noutras áreas, como a medicina, permitindo oferecer novos serviços, contribuindo para melhorar a qualidade de vida dos utentes [28].

Redes neuronais

As redes neuronais, tal como o nome indica, são compostas por vários elementos (neurónios) que nada mais são que funções matemáticas, como por exemplo: uma transformação (uma função que devolve o somatório dos produtos entre as variáveis de *input* e um peso) e uma função de ativação (uma função não linear que, por exemplo, devolve zero no caso do seu *input* ser negativo e um no caso de ser positivo). Estes elementos podem ter ligações entre si e, juntos, podem ser dispostos em camadas.

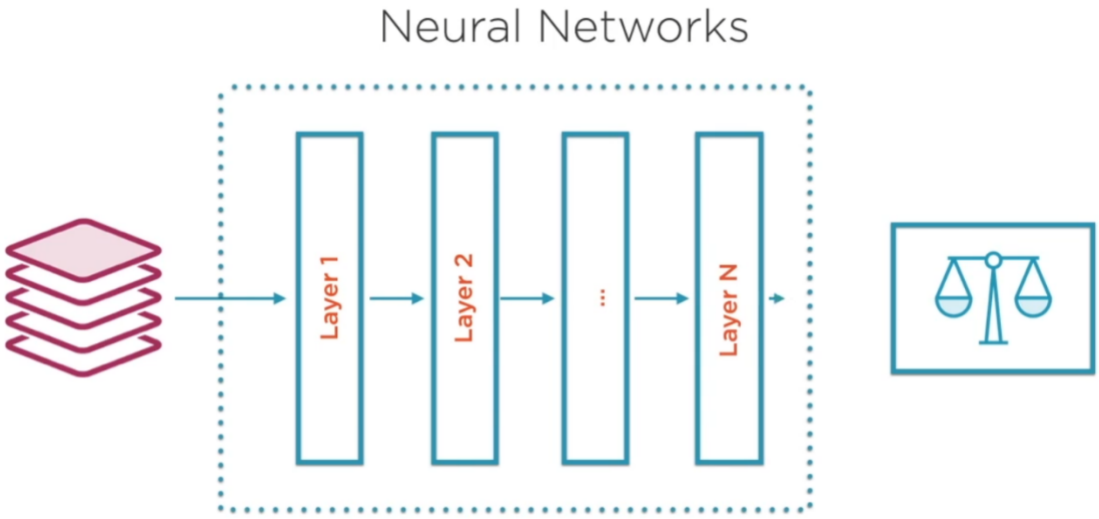


Figura 6 – Natural Language Processing with PyTorch, Pluralsight [43]

Um modelo é uma rede neuronal, à qual podemos passar *inputs* e receber uma classificação, que pode ser de vários tipos: binária no caso de querermos classificar pertença; um vetor no caso de querermos classificar segundo um conjunto de rótulos.

Por exemplo, classificar um livro segundo as categorias [“ação”, “romance”, “fantasia”] iria resultar num vetor com uma probabilidade para cada uma destas categorias.

Vemos um modelo como uma função *black-box*, ou seja, do nosso ponto de vista apenas são relevantes os dados de entrada/saída e não o seu funcionamento interno.

Fundamentalmente, existem dois passos para a implementação e verificação da *accuracy* de um modelo para classificar um livro segundo categorias (exemplo acima).

- Fase de treino
- Fase de teste

Em primeiro, partimos de um *dataset* e dividimo-lo em casos de treino e de teste. Este passo acontece para que tenhamos um conjunto de casos que o modelo ainda "não viu", de modo que não haja dependência entre a *accuracy* do modelo e os casos de teste.

Depois pegamos nos casos de treino e passamo-los pela rede neuronal, que internamente irá fazer um processo iterativo de ajuste dos parâmetros dos neurónios para tentar aproximar as suas previsões de uma classificação com as esperadas.

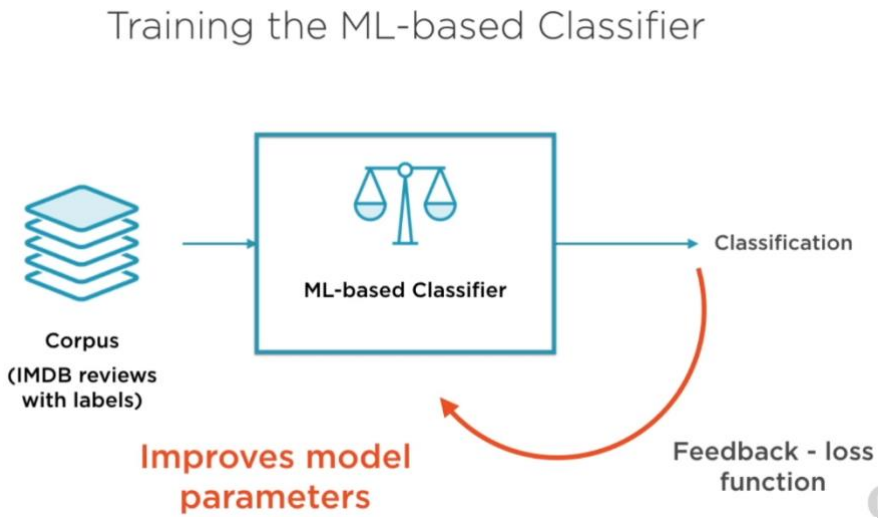


Figura 7 - Natural Language Processing with PyTorch, Pluralsight [43]

Esta fase de treino não é trivial existindo vários fatores que influenciam a capacidade de aprendizagem (por exemplo, diferentes funções de ativação que podem ser usadas ou o número de camadas escondidas).

Na fase de teste, vamos pegar na divisão do *dataset* (previamente classificado como vimos) e damo-lo como *input* ao modelo. Comparando a classificação do modelo com a real, conseguimos ter uma ideia da sua precisão (*accuracy*).

Representação dos dados

Visto que as redes neuronais funcionam como funções, para NLP precisamos de uma forma de representar o texto de forma numérica. Vamos ver as representações mais usuais usando o exemplo: "O rapaz ganhou o jogo". Primeiro temos de separar o texto nos seus constituintes (palavras) e criar um vocabulário. Após este processo existem várias formas de representar as ocorrências de uma palavra.

One-hot-encoding é uma das técnicas mais fáceis de converter texto, na qual cada palavra diferente é traduzida num vetor de booleanos com o mesmo tamanho que o vocabulário, no qual apenas a palavra a representar está ativa. No exemplo, temos um vocabulário que

pode ser um vetor com cada palavra [“o”, “rapaz”, “ganhou”, “jogo”], a representação da palavra “rapaz” seria [0,1,0,0] e a frase seria uma matriz composta pelos vários vetores. Esta forma, apesar de simples, introduz um problema de armazenamento, visto que temos uma matriz com vários valores zero que têm que ser representados.

Bag-of-words é uma forma de representação que visa melhorar o problema da forma anterior, usando a frequência ou a ocorrência de uma palavra para a representar. No exemplo, o vocabulário poderia ser representado da mesma forma por um vetor e a frase seria representada também por um único vetor com as frequências [2,1,1,1]. Apesar de melhorar o problema de armazenamento, o vetor tem o mesmo tamanho que o vocabulário e perdemos a noção de sequência temporal. Em ambas as representações, também não conseguimos inferir quaisquer relações entre as palavras no vocabulário.

Word embeddings é uma forma de representação que tem em vista a relação entre as palavras e para isto são usados pontos no espaço. Este espaço pode ter várias dimensões correspondentes a caraterísticas relacionadas com a palavra. No exemplo, se considerarmos uma dimensão género, a palavra rapaz vai se aproximar mais de masculino [21].

Figura 8 - Exemplo da representação de uma das dimensões de *Word Embeddings*

Deste modo a palavra rapaz poderia ser representada no espaço por um vetor, no qual a primeira posição é referente à dimensão género. Assim, podemos inferir relações entre os elementos de um vocabulário, por exemplo usando a distância e sentido dos vetores que ligam as palavras no espaço. No caso das palavras: homem e mulher, rei e rainha – os vetores de cada um dos pares serão semelhantes.

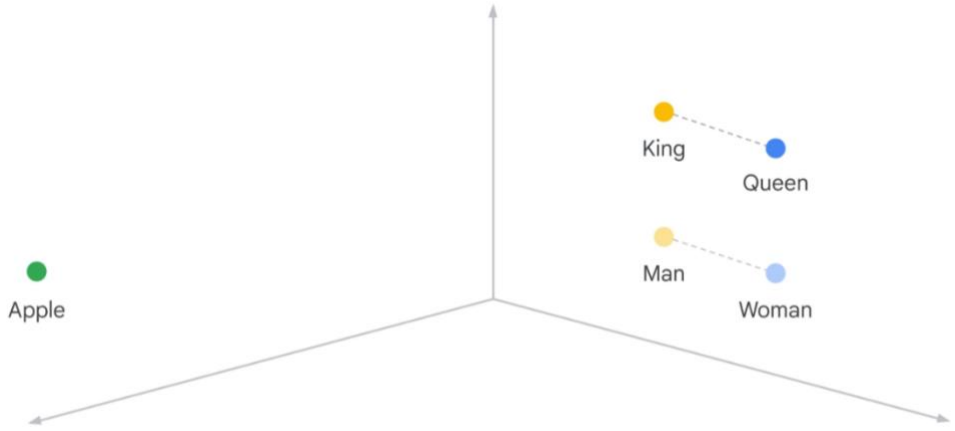


Figura 9 – *Word embeddings*, NLP on Google Cloud, Pluralsight [24]

Apesar de um conceito mais abstrato, esta forma de representação foi um grande avanço na área de NLP existindo vários modelos como: Word2vec da Google, GloVe de Stanford e FastText do Facebook, prontos a usar e com várias dimensões (quanto maior a quantidade de dimensões, melhor a *performance* e maior o tamanho do modelo) [31].

Redes avançadas

Atualmente existem modelos com base em diferentes tipologias de redes neuronais para conseguir extrair informação do texto de forma mais precisa, tais como: redes neuronais recorrentes (RNN), memória longo-curto prazo (LSTM), ou o Google BERT.

Uma das formas de conseguirmos obter o contexto de uma frase é perceber a sequência de palavras. O mesmo acontece quando se tenta completar uma frase. Por exemplo, se tivermos a frase "Estados Unidos da ____" conseguimos perceber que a próxima palavra será "América".

As RNN tentam integrar memória para conseguir ter em conta a sequência, dependendo o *input* de uma camada escondida do *output* das anteriores. Isto tem aplicações como a classificação de sentimento, na qual percebemos o contexto para saber se a frase tem uma conotação negativa ou positiva.

No entanto, este tipo de rede sofre de memória de curto prazo. Se utilizarmos o exemplo "O Mike estudou nos Estados Unidos da América. […] Ele fala ___" mesmo que tenhamos um parágrafo conseguimos perceber que a próxima palavra será "inglês", mas uma RNN não conseguirá perceber. Para colmatar este facto, foram desenvolvidas as redes LSTM, que são também RNN, mas têm uma arquitetura diferente que inclui um outro *pipeline* (Figura 5) para a memória de longo prazo, tal como se fosse outro valor de *input* na camada escondida (camada escondida é uma abstração das camadas que estão entre a primeira e última). Neste as informações de contexto passam entre as várias camadas, mas sofrem menores alterações, como se se lembrassem por mais tempo.

O BERT foi lançado pela Google em 2018 para ser usado no Google Search, para apresentar resultados de pesquisa semelhantes ao input do utilizador. É um modelo que faz uso de redes neuronais para reconhecer o contexto e significado das frases, indo além dos modelos anteriores dado que usa *Transformers*.

Estes *Transformers* são modelos de redes neuronais que conseguem ter em conta, não só as palavras anteriores, mas também as próximas num processo paralelizado [34]. Neste processo usa *word embeddings* e um mecanismo chamado *attention* (atenção em português), que atribui pesos diferentes a diferentes palavras que são consideradas mais importantes. Por exemplo, na frase "O cão é muito peludo" partindo da palavra "cão" a palavra "peludo" estará fortemente relacionada tendo um peso maior comparado, por exemplo, à palavra "muito" que é mais genérica (seguindo uma lógica: pensando na palavra "peludo" podemos logo pensar em "cão". Mas pensando na palavra "muito", não pensamos facilmente na palavra "cão". O peso da atenção varia desta forma).

Esta paralelização e melhor *performance* fez com que os *transformers* estejam atualmente na base dos sistemas modernos de NLP [2]. O BERT é apenas um dos sistemas pré-treinados que usa esta tipologia de rede, sendo outros exemplos o GPT-2 e GPT-3.

Atualmente é usada a técnica de *transfer learning*, que tem por base retreinar um modelo pré-treinado. Partimos desse modelo genérico e vamos continuar a treiná-lo usando um *dataset* mais específico, o que vai melhorar a precisão quando usado neste domínio mais restrito.

Não obstante, existem vários serviços que disponibilizam modelos de NLP pré-treinados alojados na *cloud*, os quais podem ser acedidos e integrados numa aplicação via uma API, apresentando um conjunto de vantagens.

Componentes de NLP

Dado que os objetivos do trabalho têm por base as duas primeiras componentes do SOAP (capítulo 1.3) e inerente ao tipo de conteúdo destes campos, também podemos dividir em duas categorias os serviços de NLP necessários para o seu processamento:

- Componente para reconhecer códigos SNOMED;
- Componente de NLP genérico para reconhecer outras entidades.

Assim, no decorrer das próximas secções vão ser apresentadas tecnologias para fazer este processamento de modo que se possa fundar as escolhas realizadas.

2.3 Serviços de NLP na *cloud*

Como foi referido, treinar um modelo não é trivial e isso deve-se a vários fatores. Desde a elaboração das funções de transformação e ativação, a sequência das várias camadas e o número de camadas. Mas além destes, também o facto de serem necessários dados para treinar o modelo (quantos mais os casos de treino, melhor será o modelo) e ser garantida a qualidade dos dados, que têm de estar bem identificados ("*garbage-in-garbage-out"* - o modelo pode ter uma boa precisão, mas se for treinado com casos incorretos é inútil). Todas estas variáveis levam a que o processo de treinar um modelo seja um de "tentativa-erro" muito moroso e que pode nunca vir a ser viável [45].

Existem várias vantagens generalizadas com o uso de um serviço na *cloud* para fazer NLP, destaca-se a de ter o modelo pré-treinado e pronto a usar. Estes serviços já oferecem um modelo com boa precisão, treinado a partir de um grande conjunto de dados e que vai melhorando à medida que é usado (os dados introduzidos servem como casos de treino para alimentar o modelo e, assim, melhorá-lo). Têm ainda outras vantagens, como: maior poder computacional dedicado a este tipo de processamento; maior capacidade de escalonamento; custo menor a curto-prazo (visto que se poupa o investimento inicial em *hardware* e tempo usado para treinar o modelo).

Aquando da escrita deste documento são múltiplos os serviços na *cloud* que podem ser usados para efetuar NLP. Serão abordados os mais populares resumindo as suas funcionalidades e destacando as suas vantagens, idiomas em que estão disponíveis e preços (à data de escrita).

Amazon Comprehend

O Amazon Comprehend faz parte do leque de serviços da Amazon Web Services e permite fazer reconhecimento de entidades, classificação por temas, análise de sentimento, extração de frases-chave, deteção de idioma e análise sintática [41]. Além disto, permite que se treine o modelo para reconhecer entidades personalizadas e termos específicos.

Está disponível em 12 idiomas entre eles: português, inglês, francês, alemão e espanhol.

Quanto ao preço debitado, é medido em unidades - "*unit*", equivalente a 100 carateres de texto. Cada chamada à API é cobrada com um mínimo equivalente a 3 unidades.

Tem um nível gratuito para até 50 mil unidades de texto (5 milhões de caracteres) por API por mês, apenas para novos clientes durante o primeiro ano (e exclui entidades e classificação personalizadas). A partir daí, o preço varia consoante a funcionalidade. Com quantas mais unidades por mês se fizer o processamento, mais barato fica.

O reconhecimento de entidades e deteção de linguagem têm preços que vão de 0.0001 USD por unidade (até 10 milhões de unidades) a 0.000025 USD por unidade (mínimo 50 milhões de unidades). O processo de treino do modelo tem custos de 3 USD por hora.

Microsoft Azure Cognitive Services

Cognitive Service for Language é o serviço homólogo no ecossistema da Microsoft Azure e permite também o reconhecimento de entidades, análise de sentimento, deteção de idiomas, tendo também a possibilidade de treinar o modelo para reconhecer termos específicos [30]. Além disto, a mesma API dá-nos a possibilidade de fazer um modelo de pergunta-resposta (útil para *chatbots*), tem um módulo de compreensão de linguagem natural, destinado a ser integrado em aplicações móveis e dispositivos IoT e pode sumariar textos extraindo as frases mais relevantes.

Está disponível em 124 idiomas incluindo: português de Portugal, português do Brasil, inglês do Reino Unido, inglês dos Estados Unidos, francês, alemão e espanhol. Nem todos os idiomas têm suporte para todos os módulos.

Na questão monetária, o débito é feito por "*text records*", equivalentes a 1000 carateres de texto.

Até 5 mil *text records* (5 milhões de carateres) por mês e até 1 hora de treino para entidades personalizadas, o uso é grátis. A partir deste volume de dados o preço varia consoante os módulos a usar e, em alguns casos, o próprio volume de dados.

No caso do módulo de reconhecimento de entidades e classificação de texto, o preço é de 5 USD por cada 1000 unidades de texto. E o treino avançado do modelo custa 3 USD por hora. Já para deteção de idiomas o preço varia entre 1 USD por 1000 unidades de texto por mês (até meio milhão de unidades) e 0.25 USD por 1000 unidades de texto por mês (mínimo 10 milhões de unidades).

Existem ainda pacotes que juntam alguns módulos e oferecem preços mais competitivos.

Google Cloud Natural Language

Tal como os dois serviços acima, também a Google Cloud oferece soluções de NLP [12]. As funcionalidades são também semelhantes, oferecendo: análise sintática, análise de entidades incluindo personalizadas, análise de sentimento e classificação de conteúdos.

Está disponível em 10 idiomas: inglês, espanhol, japonês, chinês (simplificado e tradicional), francês, alemão, italiano, coreano, português e russo.

O serviço está dividido em 2 APIs que oferecem diferentes funcionalidades: API Natural Language que integra análise sintática, análise de entidades, análise de sentimento e classificação de conteúdo; AutoML que pretende oferecer uma classificação mais específica ao treinar o modelo e inclui extração de entidades personalizada, análise de

sentimento personalizada e classificação de conteúdo personalizada. Sendo assim, cada uma destas API apresenta diferentes preços.

Para a API Natural Language uma chamada com um documento até 1000 carateres conta como unidade.

Existe um plano grátis até um limite de unidades por mês, que no caso de classificação de conteúdo é de 30 mil unidades e nos restantes casos é de 5 mil unidades. Depois deste volume os preços variam.

No caso de análise de entidades, tem preços desde 1 USD por 1000 unidades de texto por mês (até 1 milhão de unidades) e 0.25 USD por 1000 unidades de texto por mês (mínimo 5 milhões de unidades).

Já para a API AutoML, os preços aplicados são os do serviço mais abrangente de inteligência artificial VertexAI. A forma de medir a utilização neste serviço é um pouco diferente das referidas até ao momento, sendo o preço de operação baseado no *hardware* negociado e nas horas de utilização. A máquina mais barata na Europa tem um preço de 0.240580 USD por hora. O processo de treino tem um custo de 3.30 USD por hora.

IBM Watson

Tal como as opções anteriores, também a IBM oferece serviços *cloud* para processamento de linguagem natural [55]. Com a API Natural Language Understanding conseguimos fazer classificação de conteúdos, análise de sentimento, extração de entidades, extração de palavras-chave, relações entre entidades na frase e análise sintática. Também existe a possibilidade de criar modelos personalizados usando o Watson Knowledge Studio.

Está disponível em 13 línguas: incluindo o português.

O preço é feito em unidades, sendo que uma unidade é constituída por até 10 mil carateres.

Existem dois planos de preços: Lite e Standard. O primeiro é o plano grátis, que permite processar até 30 mil unidades por mês e ter 1 modelo de classificação e WKS (Watson Knowledge Studio) grátis. Já o segundo tem custos que vão de: 0.003 USD por chamada (até 250 mil unidades) a 0.0002 USD por chamada (mínimo de 5 milhões de unidades). Cada modelo de classificação personalizado tem um custo de 25 USD e um modelo no WKS custa 800 USD.

NLP Cloud

O serviço NLP Cloud é uma forma de alojar e treinar modelos de NLP na *cloud* [1]. Podemos adequar o modelo à tarefa, visto que existem vários modelos pré-treinados. Está disponível ainda um plano no qual podemos treinar o modelo ou até importar um modelo personalizado. Podemos efetuar várias operações, tais como: classificação de texto, análise de sentimento, análise sintática e extração de palavras-chave e entidades.

As linguagens disponíveis dependem do modelo escolhido, sendo que o português pode ser uma possibilidade.

O preço depende do *hardware*, número de ligações em simultâneo pretendidas, bem como o número de ligações por minuto e varia entre 29 USD e 2499 USD por mês.

Existe um plano grátis que permite apenas um pedido simultâneo e não tem garantias de processamento.

Tabela comparativa

De modo a mais facilmente comparar as diferentes alternativas de serviços de NLP na *cloud*, será apresentada uma tabela que contém a indicação de suporte das linguagens e funcionalidades mais relevantes, bem como o preço.

Tabela 1 - Comparação de soluções de NLP genéricas

	Amazon Comprehend	**Microsoft Azure Cognitive Services**	**Google Cloud Natural Language**	**IBM Watson**	**NLP Cloud**
Está disponível em português?	✔	✔	✔	✔	**Depende do modelo**
Permite extração de entidades?	✔	✔	✔	✔	✔
Permite extração de palavras-chave?	✔	✔	✔	✔	✔
Permite análise de sentimento?	✔	✔	✔	✔	✔
Permite treinar o modelo para reconhecer entidades e conteúdo personalizados?	✔	✔	✔	✔	✔
Tem um serviço adaptado ao vocabulário médico?	✔ **Amazon Comprehend Medical**	✔ **Text analytics for health**	✔ **Healthcare Natural Language API**	✔ **IBM Watson Annotator for Clinical Data**	✘
Preço para treinar o modelo por hora	**3 USD**	**3 USD (Grátis a primeira hora)**	**3,30 USD**	**Plano grátis (Limite 1 modelo) Plano pago – 800 USD por mês**	**Grátis**
Considerando 5 mil chamadas por mês, cada chamada com mil carateres					
Preço para fazer extração de entidades	**= 5 USD (Grátis 1º mês)**	**Grátis**	**Grátis**	**Grátis**	**Diferente métrica de medida**
Preço para fazer extração de entidades personalizadas	**= 25 USD**	**Grátis**	**Dependente de vários fatores**	**Grátis**	**Diferente métrica de medida**
Considerando 10 mil chamadas por mês, cada chamada com mil carateres					
Preço para fazer extração de entidades	**= 10 USD**	**= 10 USD**	**= 10 USD**	**Grátis**	**Diferente métrica de medida**
Preço para fazer extração de entidades personalizadas	**= 50 USD**	**= 50 USD**	**Dependente de vários fatores**	**Grátis**	**Diferente métrica de medida**
Considerando 50 mil chamadas por mês, cada chamada com mil carateres					
Preço para fazer extração de entidades	**= 50 USD**	**= 50 USD**	**= 50 USD**	**= 150 USD**	**Diferente métrica de medida**
Preço para fazer extração de entidades personalizadas	**= 250 USD**	**= 50 USD**	**Dependente de vários fatores**	**= 950 USD**	**Diferente métrica de medida**
Considerando 5 milhões de chamadas por mês, cada chamada com mil carateres					

Preço para fazer extração de entidades	**= 2500 USD**	**= 1500 USD**	**= 2500 USD**	**= 5000 USD**	**Diferente métrica de medida**
Preço para fazer extração de entidades personalizadas	**= 25000 USD**	**= 25000 USD**	**Dependente de vários fatores**	**= 5800 USD**	**Diferente métrica de medida**

Conclusões

Através da análise da tabela anterior (Tabela 1) é percetível que todas as soluções têm funcionalidades em comum, destacando-se a de extração de entidades personalizadas.

Quanto ao serviço NLP Cloud pode ser definido como um PaaS visto que serve para alojar e treinar modelos e o preço é definido pelo *hardware* a utilizar. Dos restantes serviços, a Microsoft e a IBM têm as soluções gratuitas mais interessantes. No entanto, no caso de se ultrapassar o limite do plano grátis, o panorama altera-se sendo que a solução da IBM é a mais cara por chamada e as da Amazon e da Microsoft são semelhantes em termos de preço.

Também o tamanho de cada chamada pode influenciar o preço da solução, visto que as unidades de medida de uma chamada são diferentes:

- Amazon - 1 unidade = até 100 carateres
- Microsoft – 1 unidade = até 1.000 carateres
- Google – 1 unidade = até 1.000 carateres
- IBM – 1 unidade = até 10.000 carateres

Os serviços distinguem-se pelo volume de dados que conseguem processar de uma só vez. No caso de termos chamadas com um pequeno volume de dados (constantemente menores que 1000 carateres) o serviço da Amazon pode ser o mais em conta. Se ao invés, forem consideradas chamadas com um grande volume de dados (constantemente maiores que 1.000 carateres e até 10.000) o serviço da IBM será mais vantajoso.

2.4 Bibliotecas de NLP

Apesar das vantagens de usar um serviço da *cloud*, no caso de aplicações mais específicas ou com outros requisitos de segurança (visto que usando um serviço na *cloud* temos de enviar pedidos para um provedor externo), podem ser usadas bibliotecas.

A grande vantagem desta abordagem é a possibilidade de configurar as camadas das redes neuronais e treiná-las permitindo obter uma solução à medida das necessidades. Também existem vantagens de rapidez no desenvolvimento quando comparado à criação de uma rede neuronal "do zero". Partindo de modelos pré-treinados (*transfer learning*), estes já reconhecem linguagem natural por vezes em várias línguas (havendo também modelos dedicados a vocabulários específicos), sem ter de se perder tempo a treinar e fornecer casos de teste.

Já quando comparados a um serviço *cloud* a conveniência, escalabilidade e preços (especialmente tendo em conta o *hardware* necessário para o processamento de NLP e treino da rede neuronal) são algumas caraterísticas que pesam aquando da sua escolha face a uma abordagem deste tipo.

Não obstante, visto serem uma opção viável, serão apresentadas algumas bibliotecas de NLP.

NLTK

Natural Language Toolkit (NLTK) é uma biblioteca *open-source* que tem em vista fornecer ferramentas para o processamento de linguagem natural. Conta com funções para "*tokenization, stemming, tagging, parsing*", as quais podem ser juntas num *pipeline* e assim fazer processamento mais avançado.

Apesar da sua facilidade de uso, o NLTK fornece apenas algumas funções, sendo voltado para a aprendizagem em contexto académico ao invés de um ambiente de produção.

SpaCy

SpaCy é outra biblioteca *open-source* para NLP, mas é bastante mais avançada, contando *pipelines* treinados para várias linguagens e suporte para GPU (o que acelera bastante o processamento).

É uma ferramenta poderosa visto que permite com facilidade partir de um modelo pré-treinado, como por exemplo o BERT, treiná-lo para vocabulário específico e fazer extração de entidades, classificação ou outras operações mais avançadas.

Spark NLP

Spark NLP é a biblioteca de NLP mais usada em ambiente de produção, sendo também *open-source* e contando com várias linguagens. Uma das suas vantagens é ser altamente escalável, visto que foi construída tendo por base o Apache Spark. Permite à partida fazer várias operações como classificação de sentimento, extração de entidades, deteção de linguagem e geração de texto.

Além disto tem um módulo dedicado a NLP para a área da saúde – Spark NLP for Healthcare - que permite reconhecer por exemplo, medicações e patologias no texto. No entanto, este é um módulo pago por subscrição.

2.5 Serviços de NLP adaptados ao vocabulário médico

Ao passo que os serviços anteriores são genéricos, permitindo múltiplas funcionalidades, também existem serviços adaptados a necessidades específicas. Visto que um dos objetivos é extrair do diálogo as queixas e respetivos códigos SNOMED, não é justificável de um ponto de vista de negócio estar a treinar um modelo para responder a esta necessidade quando já existem estes serviços e bibliotecas de NLP ajustados ao vocabulário médico. Além disso, estes serviços garantem um maior nível de privacidade dos dados. Serão apresentados os principais concorrentes.

Amazon Comprehend Medical

Existe uma versão do serviço Amazon Comprehend - "Comprehend Medical" que está adaptada ao vocabulário médico e permite extrair automaticamente sintomas, patologias, medicações e tratamentos [20]. Garante também que os dados que passam pelo serviço

não são usados para melhorar outros modelos (algo que não é garantido pelo serviço normal).

Este serviço tem também um nível gratuito para até 85 mil unidades de texto (sendo que 100 carateres = 1 unidade) durante o primeiro mês de utilização. A partir daí, o preço varia consoante a API a utilizar. Até 1 milhão de unidades o preço vai de 0.00025 a 0.01 USD por unidade, sendo que para extrair códigos SNOMED será de 0.0075 USD.

No entanto, este serviço não está disponível para o idioma português (apenas para inglês).

Azure Text Analytics for health

Um dos módulos do serviço Text Analytics - "Text analytics for health" compara-se ao "Comprehend Medical" na medida em que permite identificar medicações, diagnósticos, sintomas, entre outros. Está disponível em 7 idiomas dos quais o português, ainda que o modelo desta linguagem esteja em *preview* e não deva ser usado num ambiente de produção.

Até 5 mil chamadas por mês não tem custos, a partir daí tem preços desde 25 USD por 1000 unidades de texto por mês (até meio milhão de unidades).

Google Cloud Healthcare Natural Language API

Também a Google Cloud tem uma API adaptada ao vocabulário médico – "API Healthcare Natural Language" [26], disponível apenas em inglês. Os preços desta são medidos usando múltiplos de 100 face ao número de carateres de um pedido, sendo que por 1000 carateres é cobrado 0.10 USD. Por exemplo, uma solicitação com 800 carateres tem um preço de 0.08 USD. Os primeiros 2500 pedidos são grátis.

Não está disponível fora dos Estados Unidos da América.

IBM Watson Annotator for Clinical Data

Este é um dos módulos do serviço "Watson Health" e permite detetar no texto os problemas de saúde mais pertinentes, medicações, procedimentos médicos além de códigos clínicos como SNOMED.

Para ser usado tem planos feitos à medida dos clientes, dependendo dos módulos a incluir e na capacidade de processamento.

Spark NLP for Healthcare

Tal como visto acima, esta é um módulo da biblioteca do Spark NLP que permite fazer reconhecimento de entidades clínicas, tais como medicações, respetiva frequência de toma e dosagem, além de identificar códigos clínicos como SNOMED.

A sua utilização requer um *cluster* Apache Spark pode ser alojado num servidor próprio (desde que tenhamos o *hardware* necessário) ou em alternativa na *cloud* numa máquina virtual em plataformas como o Azure ou AWS. É necessária uma licença para poder usar este módulo.

Tabela Comparativa

Na Tabela 2 serão comparadas as soluções encontradas em relação às suas funcionalidades, mas também aos preços que cobram.

Tabela 2 - Comparação de soluções de NLP com funções de anotação

	Amazon Comprehend Medical	**Azure Text Analytics for health**	**Google Cloud Healthcare Natural Language API**	**IBM Watson Annotator for Clinical Data**	**Spark NLP for Healthcare**
Tipo de serviço	**Cloud**	**Cloud**	**Cloud**	**Cloud**	**Biblioteca**
Está disponível em português?	✗	✓ **Modelo em preview**	✗	✗	✓ **Alguns modelos**
Permite deteção de códigos SNOMED?	✓	✓	✓	✓	✓
Garantias de privacidade dos dados?	✓	✓	✓	✓	**Não se aplica**
Considerando 50 mil chamadas por mês, cada chamada com quinhentos carateres					
Preço para análise extraindo os códigos SNOMED	**= 1875 USD**	**= 1125 USD**	**= 2375 USD**	**Mediante orçamento**	**Depende do *hardware* (Standard Azure Marketplace = 6963 USD)**
Considerando 50 mil chamadas por mês, cada chamada com mil carateres					
Preço para análise extraindo os códigos SNOMED	**= 3750 USD**	**= 1125 USD**	**= 4750 USD**	**Mediante orçamento**	**Igual aos preços acima**

Componente para reconhecer códigos SNOMED

De entre os vários serviços apresentados que permitem fazer NLP e estão adaptados ao vocabulário médico (Secção 2.5), todos eles permitem o reconhecimento de códigos SNOMED.

Destaca-se o serviço Spark NLP for Healthcare que, ao invés de um serviço *cloud,* é uma biblioteca.

Assim, o leque de escolha de serviços *cloud* reduz-se às soluções da: Amazon, Azure, Google e IBM.

Podemos excluir a solução da IBM, visto que é uma solução à medida e os preços desta estão sob orçamento. Também a Google, apesar de disponibilizar as restantes funcionalidades do Healthcare Natural Language API, tem o vocabulário de SNOMED restrito a pedidos a partir dos EUA “The request must also originate from the US.” (Google, 2023, [23]).

Quanto às restantes opções, foi necessário avaliar a sua *performance.* Na tentativa de fazer esta averiguação, foram conduzidos os testes descritos na Secção 6.2.

2.6 Decisão em relação a serviços *cloud* vs bibliotecas

Após terem sido explorados tanto serviços *cloud* (Secções 2.3 e 2.5) como bibliotecas (Secção 2.3) para fazer o processamento de linguagem natural, foi optado por uma abordagem que maximize o uso de soluções *off-the-shelf* (um produto comercial pronto, ao invés de um produto personalizado como é o caso das bibliotecas) com base em **serviços na nuvem**.

Esta decisão tomou-se tendo em conta vários aspetos:

- Existem serviços *cloud* que cumprem os requisitos pretendidos, mais especificamente, que permitem o reconhecimento de códigos SNOMED e que permitem a extração de entidades personalizadas;
- Não será necessário treinar um modelo (no caso do serviço de reconhecimento de códigos SNOMED), dispensando a recolha de dados para este treino, bem como o tempo que seria usado;
- Dado que é objetivo fazer uma aplicação prova de conceito, esta abordagem permite a rápida implementação e consecutiva redução do *time-to-market*;
- A empresa MedicineOne tem preferência por uma solução deste tipo, visto que vai de encontro à sua prática anterior.

2.7 Motores *speech-to-text*

Dado que o objetivo deste trabalho foi fazer uma aplicação que permita fazer registos usando ditado, tornou-se necessário ter um mecanismo para converter o diálogo para texto, de modo a que depois fosse processado.

Era expectável que fosse usado o motor de *speech-to-text* nativo dos dispositivos iOS. No entanto, serão exploradas algumas alternativas que também seriam viáveis de usar, sejam elas serviços, os quais podem ser acedidos através de uma API tal como os serviços Amazon Transcribe, Azure Speech to Text, Google Cloud Speech to Text ou Assembly AI. Ou bibliotecas que podem ser usadas no iOS, tal como o Mozilla DeepSpeech.

Tabela Comparativa

A tabela seguinte (Tabela 3) permite comparar os mecanismos de *speech-to-text* quanto às suas caraterísticas e preços.

Tabela 3 - Comparação de soluções *speech-to-text*

	SFSpeechRecognizer	**Amazon Transcribe**	**Azure Speech to Text**	**Google Cloud Speech-to-Text**	**AssemblyAI**	**Mozilla DeepSpeech**
Está disponível em português?	✔	✔	✔	✔	✔	✔
Permite transcrição assíncrona?	✔	✔	✔	✔	✔	✔

Permite transcrição em tempo real?	✔	✔	✔	✔	✘ **Não em português**	✔
Corre localmente no dispositivo?	✔	✘	✘	✘	✘	✔
Preço por minuto (transcrição assíncrona)	**Grátis**	**= 0.024 USD (até 250 mil minutos)**	**= 0.01666 USD**	**2 planos: - Com Data logging = 0.024 USD - Sem Data logging = 0.016 USD**	**= 0.015 USD**	**Grátis**
Preço por minuto (transcrição em tempo real)	**Grátis**	**Igual aos preços acima**	**= 0.04166 USD**	**Igual aos preços acima**	**Igual aos preços acima**	**Grátis**

Conclusões

No contexto de empresa era expectável que fosse usado o motor *speech-to-text* do iOS – SFSpeechRecognizer da *framework* Speech da Apple [3]. Além de ser a prática anterior da empresa, prende-se com o facto de estar disponível em várias línguas, da qual português e tem a vantagem de, em alguns dispositivos, não requerer *internet* para o processamento.

Não obstante, no seu emprego (registo da anamnese no processo de consulta, Secção 1.2) mostrou-se necessária a escolha de um serviço melhor adaptado ao vocabulário médico, abordado na Secção 8.4.

2.8 Serviços de tradução

Dado que a maioria dos serviços *cloud* não têm suporte para a língua portuguesa, foi necessário traduzir o texto do inglês para português antes de fazer o processamento dos códigos SNOMED.

Com este objetivo foi usado um serviço *cloud* para fazer a tradução e são várias as opções de serviços encontrados. À semelhança do que foi feito nas seções acima, serão aferidas as caraterísticas dos principais serviços.

Tabela Comparativa

Na Tabela 4 são apresentados alguns serviços de tradução, incluindo dos principais provedores *cloud*, bem como a sua capacidade de traduzir português, o seu cumprimento das normas de privacidade de dados quanto ao armazenamento e uso para melhoria dos modelos. São ainda comparados os preços relativamente ao uso dos serviços.

Tabela 4 – Comparação de serviços de tradução na *cloud*

	Amazon Translate	**Azure Cognitive Services Translator**	**DeepL API**	**Google Cloud Translation AI**	**IBM Watson Language Translator**
Traduz de português?	✔	✔	✔	✔	✔
Cumpre com as normas de privacidade dos dados (GDPR) tendo em conta o armazenamento?	✔ **Necessário um acordo de negócio**	✔ **Necessário um acordo de negócio**	✔ **Apenas DeepL API Pro**	✔ **Necessário um acordo de negócio**	✔
Plano grátis (nº de carateres por mês)	**2 milhões nos primeiros 12 meses**	**2 milhões**	**500 mil (DeepL API FREE)**	**500 mil**	**1 milhão**
Preço por milhão de carateres (fora do plano grátis)	**= 15 USD**	**= 10 USD**	**= 20 EUR**	**= 20 USD**	**Preço mediante orçamento**

Conclusões

Tendo em conta que o vocabulário contemplado nas traduções tem alguma especificidade, prova-se benéfico escolher o serviço cuja tradução seja mais satisfatória. Além disto, pela análise da tabela é percetível que as soluções da Azure e Amazon são as que ficam mais em conta e aquelas para as quais a empresa conseguiu disponibilizar um ambiente de teste. Assim, serão comparadas na Secção 6.3.

2.9 Serviço de Backend

Como se entende pela análise dos serviços acima, serão usados mais que um serviço *cloud* para processar um pedido (englobando serviços de tradução, NLP e possivelmente *speech-to-text*).

Assim, fazer o processamento dos sucessivos pedidos para cada um dos serviços na aplicação móvel não é uma opção que faça sentido, por vários motivos:

- A aplicação móvel teria de conhecer todas as API, bem como tratar os dados retornados por elas e, no caso de estarem encadeadas, formatá-los para poder fazer a próxima chamada;
- Estando mais suscetível a quebras de rede, a aplicação pode perder a ligação entre chamadas e ter de repetir chamadas (mais chamadas significam mais custos);
- Fazer os pedidos a partir da *app* móvel aumenta o tráfego na rede ao qual está ligada e acrescenta um *delay* a cada chamada.

Convém que se tenha uma única API a qual possa ser invocada e apenas receber os resultados.

Também por se tratar de uma aplicação como prova de conceito, o uso de uma abordagem *serverless* apresenta várias vantagens:

- É uma forma mais rápida de implementar, quando comparada a um servidor dedicado;

- Permite poupar nos custos, visto que usualmente são serviços pagos por chamada;
- O código fica mais simples, visto que apenas temos funções que são dedicadas ao desempenho de uma tarefa.

Todos estes aspetos contribuem para um *time-to-market* mais rápido.

Tabela Comparativa

Com o objetivo de implementar funções *serverless* pode-se optar por diversos serviços disponíveis no mercado, cujas linguagens de programação disponíveis e preços praticados são comparados na Tabela 5.

Tabela 5 - Comparação de serviços de *serverless* functions

	Amazon Lambda	Azure Functions	Google Cloud Functions	IMB Cloud Functions	Oracle Cloud Functions
Linguagens de programação	**Java, Go, PowerShell, Node.js, C#, Python e Ruby**	**C#, TypeScript, JavaScript, F#, Java, Python e Powershell**	**JavaScript, Go, Java, C#, Ruby, PHP e Python**	**JS/NodeJS, Swift, Java, Python, Ruby, Go, PHP e .NET**	**Java, Python, Node, Go e Ruby**
Plano grátis por mês	**• 1 milhão de pedidos • 400 mil GB-s**	**• 1 milhão de pedidos • 400 mil GB-s**	**• 2 milhões de pedidos • 400 mil GB-s**	**• Pedidos ilimitados • 400 mil GB-s**	**• 2 milhão de pedidos • 400 mil GB-s**
Preço por GB-segundo por mês	**=0,0000166667 USD**	**= 0.000016 USD**	**= 0.0000025 USD**	**= 0.000017 USD**	**= 0.00001417 USD**
Valor adicional por milhão de pedidos por mês	**= 0.20 USD**	**= 0.20 USD**	**= 0.40 USD**	**Grátis**	**= 0.20 USD**

Conclusões

Pela tabela percebe-se que os serviços disponibilizam algumas das mais populares linguagens de programação [9] e ainda que as várias opções diferem a nível de preços nos planos pagos, mas são equiparáveis no caso dos planos grátis.

Para o contexto do estágio, a escolha foi condicionada à opção Amazon Lambda, visto que a infraestrutura atual da empresa já faz uso deste mesmo serviço.

2.10 Resumo do capítulo

Neste capítulo foram apresentados os *softwares* presentes no mercado que vão de encontro aos objetivos pretendidos para o trabalho e, provando-se que nenhum destes responde às necessidades colocadas (principalmente por não estarem disponíveis em português), justificou-se o desenvolvimento de uma nova solução. De modo a entender o funcionamento dos *softwares* analisados, foi introduzida a principal tecnologia que os compõe – NLP. De seguida, foram analisados vários serviços e bibliotecas que permitem implementar esta tecnologia, tendo em conta a finalidade pretendida (reconhecimento de códigos SNOMED e anotação de biometrias).

No seguimento do capítulo foi justificada a decisão do uso de serviços *cloud*, bem como analisados os componentes acessórios que deverão ser usados para formar a solução final.

Capítulo 3
Metodologia

Este capítulo pretende explicar a metodologia de trabalho usada e como foi feito o planeamento de tarefas neste estágio. Serão descritos: a metodologia de desenvolvimento escolhida, acompanhada da justificação desta decisão e as ferramentas utilizadas no auxílio do desenvolvimento. O capítulo inclui ainda a calendarização estimada.

3.1 Scrum

Tendo em vista que foi um projeto de *software* no qual se pretendia desenvolver uma aplicação móvel e um serviço de *backend*, fez sentido que o desenvolvimento fosse feito de forma iterativa, tendo envolvido o cliente (MedicineOne) neste processo para verificação e validação. Também porque foi um produto que visava a melhoria da eficácia de um processo (registo de notas clínicas), a usabilidade deve foi tida em conta, sendo avaliada e melhorada.

Assim, a metodologia escolhida foi a mesma usada pela equipa de desenvolvimento da empresa – **Scrum**.

Esta está inserida na categoria das **metodologias ágeis**, visando uma rápida e suave resposta face a propostas de alteração, ou seja, consente a mudança de alguns requisitos durante o decorrer do desenvolvimento [47]. Dessarte permite a obtenção de um produto que vai mais ao encontro das expectativas do cliente.

Scrum é uma *framework* que define uma estrutura de desenvolvimento ágil, na qual uma iteração corresponde a um *sprint*. Os *sprints* têm associada uma duração (por exemplo de 1 mês) e durante um *sprint* é expectável que sejam efetuadas reuniões diárias (*daily scrum*) nas quais é avaliado o progresso diário e discutidos os próximos passos. Existem artefactos nomeadamente: *product backlog* – que define as funcionalidades a serem desenvolvidas no decorrer do projeto; *sprint backlog* – que define as funcionalidades a serem desenvolvidas no decorrer do *sprint*. A equipa tem de ser constituída por elementos com as funções: *scrum master* – responsável por coordenar a equipa e garantir que o desenvolvimento decorre consoante o planeamento; *product owner* – representa os clientes e *stakeholders* assim como as suas necessidades e prioriza as funcionalidades. No final de um *sprint* é feita uma reunião para rever o progresso do mesmo, assim como planear o próximo.

Os *sprints* neste projeto tiveram a duração de 2 semanas, com reuniões no final do *sprint*. Os artefactos produzidos ficaram internos à empresa. Não foram destacadas funções visto que a equipa apenas foi constituída pelo estagiário e orientadores.

3.2 Ferramentas auxiliares

De modo a auxiliar a gestão de projeto, as consequentes tarefas e o trabalho realizado foram utilizadas algumas ferramentas descritas abaixo.

JIRA

Trata-se de um software online desenvolvido pela Atlassian [6] que possibilita funções de gestão de projeto, acompanhamento de problemas e documentação. É usado por equipas de desenvolvimento de software principalmente para planear e organizar o trabalho. Pode ser usado em equipas ágeis (como Scrum e Kanban), equipas DevOps, entre outras.

Foi ser usado no decorrer do desenvolvimento para designar as tarefas a realizar durante os *sprints* e acompanhar o estado de cada tarefa, assim como documentar o seu progresso.

GIT

É um sistema de controlo de versões, que auxilia o registo de alterações feitas no código ao longo do tempo. Assim, permite a vários desenvolvedores trabalhar em conjunto podendo facilmente reverter alterações indesejadas ou juntar código ao mesmo projeto [10].

Agregado a um repositório este mecanismo permitiu guardar o histórico de todo o desenvolvimento, assim como o código-fonte.

VS Code

Editor de código desenvolvido pela Microsoft que suporta múltiplas linguagens, extensões e integração com o Git [54]. Foi o principal editor de código durante o desenvolvimento.

Xcode

Apresenta-se como um ambiente de desenvolvimento integrado (IDE) da Apple para macOS usado para desenvolver aplicações para o ecossistema dessa empresa [4]. Foi usado no desenvolvimento da aplicação iOS.

Postman

Ferramenta de desenvolvimento de API's que permite executar testes enviando pedidos HTTP e obtendo as respostas [40]. Os pedidos podem ser guardados, numa tentativa de automatizar a testagem. Foi utilizado para efeitos de teste da API.

Jupyter Notebook

Representa um editor *web* que combina código, resultados e explicações num único documento [42]. Foi uma ferramenta usada para avaliar num contexto prático os serviços *cloud* e bibliotecas a ser usados no decorrer do estágio.

3.3 Linguagens de programação

Python

Linguagem de programação escolhida para ser usada na construção da API e conexão aos serviços *cloud*, assim como para testar em ambiente prático as tarefas de NLP.

Esta decisão prende-se com o facto de existir uma grande quantidade de bibliotecas para esta linguagem: seja dos serviços da AWS (como o boto3), Azure, Google, mas também em questões de *machine learning* (com bibliotecas como NumPy, Pandas, entre outras). Além disso, também é usada no âmbito da empresa.

Swift

Para desenvolver a aplicação iOS foi usada a linguagem Swift. Além de ser uma linguagem de programação mais moderna que Objective-C (a outra alternativa para programação nativa em iOS) permite também alcançar mais *performance* que esta.

3.4 Calendarização

De modo a programar os trabalhos, foi realizada uma calendarização que tenta estimar as tarefas a realizar, permitindo que se estabeleçam metas e contribuindo para uma noção de progresso. Foram elaborados planeamentos para os primeiro e segundo semestres apresentados nas subsecções seguinte, tendo em conta os objetivos e uma estimação dos mesmos. As figuras estão presentes no Apêndice A para facilitar a leitura.

Primeiro semestre

Inicialmente, foi feita uma calendarização que tinha em vista os objetivos do estágio propostos inicialmente, apresentada na Figura 10.

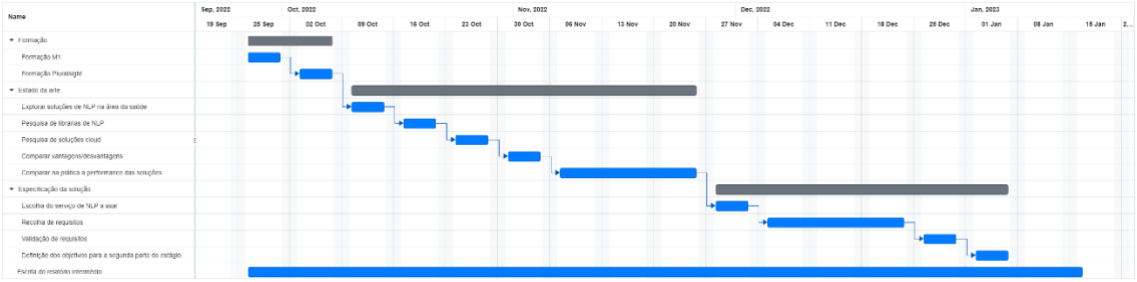


Figura 10 - Calendarização antes da alteração de objetivos

No entanto, por motivos de prioridade no negócio foram feitas alterações aos objetivos da proposta (Secção 1.3) que alteraram ligeiramente a calendarização. Estas foram sugeridas a 6 de dezembro e, a partir desta data, o planeamento mudou ligeiramente como se apresenta na Figura 11.

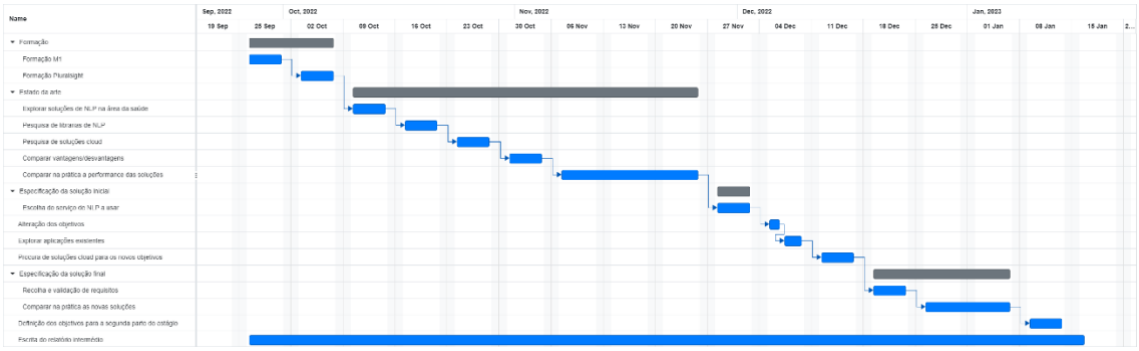


Figura 11- Calendarização depois da alteração de objetivos

Segundo semestre

A calendarização para o segundo semestre foi elaborada após a conclusão do primeiro. Nela foi projetado estudar e definir o restante da arquitetura, bem como as fases de desenvolvimento e testes. Foi ainda previsto alcançar uma aplicação funcional cumprindo os requisitos mínimos (MVP) a 21 de abril. As datas e respetivos objetivos estão refletidos na Figura 12.

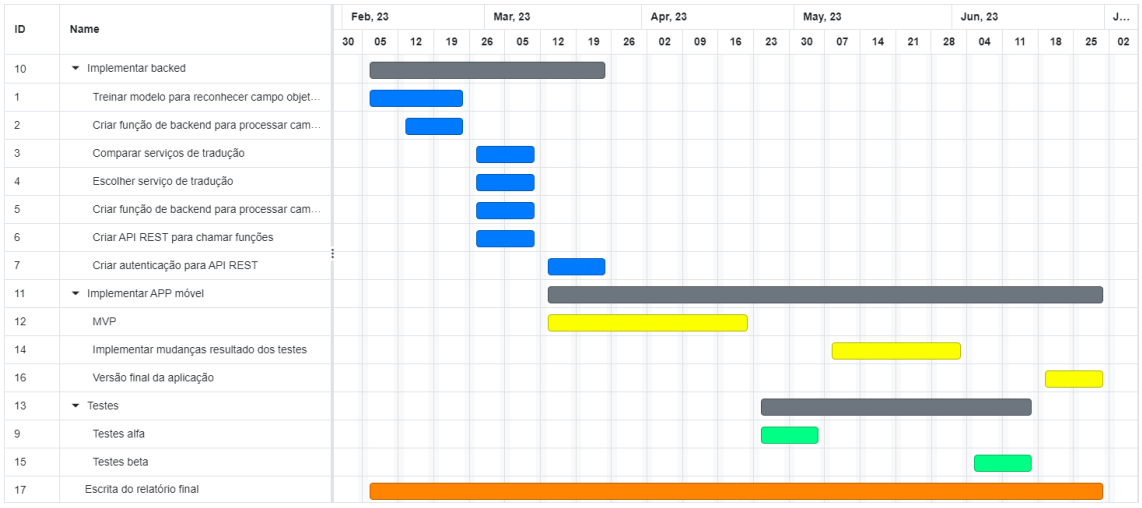


Figura 12 - Calendarização do segundo semestre

Na Secção 10.1 é feita a reflexão sobre o cumprimento da calendarização efetuada.

Capítulo 4
Análise de requisitos

Neste capítulo será sintetizado o âmbito do desenvolvimento e apresentados os principais *stakeholders* do problema, bem como as principais *user stories* e requisitos.

4.1 Âmbito

Anteriormente no documento foram apresentados a proposta de estágio e o racional por detrás da mesma. Agora numa perspetiva de desenvolvimento, percebe-se que o âmbito da proposta foi o desenvolvimento de uma **aplicação** como "**prova de conceito**", que permita cumprir a premissa de: **permitir ao médico durante a consulta ditar as queixas e biometrias do paciente, para que estas sejam identificadas automaticamente**.

4.2 Stakeholders

Os principais *stakeholders* envolvidos no processo são:

- Médico;
- Paciente;
- Provedores *cloud;*
- MedicineOne.

4.3 User stories

De modo a entender as funcionalidades requisitadas pelo cliente, vamos introduzir as *user stories*. Estas estão organizadas pela área de registo médico, segundo o método SOAP:

- **S**ubjetivo;
- **O**bjetivo;
- **A**valiação;
- **P**lano.

De notar que os objetivos apenas visam os dois primeiros campos, tal como referido no Capítulo 1.

Subjetivo

De seguida apresentam-se as *user stories* relativas ao campo "subjetivo".

US.S-1.0 : Ditar campo subjetivo	
Descrição	**Como médico quero ditar as queixas do doente, para que sejam preenchidas no campo "Subjetivo".**
Critérios de aceitação	• **Estar disponível o mecanismo *speech-to-text* para que o médico possa começar a ditar** • **Reconhecer corretamente as frases ditas, transformá-las em texto e preencher o campo "Subjetivo"**
Ambiente de testes	• **Ter um dispositivo com a aplicação, microfone e estar ligado à internet**
Detalhes técnicos	• **Reconhecimento de termos técnicos da medicina e regionalismos** • **Apenas estar disponível quando ligado à internet**

US.S-2.0 : Reconhecer códigos SNOMED CT	
Descrição	**Como médico quero que a aplicação analise o texto para que sejam reconhecidos os códigos SNOMED CT nele presentes.**
Critérios de aceitação	• **A API de análise de texto estar disponível** • **Identificar corretamente os códigos SNOMED CT correspondentes às queixas presentes no texto**
Ambiente de testes	• **Ter um dispositivo com a aplicação, o campo "Subjetivo" preenchido e estar ligado à internet**
Detalhes técnicos	• **O serviço apenas está disponível em inglês e o texto está em português, logo tem de haver uma tradução** • **Os códigos têm uma hierarquia na qual pode um código pode abranger mais queixas ou ser mais específico. Assim, é difícil encontrar o código pretendido.**

US.S-3.0 : Remover código reconhecido	
Descrição	**Como médico quero poder remover um código da lista de códigos reconhecidos para que esse código não seja associado ao registo.**
Critérios de aceitação	• **Ao remover um código, este desaparece da lista**
Ambiente de testes	• **Ter a lista de códigos identificados**
Detalhes técnicos	-

US.S-4.0 : Editar código reconhecido	
Descrição	**Como médico quero poder editar um código da lista de códigos reconhecidos para que o possa substituir por outro.**
Critérios de aceitação	• **Ao editar um código, este é substituído na lista**
Ambiente de testes	• **Ter a lista de códigos identificados**
Detalhes técnicos	• **Os médicos não sabem de memória os códigos, logo devem ser apresentados códigos alternativos.**

Objetivo

À semelhança das anteriores são apresentadas *user stories*, no entanto relativas ao campo "objetivo".

US.O-1.0 : Ditar campo objetivo	
Descrição	**Como médico quero ditar as biometrias do doente, para que sejam preenchidas no campo "Objetivo".**
Critérios de aceitação	• **Estar disponível o mecanismo *speech-to-text* para que o médico possa começar a ditar** • **Reconhecer corretamente as frases ditas, transformá-las em texto e preencher o campo "Objetivo"**
Ambiente de testes	• **Ter um dispositivo com a aplicação, microfone e estar ligado à internet**
Detalhes técnicos	• **Reconhecimento de números, medidas e unidades** • **Apenas estar disponível quando ligado à internet**

US.O-2.0 : Reconhecer parâmetros das biometrias	
Descrição	**Como médico quero que a aplicação analise o texto para que sejam reconhecidos os parâmetros nele presentes.**
Critérios de aceitação	• **A API de análise de texto estar disponível** • **Identificar corretamente os parâmetros presentes no texto**
Ambiente de testes	• **Ter um dispositivo com a aplicação, o campo "Objetivo" preenchido e estar ligado à internet**
Detalhes técnicos	• **O serviço apenas está disponível em inglês e o texto está em português, logo tem de haver uma tradução**

US.O-3.0 : Remover parâmetro reconhecido	
Descrição	**Como médico quero poder remover um parâmetro da lista dos reconhecidos para que esse não seja associado ao registo.**
Critérios de aceitação	• **Ao remover um parâmetro, este desaparece da lista**
Ambiente de testes	• **Ter a lista de parâmetros identificados**
Detalhes técnicos	-

US.O-4.0 : Editar parâmetro reconhecido	
Descrição	**Como médico quero poder editar um parâmetro reconhecido para corrigir os dados ou unidades nele presentes.**
Critérios de aceitação	• **Ao editar um parâmetro os seus dados são atualizados**
Ambiente de testes	• **Ter a lista de parâmetros identificados**
Detalhes técnicos	• **Podemos ter dados identificados e não saber a que parâmetro correspondem ex.120 corresponde à pressão arterial ou ao peso?**

4.4 Requisitos funcionais

Nas tabelas abaixo serão apresentados os requisitos funcionais priorizados segundo a escala de MoSCoW. Para se entender melhor as iniciais utilizadas representam:

- **M**ust have;
- **S**hould have;
- **C**ould have;
- **W**on't have (this time).

Área de registo	Identificação	Requisito	Prioridade	User story
S	**REQ.S-1.0**	**Ditar texto para que o campo seja preenchido**	**M**	**US.S-1.0**
S	**REQ.S-2.0**	**Reconhecer códigos SNOMED CT a partir do texto**	**M**	**US.S-2.0**
S	**REQ.S-3.0**	**Visualizar quais os códigos reconhecidos**	**M**	**US.S-2.0**
S	**REQ.S-4.0**	**Eliminar um código reconhecido**	**M**	**US.S-3.0**
S	**REQ.S-5.0**	**Editar um código reconhecido**	**S**	**US.S-4.0**

Área de registo	Identificação	Requisito	Prioridade	User story
O	**REQ.O-1.0**	**Ditar texto para que o campo seja preenchido**	**M**	**US.O-1.0**
O	**REQ.O-2.0**	**Reconhecer a altura de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-3.0**	**Reconhecer o peso de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-4.0**	**Reconhecer a frequência cardíaca de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-5.0**	**Reconhecer a frequência respiratória de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-6.0**	**Reconhecer a medição de tensão arterial de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-7.0**	**Reconhecer a saturação de oxigénio de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-8.0**	**Reconhecer a saturação de dióxido de carbono de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-9.0**	**Reconhecer os níveis de glicémia de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-10.0**	**Reconhecer a medição da cintura de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-11.0**	**Reconhecer a medição da anca de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-12.0**	**Reconhecer a medição do perímetro cefálico de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-13.0**	**Reconhecer a medição do perímetro torácico mínimo de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-14.0**	**Reconhecer a medição do perímetro braquial de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-15.0**	**Reconhecer medições de dinamometria de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-16.0**	**Reconhecer a medição de dor de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-17.0**	**Reconhecer a medição da percentagem de massa gorda de um paciente**	**M**	**US.O-2.0**

O	**REQ.O-18.0**	**Reconhecer a medição da prega tricipital de um paciente**	**M**	**US.O-2.0**
O	**REQ.O-19.0**	**Visualizar quais os parâmetros reconhecidos**	**M**	**US.O-2.0**
O	**REQ.O-20.0**	**Eliminar um parâmetro reconhecido**	**M**	**US.O-3.0**
O	**REQ.O-21.0**	**Editar um parâmetro reconhecido**	**M**	**US.O-4.0**

4.5 Requisitos não funcionais

Requisitos não funcionais são restrições às quais o sistema deve obedecer, tratam questões de qualidade do sistema e, por isso, muitas das vezes são chamados de atributos de qualidade (adaptado de Ian Sommerville, 2011 [50]).

Apesar de ser objetivo o desenvolvimento de uma prova de conceito, existem alguns requisitos não funcionais que devem ser tomados em conta. Um requisito deste tipo afeta vários aspetos do sistema. São eles:

Segurança

A segurança consiste na capacidade de o sistema bloquear o acesso de entidades não autorizadas. Ou seja, pretendemos garantir que uma entidade não autenticada não possa aceder a um recurso protegido. São casos: o acesso às funcionalidades da aplicação, bem como aos serviços *cloud* através da API REST.

Fonte	**Utilizador não autenticado**
Estímulo	Faz um pedido à API REST
Artefacto	API
Ambiente	Operação normal
Resposta	1. O sistema bloqueia o acesso; 2. Devolve o erro HTTP 401 correspondente à não autorização.
Métrica de resposta	Utilizador não autenticado não conseguiu aceder a um recurso protegido

Usabilidade

A usabilidade consiste na capacidade de os utilizadores de um sistema alcançarem os seus objetivos de forma eficaz, eficiente e que os satisfaça (adaptado de ISO 9241-11). Desta forma, foi objetivo fazer uma aplicação na qual os utilizadores possam fazer o processamento de um campo com um mínimo de 3 passos.

Fonte	**Utilizador autenticado**
Estímulo	Inicia a captação, dita, termina e processa.
Artefacto	Aplicação
Ambiente	Operação normal com acesso ao microfone concedido
Resposta	1. O sistema transforma o áudio em texto; 2. O sistema envia o texto para a API para processamento; 3. O sistema recebe a resposta e mostra a informação no ecrã.
Métrica de resposta	O processamento de um campo apenas necessitou de três interações.

Capítulo 5
Riscos

Tendo em conta a natureza experimental do trabalho, existem riscos inerentes que podem condicionar o alcance dos objetivos ou provocar mudanças no plano de trabalhos. Foi feito um levantamento destes riscos usando um conjunto de critérios para classificar a sua probabilidade e impacto segundo o formato:

- **Risco:** Descrição concisa e clara, que ajuda a perceber os problemas que podem impedir o projeto de ser bem-sucedido.
- **Probabilidade:** Probabilidade de o risco ocorrer. Segue uma escala de 1 a 5:
    - 1 - Probabilidade Muito Baixa;
    - 2 - Probabilidade Baixa;
    - 3 - Probabilidade Média;
    - 4 - Probabilidade Alta;
    - 5 - Probabilidade Muito Alta.
- **Impacto:** Impacto/Efeito nos objetivos caso o risco ocorra. Segue uma escala de 1 a 5:
    - 1- Impacto Muito Baixo;
    - 2 - Impacto Baixo;
    - 3 - Impacto Médio;
    - 4 - Impacto Alto;
    - 5 - Impacto Muito Alto.
- **Consequências**: Frase concisa que descreva de forma clara o(s) resultado(s) negativo(s) que podem ocorrer derivados do risco.
- **Ações:** Ação associada de modo a diminuir o impacto de um risco.

Identificação	Risco	Prob.	Impacto	Consequências	Ações
RISK-1.0	**Cliente efetua uma alteração nos requisitos**	**2**	**5**	**Face a uma alteração nos requisitos será necessário pesquisar, desenvolver e documentar as alterações. Consequentemente os objetivos e plano de trabalhos serão alterados.**	**CP-1.0**
RISK-2.0	**Nenhum modelo para reconhecer códigos SNOMED tem uma *performance* favorável.**	**2**	**5**	**Não será possível identificar códigos SNOMED usando um modelo disponível na *cloud*.**	**CP-2.0**

RISK-3.0	**Nenhum modelo de tradução tem uma performance favorável.**	**2**	**5**	**Não será possível traduzir os textos para usar um serviço de reconhecimento de códigos em inglês.**	**CP-2.0**
RISK-4.0	**Custos do serviço *cloud são* insuportáveis.**	**3**	**4**	**Será necessário mudar de provedor ou treinar um modelo *offline*.**	**CP-3.0**
RISK-5.0	**Não existem dados para treinar um modelo.**	**4**	**4**	**O modelo pode ter uma fraca *performance*.**	**CP-4.0**
RISK-6.0	**Um modelo personalizado não tem boa *performance*.**	**2**	**5**	**O modelo não irá funcionar como o esperado.**	**CP-5.0**

Medidas de contingência

De modo a prever uma ação no caso de estes riscos impactarem o projeto foram elaboradas medidas de contingência que correspondem às ações da tabela anterior.

CP-1.0 Alteração dos requisitos: Face a uma alteração nos requisitos será necessário reavaliar as tarefas a realizar, bem como reajustar os objetivos. Depois de traçada uma nova estratégia terá de se modificar o plano de trabalhos.

CP-2.0 Nenhum modelo ter *performance* favorável: Se nenhum dos modelos tiver uma *performance* de acordo com o esperado será necessário reajustar as expectativas e perceber se a *performance* atual, apesar de não ótima, é suficiente. Caso o desempenho não seja de todo satisfatório, terá de se partir para o treino de um modelo personalizado. Este processo, além de bastante mais moroso que uma solução *off-the-shelf*, usualmente necessita de uma grande quantidade de dados. Este processo irá provocar uma modificação no plano de trabalhos, bem como nos objetivos que podem ser atingidos.

CP-3.0 Custos do serviço *cloud* são insuportáveis: Se os custos de um serviço *cloud* se tornarem insuportáveis, tem de se procurar uma alternativa mais barata no mercado e que permita manter a funcionalidade. No entanto, ao mudar de provedor pode provar-se necessário fazer um trabalho de configuração e até desenvolvimento adaptado à nova solução, algo que irá alterar o plano de trabalhos. Se nenhuma alternativa resolver este problema, terá que se reavaliar a estratégia de desenvolvimento para perceber se existe possibilidade de alojar um serviço.

CP-4.0 Não existem dados para treinar um modelo: A falta de dados para treinar um modelo pode ser colmatada com uma geração artificial e usando técnicas para aumento de dados, no entanto, não será a solução ótima resultando num modelo menos adaptado ao "mundo real". Nesta abordagem, conseguir casos para geração é normalmente um processo iterativo (visto que não se consegue obter todos os casos de uma só vez), podendo alterar a calendarização.

CP-5.0 Um modelo personalizado não tem boa *performance*: Ao treinar um modelo personalizado, se este não alcançar uma boa *performance*, será necessário fazer uma reflexão sobre os dados, técnicas e tecnologias usadas. Uma mudança de abordagem pode provar-se necessária, afetando o plano de trabalhos.

Dimensão crítica do sistema

Dada a natureza do sistema e o seu uso expectável em ambiente clínico, prova-se necessária uma análise de quão crítico será.

Um sistema crítico é "aquele cuja falha poderá pôr em causa vidas humanas, o ambiente, a organização para o qual opera ou levar a prejuízos financeiros" (adaptado de Ian Sommerville).

O sistema desenvolvido adequar-se-ia a esta definição visto que seria usado no processo de diagnóstico de um paciente aquando de uma consulta, podendo uma falha prejudicar o mesmo. No entanto, no decorrer do estágio o sistema apenas foi uma prova de conceito para poder comprovar se os serviços de NLP e ditado se adequam ao caso de uso. Os registos efetuados usando o sistema desenvolvido não serão introduzidos num sistema real. Além disso, no futuro uma integração com o EHR da empresa (M1) passaria a responsabilidade crítica para este sistema. Em todo o caso, o médico teria de confirmar as informações antes de estas serem introduzidas.

O presente sistema apenas pretende agilizar o processo de registo do médico aquando de uma consulta e não substitui o seu diagnóstico, apenas tem a finalidade de facilitar a introdução de informações. Deste modo, podemos dizer que não será um sistema crítico.

Capítulo 6
Definição da arquitetura inicial

Após terem sido apresentados os serviços que podem ser usados para resolver o problema proposto, neste capítulo apresenta-se a arquitetura projetada para a solução. Serão fundamentadas algumas decisões integrais para o desenho da arquitetura. Em continuação será apresentado um diagrama discriminando os vários componentes da arquitetura. Por fim, irá ser abordado o padrão arquitetural a usar no desenvolvimento da aplicação móvel.

6.1 Planta da solução

Tendo em vista os objetivos colocados, podemos ser aproximados os componentes da solução (ainda que nem todas as tecnologias estejam definidas). Deste modo, será usado o modelo C4 que define um conjunto hierárquico de diagramas: contexto, *container*, componentes e opcionalmente código (ordenados do maior nível de abstração para o menor) – para aproximar uma arquitetura.

Diagrama de contexto

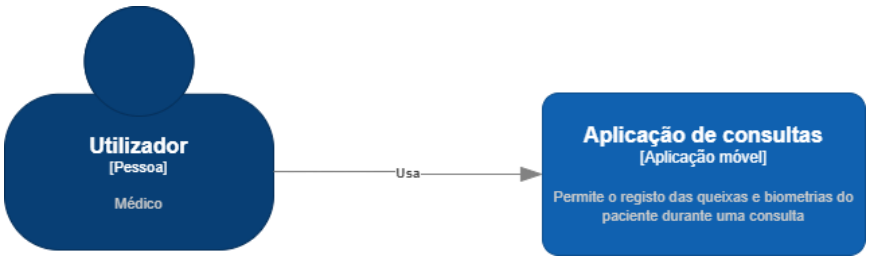


Figura 13 - Diagrama de contexto

O diagrama representado pela Figura 13 tem como foco demonstrar os atores e principais sistemas, permitindo perceber quais deles interagem diretamente com o utilizador.

Diagrama de *containers*

Figura 14 - Diagrama de *containers*

Descendo um grau de abstração, este diagrama (Figura 14) permite perceber quais os principais componentes do sistema: a aplicação móvel, a API de *backend* e os serviços *cloud* usados pela *backend* para o processamento.

6.2 Análise prática dos serviços para reconhecer códigos SNOMED

De modo a tomar uma decisão sobre quais os provedores de serviços usar para fazer o reconhecimento de códigos SNOMED, a *performance* de cada um deles foi de ser analisada: seja o tempo de processamento ou a taxa de acerto (*accuracy*).

A empresa conseguiu ambientes de desenvolvimento orientados às soluções da Amazon e Azure. Para saber qual delas seria a melhor escolha, no corrente caso de uso (reconhecer códigos SNOMED CT), foram elaborados alguns testes de modo a avaliar a precisão e tempo de resposta de cada uma delas. Seguem-se os métodos de teste e resultados.

Dataset

De modo a avaliar a precisão do serviço para reconhecer códigos SNOMED CT, foi preciso algo que permitiu saber se estes códigos estão a ser bem identificados ou não. Neste sentido, foi utilizado um *dataset* público disponível para descarregar em https://doi.org/10.1371/journal.pone.0209547.s002, encontrado a partir do artigo de acesso livre "Qualitative analysis of manual annotations of clinical text with SNOMED CT" [35] e que faz parte dos materiais de um projeto europeu "ASSESS CT" [5].

O artigo pretendia analisar as discrepâncias existentes entre anotações de códigos SNOMED CT feitas por várias pessoas ou em várias linguagens. Visto que estes códigos abrangem um grande leque de conceitos, podem ser utilizados vários códigos para descrever o mesmo problema; no entanto alguns são mais específicos e outros são mais abrangentes. Este facto faz com que exista alguma subjetividade na forma como se atribui um código SNOMED CT a uma patologia.

O *dataset* contém então um conjunto de 60 textos clínicos para várias linguagens (exclui o português), no qual cada termo está associado ao código correspondente. Apresenta-se no formato *excel*, no qual cada linha corresponde a uma palavra e uma das colunas contém o código SNOMED CT (Figura 15).

A	B	C	D	E	F	G
Language	Annotator_ID	Snippet_ID	Sentence_ID	Token	Chunk	CODE
EN	EN1	NL0	6	Physical	6	5880005
EN	EN1	NL0	6	examination	6	5880005
EN	EN1	NL0	6	revealed		
EN	EN1	NL0	6	no		
EN	EN1	NL0	6	signs		
EN	EN1	NL0	6	of		
EN	EN1	NL0	6	heart	7	84114007
EN	EN1	NL0	6	failure	7	84114007
EN	EN1	NL0	6	.		

Figura 15 - Dataset com códigos SNOMED CT anotados

Para o efeito de testar os serviços de anotação, foi feita uma restrição à língua inglesa e comparado se os códigos que foram reconhecidos correspondem a um dos códigos anotados. Deste modo, foi possível comparar se a anotação feita pelo serviço se equipara à de um anotador. Um exemplo de um destes textos seria:

"Your patient was admitted for an excision of a histologically verified malignant melanoma on the left buttock. The tumor was excised under local anesthetic on 25.11.2009 with a safety zone of 3 cm. Additional excision of a nevus paravertebral to the left thoracic spine a nevus on the back of the right hand proximal and ablation of erythematosquamous plaque from the right temple."

De notar que, tal como se pretende mostrar no artigo, a anotação destes códigos está sempre sujeita à subjetividade e o mesmo acontece com estes serviços – visto usarem técnicas de inteligência artificial, a sua anotação será tal qual os exemplos que foram fornecidos para treinar o modelo.

O seu uso é plausível neste contexto, dado que contém anotações de códigos feitas por especialistas e que seriam úteis em contexto médico. Será apresentado qual dos serviços se aproxima mais destas anotações.

Ambiente e métodos de testagem

Com o objetivo de comparar a *performance* de cada serviço, foi usada a mesma linguagem de programação Python e as respetivas API. O código foi feito em Jupyter Notebook.

Desde logo, foi notório que no *dataset* várias palavras foram anotadas com o mesmo código, dado que os códigos incluem noções de lateralidade, temporalidade, entre outros. Também se infere que, os códigos SNOMED CT ao estarem em constante evolução, poderão existir novas entradas desde que o *dataset* foi produzido.

Assim, após obtidos as classificações feitas pelo serviço *cloud*, foram selecionados os códigos correspondentes no *dataset* e comparados. Nesta análise, resultaram códigos que

foram reconhecidos pelo serviço e não estavam presentes no *dataset*, assim como o contrário.

De modo a poder comparar os serviços serão focados os seguintes aspetos (para cada texto clínico):

- Nº de códigos bem identificados – aqueles que foram identificados pelo serviço e correspondem ao código anotado;
- Nº de códigos errados – aqueles que foram identificados pelo serviço mas não correspondem ao código anotado;
- Nº de códigos mal identificados – aqueles que foram identificados pelo serviço mas que não têm anotação;
- Nº de excertos diferentes identificados – conjunto de frases nas quais foram identificados códigos pelo serviço;
- Nº de excertos diferentes anotados – conjunto de frases nas quais foram anotados códigos;
- Relação dos excertos identificados e dos excertos anotados – os excertos de texto no qual foram identificados códigos e que também tinham anotações.

A Figura 16 procura representar visualmente as métricas usadas, sendo o diagrama de correspondência referente aos aspetos de códigos e o diagrama de Venn referente aos aspetos de excertos.

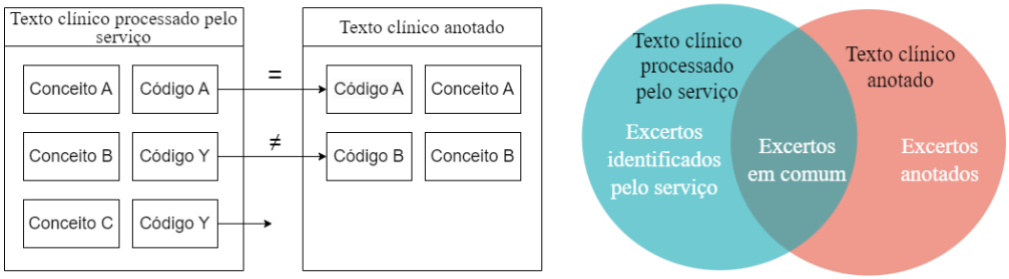


Figura 16 - Diagrama que retrata as diferentes métricas a analisar

Também o tempo de processamento de cada um destes serviços esteve sob análise, medindo o intervalo de tempo desde que se invocava a API do serviço até que esta retornava uma resposta.

Resultados

Após terem sido executadas ambas as API com todos os textos clínicos do *dataset*, foram obtidos os resultados apresentados na tabela.

Tabela 6 - Comparação da precisão dos serviços Amazon Comprehend Medical e Azure Text Analytics for health

	Amazon Comprehend Medical	**Azure Text Analytics for health**
Nº de textos clínicos	**60**	
Nº de códigos diferentes anotados	**1334**	
Dos excertos diferentes anotados para cada texto clínico		
O serviço identificou em média	**89%**	**69%**
Desvio padrão	**45%**	**30%**

Intervalo de confiança de 95%	**[77.6 , 100]**	**[61.4 , 76.6]**
Dos excertos identificados pelo serviço		
Percentagem dos códigos mal identificados (95% confiança)	**14 ± 3.54 %**	**13 ± 3.29 %**
Percentagem dos códigos errados (95% confiança)	**40 ± 4.55 %**	**45 ± 5.57 %**
Percentagem dos códigos bem identificados errados (95% confiança)	**44 ± 4.3 %**	**41 ± 5.57 %**
Percentagem dos códigos em comum com os anotados (95% confiança)	**32 ± 3.8 %**	**23 ± 3.29 %**

A partir dos resultados apresentados na Tabela 6 e na Figura 17 pode-se concluir que o Amazon Comprehend Medical obtém geralmente os melhores resultados, não só classificou mais excertos como os classificou mais corretamente.

De notar que estes resultados foram obtidos usado um *dataset* com 60 textos clínicos, de variados temas da medicina geral.

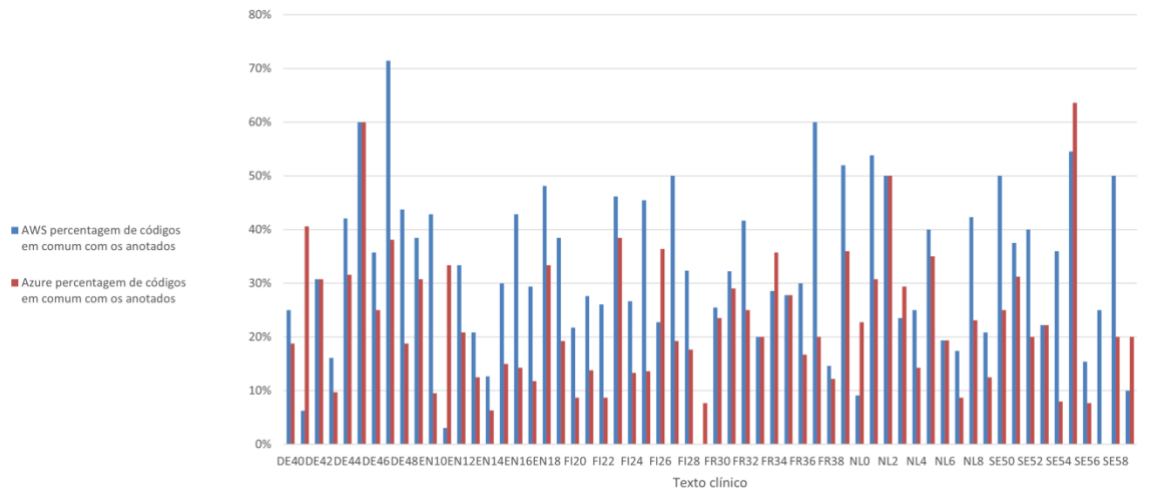


Figura 17 - Percentagem de códigos em comum com o *dataset* anotado por texto clínico

Foi concluído então que a solução da AWS frequentemente se aproxima mais das anotações feitas pelos especialistas. Também outro facto que ajuda a estes resultados é a inclusão de várias alternativas de códigos em cada excerto (para cada são apresentados os *top* 5 códigos SNOMED CT), o que lhe dá mais hipóteses de acertar.

Tabela 7 - Comparação do tempo médio demorado a completar uma chamada dos serviços Amazon Comprehend Medical e Azure Text Analytics for health

	Amazon Comprehend Medical	**Azure Text Analytics for health**
Média do nº de carateres por chamada	**734**	
Dos excertos diferentes anotados para cada texto clínico		
Tempo médio demorado a completar a chamada em segundos (95% confiança)	**1.139 ± 0.158 s**	**6.280 ± 0.39 s**

O serviço escolhido para identificar os códigos SNOMED foi: Amazon Comprehend Medical – não só pela sua *performance*, mas porque o preço é razoável e faz parte do leque de serviços da AWS, algo que já é usado no contexto da empresa.

6.3 Análise prática dos serviços de tradução

Dada a escolha do serviço para identificar códigos SNOMED e tendo em conta que este apenas está disponível em inglês, prova-se necessário ter um serviço de tradução. Também pela natureza dos textos a traduzir, a escolha deste serviço tem que ter em conta a sua *performance* neste domínio.

Neste sentido, usando os ambientes de desenvolvimento fornecidos pela empresa anteriormente referidos, foi elaborada uma estratégia de testagem com a finalidade de avaliar estes serviços. Seguem-se os métodos e resultados.

Método usado

De modo a poder fazer uma análise objetiva da *performance* dos serviços de tradução, existem diferentes abordagens possíveis. No caso, foram consideradas duas hipóteses:

- **Uso de um indicador de tradução** → BLEU é um algoritmo popular que permite comparar a tradução esperada (criada por um humano) e a tradução efetuada, tendo um bom grau de correlação com o que é uma boa tradução. A aplicação do algoritmo resulta numa pontuação de 0 a 1 na qual quanto maior a pontuação, mais a tradução se assemelha a uma tradução por um profissional.

No entanto, tem limitações como não conseguir perceber aspetos como fluência, escolha de termos e sinónimos (apenas compara com a tradução esperada).

- **Avaliar manualmente** → Traduzir usando os vários serviços e escolher a melhor tradução ou classificar como uma má tradução. Dá-nos uma percentagem na qual um serviço foi preferido em relação a outro. Podemos ainda analisar quais as áreas na quais um serviço foi superior ao outro.

Quanto ao uso de um indicador, dado o domínio dos dados (textos clínicos), fazer uma tradução de referência iria envolver o conhecimento de termos técnicos, tanto em português como em inglês. Além disso, o tempo envolvido para fazer traduções de referência seria elevado.

Assim, o método usado na análise baseia-se numa **avaliação manual das traduções** para perceber se ambos os serviços obtiveram uma tradução satisfatória, se um serviço foi superior ou se nenhum alcançou um bom resultado.

Dados e ambiente de testagem

Visto que este processo de tradução será aplicado aos textos do campo “subjetivo” do SOAP faz sentido que os testes fossem efetuados com exemplos reais deste campo. Para isso, a empresa fez *queries* às suas bases de dados e foram extraídos cerca de 400 mil exemplos tais como os que são apresentados na Figura 18.

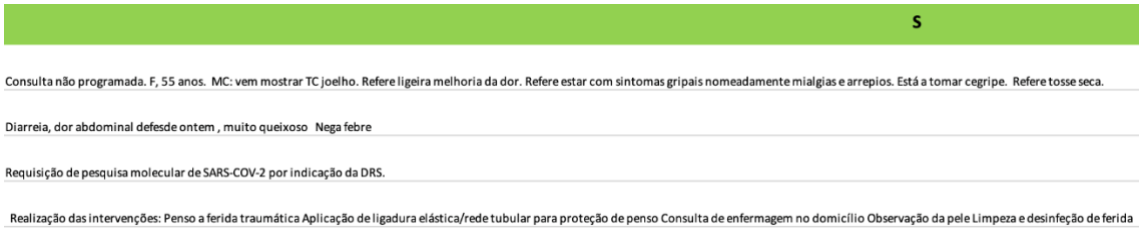


Figura 18 - Exemplo de textos do campo "Subjetivo"

Com a finalidade de comparar estes serviços de tradução foi elaborado código Python num Jupyter Notebook, de modo a importar os exemplos, executar os serviços e selecionar qual das traduções foi favorita (Figura 19). De notar que, para tentar manter a imparcialidade, a ordem na qual foram apresentadas as traduções de cada um dos serviços foi aleatória.

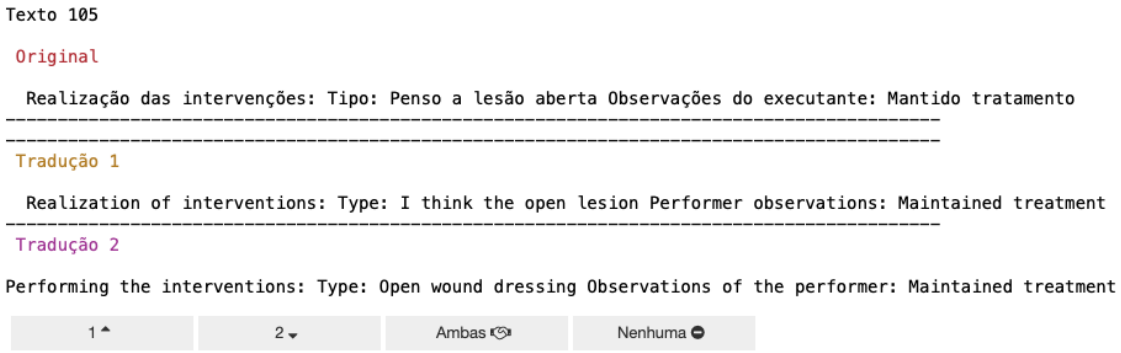


Figura 19 - Script para classificar traduções

Para tentar obter uma generalização de qual dos serviços é melhor, foi feita uma amostragem tomando um grau de confiança de 95% e uma margem de erro de 5%. O tamanho da amostra a considerar foi então calculado usando a fórmula: $\frac{z^2 * p(1-p)}{e^2}$. Obteve-se 384, que foi o número de excertos aleatoriamente retirados do *dataset*.

Resultados

Após execução do processamento e classificação das traduções produzidas pelos serviços, tendo em conta a amostra de excertos selecionada, apresenta-se a distribuição de percentagens relativas à escolha feita por meio da Figura 20.

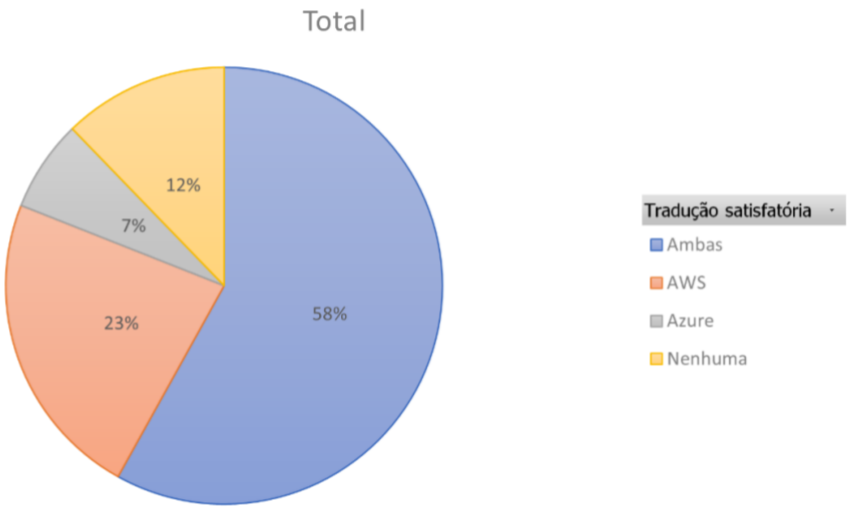


Figura 20 - Distribuição das opções escolhidas como uma tradução satisfatória

Analisando este gráfico (Figura 20) é percetível que os serviços de tradução têm uma tradução semelhante e satisfatória em 58% dos casos. No entanto, o serviço da AWS destaca-se por ter tido uma tradução preferida em 23% dos casos, enquanto o Azure apenas foi favorito em 7%. Uma análise mais profunda permitiu perceber as razões pelas quais um serviço foi escolhido em relação ao outro (Figura 21).

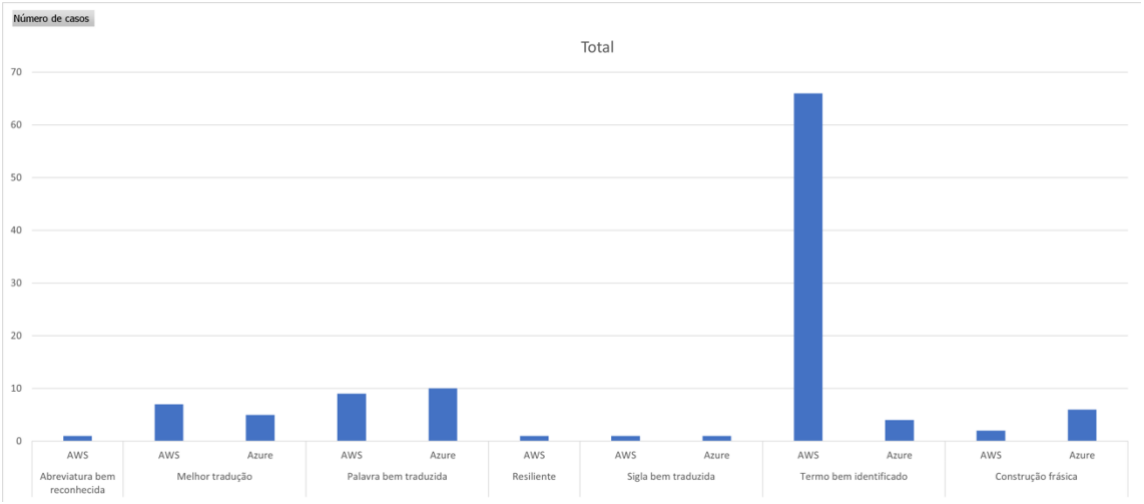


Figura 21 - Razões para a preferência de uma tradução

Consoante estas razões destaca-se a capacidade do serviço da AWS em identificar termos, sejam eles doenças ou intervenções (lista completa no Apêndice C).

De entre estes, destaca-se o termo “penso” usado, por exemplo, no contexto “penso a úlcera” ou “penso a ferida” é um dos termos frequentes que apenas foi identificado pelo serviço da AWS. Ao passo que o serviço do Azure traduziu muitas vezes este termo como se fosse o presente do verbo “pensar”.

Ainda explorando os casos em que nenhuma das traduções foi satisfatória, percebemos que as razões são variadas: muitas vezes têm que ver com o texto a ser traduzido que tem algumas inconsistências (tais como abreviaturas ou erros ortográficos) e, por isso, resulta numa tradução insatisfatória por parte do serviço; o não reconhecimento de alguns termos por parte de ambos os serviços; ou ainda uma tradução errada de algumas palavras no contexto em que aparecem.

Em suma, resultado desta comparação, foi escolhido o provedor AWS para este serviço de tradução, dado que produziu um maior número de traduções satisfatórias devido à sua superior capacidade de reconhecer termos presentes no vocabulário dos textos clínicos.

6.4 Modelo da arquitetura

Tendo em conta a secção anterior, ao apontar a Amazon como o provedor dos serviços *cloud*, pode ser agora representado um exemplo da arquitetura da solução. Para isto será usado o diagrama de componentes C4 (Figura 22).

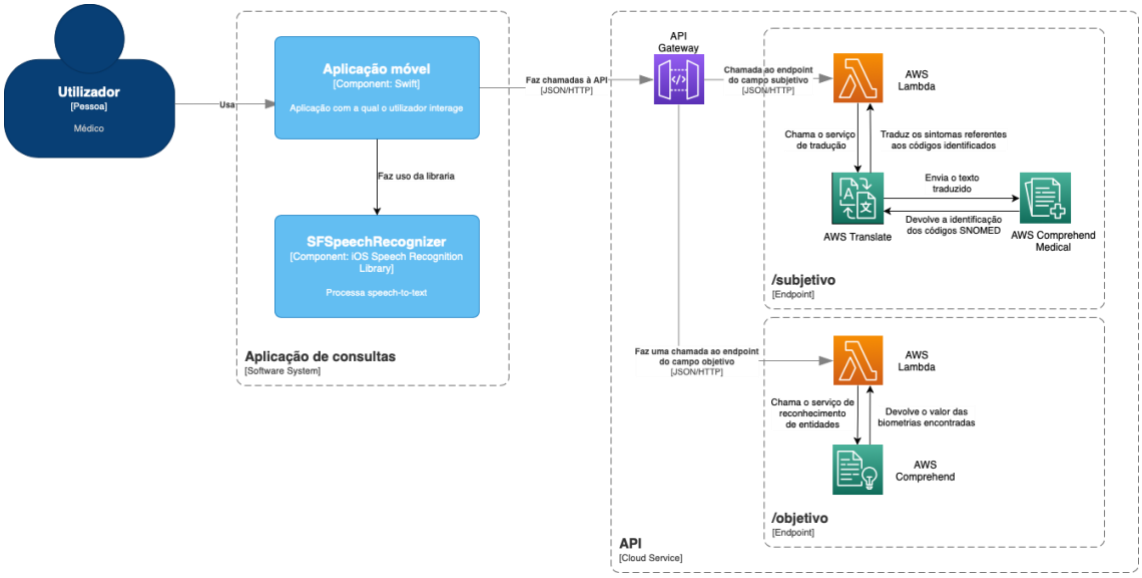


Figura 22 - Diagrama de componentes

Este diagrama foi elaborado antes de se ter iniciado o desenvolvimento, tendo sido alterado resultado da mudança de provedores. É reformulado na Secção 9.1.

6.5 App móvel - padrões arquiteturais

Também seguindo uma visão de arquitetura, ao desenvolver a aplicação móvel, podem ser seguidos vários modelos que dividem de forma diferente as camadas de apresentação, lógica e dados. Serão apresentadas e comparadas as diferenças entre "os 3 modelos mais comuns" (Galuh, 2022 [22]), para no final fundamentar a escolha de um deles.

MVC

A sigla MVC significa *Model View Controller* e divide em 3 componentes as várias camadas. É um modelo clássico tendo sido formulado na década de 70, com o objetivo de separar código e tornar uma aplicação mais modular [48].

A componente *Model* trata dos dados, é nela que são acedidas as bases de dados e recebidos pedidos de escrita por parte do *Controller*. Este último tem a função de processamento de toda a lógica e eventos resultados das ações do utilizador na camada *View* atualizando-a conforme. A camada de apresentação é então a camada *View*, que tem todos os componentes gráficos que compõem a UI e pode obter os dados diretamente do *Model*.

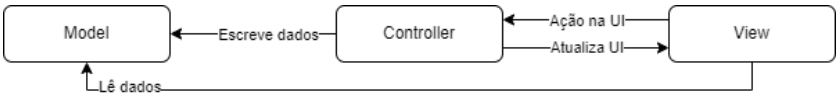


Figura 23 - Esquema do padrão MVC

MVP

É um modelo que evolui do MVC com o objetivo de evitar que o *Model* comunique diretamente com a *View*. Para isso, o componente *Controller* é substituído por um *Presenter*, que agora vai buscar dados ao *Model* e também é este que os envia para a camada de apresentação. O *Presenter* tem à mesma uma referência para a *View* devendo esta relação ser um para um [7].

Figura 24 - Esquema do padrão MVP

Isto permite o total desacoplamento da camada de apresentação o que permite uma maior reutilização de código.

MVVM

O padrão MVVM divide as camadas em *Model*, *View* e *ViewModel*. A grande diferença é o componente *ViewModel* que é responsável não só por ler e escrever dados do *Model*, mas como receber atualizações de dados (além do processamento da lógica). Também a *View* difere na medida em que é esta que referencia o *ViewModel*, pede a execução de funções através de interfaces e observa as atualizações dos dados no *ViewModel*, atualizando a UI conforme.

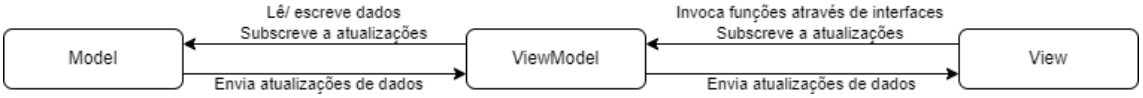


Figura 25 - Esquema do padrão MVVM

Assim, este modelo assenta no conceito de *data binding* que permite a sincronização de dados entre os vários componentes. Oferece vantagens na medida em que o *ViewModel* não tem referência da *View*, o que permite que haja a troca de *Views* sem que se troque o *ViewModel* ou a existência de várias *Views* por *ViewModel*.

Conclusões

Apesar de mais complexo, o padrão MVVM é aquele que permite a maior separação entre componentes e por isso contribui para uma solução mais modular, na qual o código está mais organizado. Também por estes motivos a testagem de uma aplicação deste tipo fica facilitada, visto que os componentes não dependem necessariamente uns dos outros.

É este padrão que foi adotado para o desenvolvimento da aplicação móvel, não só pelas vantagens referidas, mas também porque vai de encontro à prática da empresa.

Capítulo 7
Desenvolvimento da *backend*

Este capítulo irá abordar a construção da *backend* usando os serviços escolhidos, dedicando uma subsecção a cada um dos componentes.

7.1 Campo "Subjetivo"

O processamento do texto referente ao campo "Subjetivo" do SOAP foi feito com recurso ao serviço AWS Comprehend Medical. Dado que o mesmo não suporta o idioma português foi necessário traduzir os textos para inglês antes de os poder analisar, usando o serviço AWS Translate, como foi referido anteriormente (Secções 6.2 e 6.3).

Para este efeito, foi criada uma função AWS Lambda a qual irá receber os *inputs*, bem como coordenar os serviços, receber os resultados do processamento e devolvê-los.

Foi utilizada a linguagem Python na versão 3.8. Sendo que esta função recebe como *input* um pedido HTTP, em primeiro lugar foi definido o formato dos dados do pedido como se exemplifica na Figura 26.

Figura 26 - Exemplo de um pedido ao *endpoint* subjetivo

Através da função Lambda conseguimos ter acesso ao *body* do pedido e assim obter o texto correspondente ao valor associado à chave "text" do JSON, com o qual vamos passar a operar. Descrevendo em passos, as etapas seguintes serão:

1. Chamar o serviço de tradução usando o texto como *input*
2. Guardar o resultado da tradução
3. Chamar o serviço de reconhecimento de códigos SNOMED usando a tradução como *input*
4. Processar o JSON resultante da anterior chamada para obter a lista de sintomas identificados
5. Traduzir cada um dos sintomas

6. Adicionar ao JSON resultante da chamada 3 as traduções dos sintomas
7. Devolver em formato JSON a tradução do texto original, bem como os sintomas e respetivos códigos identificados

De notar que, em qualquer um destes passos pode ocorrer um erro, sendo mais provável naqueles que envolvem chamadas a outros serviços. Estes erros foram previstos e tratados adequadamente.

Foi elaborado um esquema do fluxo do processamento na notação BPMN presente no Apêndice D.

Ainda acrescentar que, visto que os serviços usados pertencem todos ao leque de serviços da AWS não é necessário usar quaisquer credenciais no código da função Lambda nem aquando dos pedidos a estes serviços. Apenas têm de ser configuradas as permissões da função Lambda para que possa usar métodos dos serviços visados.

Formato de saída

A resposta ao pedido HTTP contém no campo *body* um JSON onde são incluídas: a resposta do serviço AWS Comprehend Medical, bem como as traduções referentes aos sintomas identificados. Podemos sintetizar a estrutura da seguinte forma:


```
{
  "translation": "The user complains of fever and headaches.",
  "codes": [
    {
      "Id": 1,
      "Text": "fever",
      "Category": "MEDICAL_CONDITION",
      "Score": 0.9996768236160278,
      "SNOMEDCTConcepts": [
        {
          "Description": "Fever (finding)",
          "Code": "386661006",
          "Score": 0.9277660250663757,
          "Translation": "Febre (achado)"
        }
      ]
    },
    {
      "Id": 2,
      "Text": "headaches",
      "Category": "MEDICAL_CONDITION",
      "Score": 0.9997434020042419,
      "SNOMEDCTConcepts": [
```

```
            {
                "Description": "Headache (finding)",
                "Code": "25064002",
                "Score": 0.7656606435775757,
                "Translation": "Cefaleias (achado)"
            }
        ]
    }
  ]
}
```


Otimizações

Dado que cada chamada ao serviço de tradução implica um custo, optou-se por alterar um pouco a abordagem de modo a poder diminuir o número de chamadas a este serviço.

Foi então introduzida uma base de dados para que se possa ir buscar a tradução de um sintoma, caso ela já tenha sido adicionada, e desta forma poupar chamadas ao serviço de tradução.

Esta solução tem desvantagens do ponto de vista em que é sempre feita uma chamada à base de dados (caso esta guarde a tradução ou não) e tem o acrescentado armazenamento das traduções. No entanto, ao ser bastante mais barata por chamada, esta solução permite reduzir custos a médio e longo prazo.

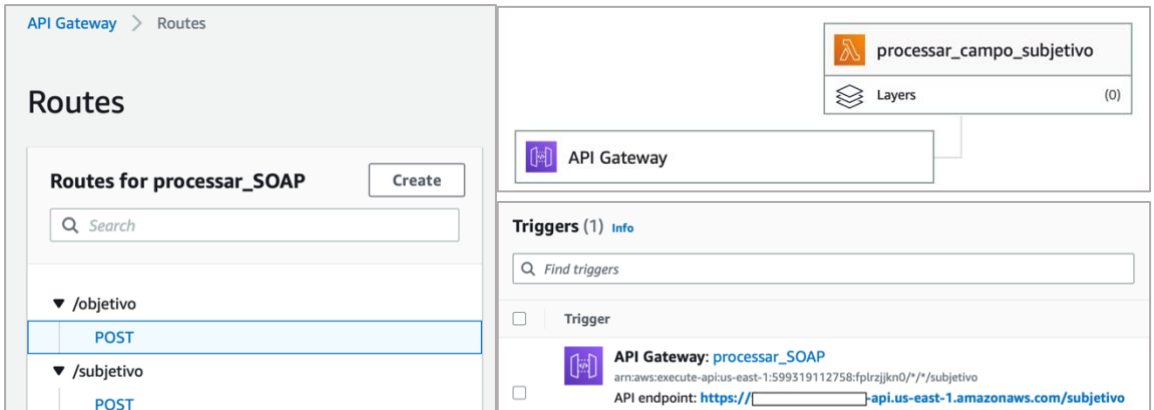
O serviço de base de dados utilizado foi o AWS DynamoDB e durante os testes efetuados não foi atingido o limite do plano gratuito que oferece 25GB e 200 milhões de pedidos por mês.

O esquema de processamento foi reformulado para refletir estas alterações e está presente no Apêndice D.

7.2 Disponibilização dos *endpoints*

A lógica do processamento é levada a cabo pelas funções Lambda, não obstante, é necessário que tenhamos forma de as invocar.

Foi arquitetado usar o serviço AWS API Gateway de modo a disponibilizar uma API REST. Sendo também este um serviço do leque da AWS não é necessário código para encaminhar pedidos para a lambda, apenas associar os dois serviços. Na Figura 27 vemos a configuração dos *endpoints*, bem como a associação com a Lambda na consola da AWS.



7.3 Autenticação

Após criadas as funções lambda e disponibilizado o *endpoint*, o acesso fica disponível publicamente. Assim, para atender ao requisito de segurança (Secção 4.5) é controlado o acesso à API usando o AWS Cognito.

Recorrendo a este serviço consegue-se integrar facilmente mecanismos de *login* e implementar a gestão de acesso aos métodos da API Gateway através de uma *user pool* na qual são registados os utilizadores autorizados.

Esta coleção de utilizadores foi configurada para apenas aceitar registos efetuados pelo administrador e a autenticação dos utilizadores é feita usando o mecanismo "USER_SRP_AUTH" que internamente faz uso do protocolo Secure Remote Password.

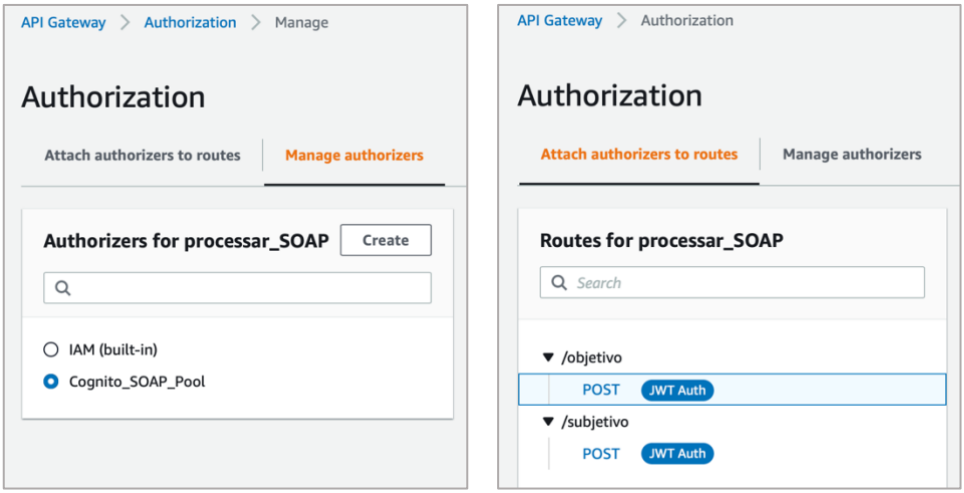


Figura 28 - Autorização da API Gateway usando o serviço Cognito

Após configurados os mecanismos de autorização aos *endpoints* na API Gateway (Figura 28) é possível utilizar o SDK do AWS Cognito para autenticar um utilizador com email e password. Depois de autenticado, é gerado um JWT o qual vai ser enviado no *header* dos pedidos à API e permite ao serviço API Gateway gerir a autorização.

7.4 Dados do campo “Objetivo”

Os dados do campo “Objetivo” do SOAP a identificar (Secção 1.3) constituem resultados de medições de várias biometrias (Apêndice B). Assim, estes registos surgem na generalidade como pares de “chave - valor”, nos moldes “biometria - resultado da medição” por exemplo:

> Altura 1 metro e 73. Peso 76 quilos. Tensão arterial sistólica 14, diastólica 7. Saturação 98 por cento. Medida da cintura 79 centímetros.

No entanto, apesar de existir um conjunto finito de biometrias a identificar, nos textos as chaves podem ter várias variações e introduzir informações adicionais.

- [Altura] – O doente mede 1 metro e 70;
- [Tensão arterial] – Pressão arterial 15/8;
- [Glicémia] – Glicémia em jejum 130;

entre outros.

Dados reais

De modo a ter uma ideia dos textos registados pelos médicos neste campo, a empresa fez *queries* às suas bases de dados e extraiu cerca de 140 mil exemplos (Figura 29).

Após uma breve análise percebemos que o formato dos registos se distancia do esperado, na medida em que:

- As chaves que identificam as biometrias são abreviadas;
- Os valores por vezes não têm unidades;
- São registadas informações pertencentes a outros campos do SOAP;

entre outros.

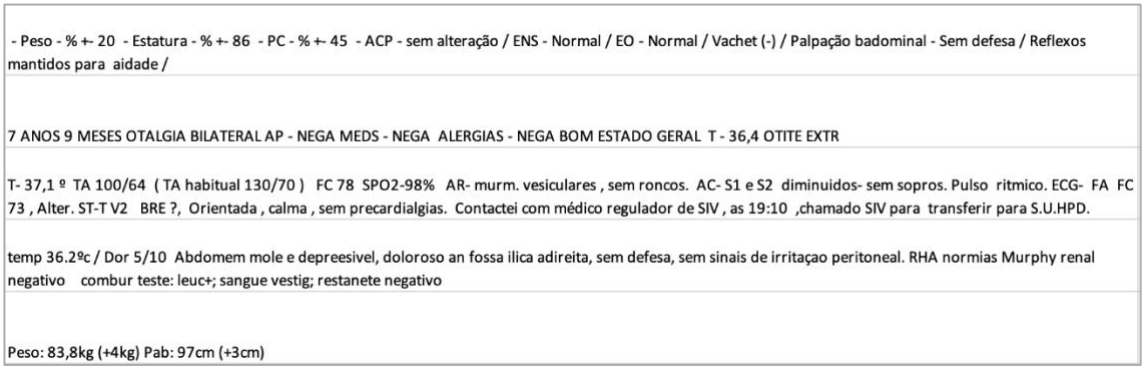


Figura 29 - Exemplo de registos do campo "Objetivo"

Em particular o uso de abreviaturas é algo que, apesar de presente nos textos, contrasta com o discurso oral no qual se opta por dizer as palavras por extenso.

Estratégia para a identificação de biometrias

Tendo em conta que se pretende ditar as biometrias para que estas sejam identificadas num par de “chave-valor”, foi necessário encontrar um mecanismo que permitisse fazer este processamento e ao mesmo tempo lidasse com alguma variação nos dados.

De acordo com os objetivos do estágio e tendo em conta as vantagens do uso de um modelo de processamento de linguagem natural, tomou-se a decisão de usar um serviço de reconhecimento de entidades personalizado, o AWS Comprehend. Este serviço é internamente um modelo de *machine learning*, o qual podemos treinar fornecendo-lhe exemplos e usar para fazer previsões num texto (Figura 30).

DILUTION

If you purchase units in this offering, you will experience dilution to the extent of the difference between the public offering price of the units (attributing no value to the warrants) and the net tangible book value per share of our common stock immediately after this offering.

Our net tangible book value as of June 30, 2017 was approximately $15.9 million, or $0.5589 per share of common stock. Net tangible book value per share is equal to our total tangible assets minus total liabilities, all divided by the number of shares of common stock outstanding as of June 30, 2017.

OFFERED_SHARES OFFERING_PRICE

After giving effect to the sale of 3,265,309 units at a price of $2.45 per unit, and after deducting our estimated placement agent fees and offering expenses payable by us, and attributing no value to the warrants, our as adjusted net tangible book value would have been approximately $23.3 million, or approximately $0.7357 per share of common stock, as of June 30, 2017. This represents an immediate increase in net tangible book value of approximately $0.1768 per share to existing stockholders and an immediate dilution of approximately $1.714 per share to new investors. The following table illustrates this calculation on a per share basis:

		OFFERING_PRICE
Public offering price per unit		$ 2.45
Net tangible book value per share as of June 30, 2017	$ 0.5589	
Increase per share attributable to this offering	$ 0.1768	
As adjusted net tangible book value per share as of June 30, 2017, after giving effect to this offering		$ 0.7357

Figura 30 - Exemplo do uso do AWS Comprehend para reconhecimento de entidades

Dados de treino

De modo a poder treinar um modelo de reconhecimento de entidades, é necessário fornecer exemplos de textos, bem como das entidades a identificar presentes nestes.

Face a esta necessidade, uma abordagem seria usar os exemplos reais retirados das bases de dados e identificar as biometrias. No entanto, como observado acima, estes dados na maioria não refletem o discurso oral.

Como alternativa e tendo em conta que os valores esperados são formatados como “chave - valor”, foram geradas frases contendo várias combinações e variações destas biometrias, como se apresenta na Figura 31.

Mede 1 metro e 42 Estatura 1 e 64 Altura 1,52m ---------- Peso 56kg Peso 117 quilogramas Peso 86 quilos ---------- Temperatura 34.7ºC Temperatura 40 e 3 Temperatura 37.2	Cintura 78 cm. Quadril 74 ponto 2. Dor 6. Glic 166 miligramas. Frequência cardíaca 142 batimentos por minuto. Dinamometria manual direita 32 N e 3. Perímetro braquial 39 ponto 1. Peso 143 quilos. Temp 34 e 2 Tensão 141-119 mmHg Mede 1 e 95 Perímetro torácico máximo 72,5 Massa gorda 38 Dinamometria escapular esquerda 38 quilogramas força Perímetro torácico mínimo 36.7 Dinamometria escapular direita 15 quilos Dinamometria manual esquerda 55 ponto 3 Newtons Freq resp 11 rpm Prega tricipital 45 e 3 Dinamometria manual direita 38 ponto 7 quilos Anca 79cm Oxigénio 73% Temp 42.4 Perímetro torácico mínimo 54 cm Frequência cardíaca 152bpm Peso 96 q uilos Dinamometria manual direita 51 e 1 N Dinamometria manual esquerda 43 quilos força Glic 217 mili gramas Dinamometria lombar 132 e 6 Newtons Estatura 1 metro e 28 centímetros Dinamometria escapular e squerda 39 quilos Dinamometria escapular direita 17 N Saturações 98 por cento Freq resp 22 Dinamometria lombar 215 kgf Prega tricipital 53 e 2 Dor 0 FC 158 bpm Cintura 84 centímetros Quadril 100 centímetros e 5 Perímetro torácico máximo 43 centímetros e 7 milímetros Temp 34.4 Perímetro torácico mínimo 74 cm Dinamometria escapular esquerda 28 e 3 kgf

Figura 31 - Exemplo dos textos gerados

Estas frases contêm arranjos aleatórios de biometrias, assim como cada biometria tem uma variação aleatória de uma forma como pode ser escrita, do seu valor e unidades. Para isto, foi feita uma lista de algumas das variações possíveis.

Esta abordagem permite-nos aproximar do discurso esperado e ao mesmo tempo gerar uma grande quantidade de exemplos distintos.

Contudo os dados retirados das bases de dados não foram descartados, mas usados em conjunto com estes exemplos servindo em parte como exemplos negativos.

Labeling

Após ter um conjunto de textos representativos daqueles sobre os quais o modelo vai operar, para o treinar é necessário identificar as entidades que estão presentes em cada um deles, bem como em que lugar do texto aparecem.

Este processo de identificação referido como "labeling" foi feito usando a ferramenta *open-source* – Doccano [37].

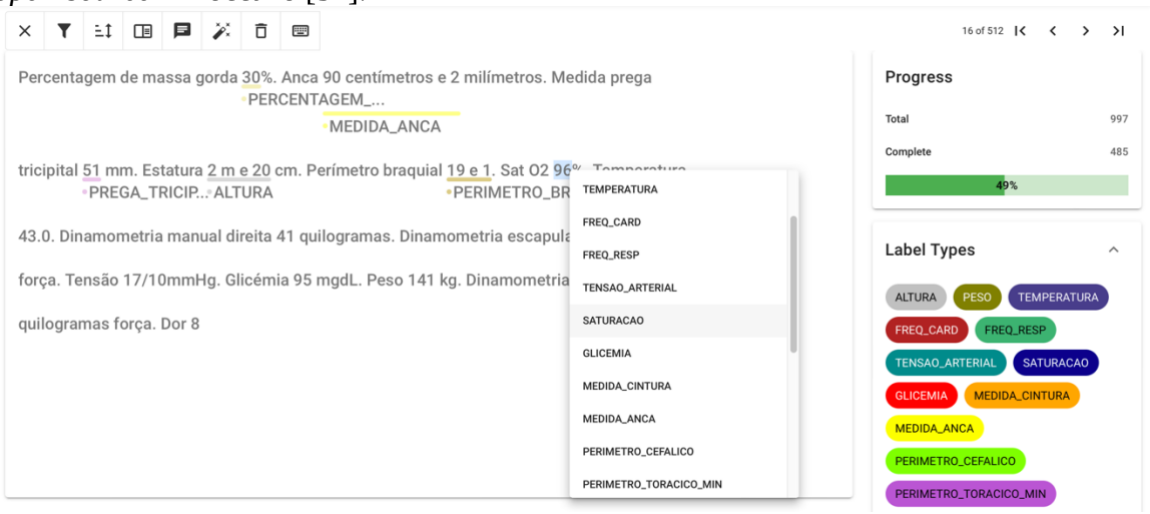


Figura 32 - Exemplo da ferramenta Doccano

Com o auxílio desta ferramenta é selecionado um excerto e é-lhe feito corresponder uma entidade (Figura 32). Este processo deve ser consistente entre os vários textos, na medida em que deve seguir uma mesma estratégia.

Nos vários textos foram identificadas as várias entidades pelos seus valores (até ao último algarismo), excluindo a unidade.

Cada um dos textos anotados é exportado pela ferramenta como um JSON contendo as anotações, bem como os carateres inicial e final onde na frase estas ocorrem (Figura 33). O conjunto é então agregado num ficheiro contendo um JSON por linha, formato JSONL.


```
{
    "id":64,
    "text":"Pulso 128bpm Peso 103 quilos Mede 194 centímetros Medida da anca 93,4 Dinamometria lombar 94 ponto 2 quilogramas força Dor 1",
    "Comments":[],
    "label":[
        [6,9,"FREQ_CARD"],
        [18,21,"PESO"],
        [34,37,"ALTURA"],
        [65,69,"MEDIDA_ANCA"],
        [90,100,"DINAMOMETRIA_LOMBAR"],
        [123,124,"DOR"]
    ]
}

{
    "id":363,
    "text":"Estatura 1 metro e 98 Perímetro torácico mínimo 44.6 Peso 39 Dinamometria lombar 80 kgf Medida da cintura 77 centímetros e 9 mm",
    "Comments":[],
    "label":[
        [9,21,"ALTURA"],
        [48,52,"PERIMETRO_TORACICO_MIN"],
        [58,60,"PESO"],
        [81,83,"DINAMOMETRIA_LOMBAR"],
        [106,124,"MEDIDA_CINTURA"]
    ]
}
```


Figura 33 - Exemplo da estrutura contendo as anotações

Esta estrutura é depois convertida para o formato especificado pelo serviço que efetua o reconhecimento de entidades de modo a ser importado.

7.5 AWS Comprehend

A análise feita na Secção 2.3 mostrou que existem várias alternativas a usar quando pretendemos efetuar o reconhecimento de entidades personalizadas. Todavia visto terem sido utilizados serviços do provedor AWS para o processamento do campo “subjetivo”, também foi projetado usar o mesmo provedor aquando do processamento do campo “objetivo”.

Nesta plataforma para treinar um modelo de reconhecimento de entidades personalizado apenas é necessário converter o ficheiro produzido pelo processo de *labeling* para o formato CSV especificado, fazer *upload* do ficheiro para o serviço de armazenamento AWS S3 e especificar a lista de *labels* existente (Figura 34).

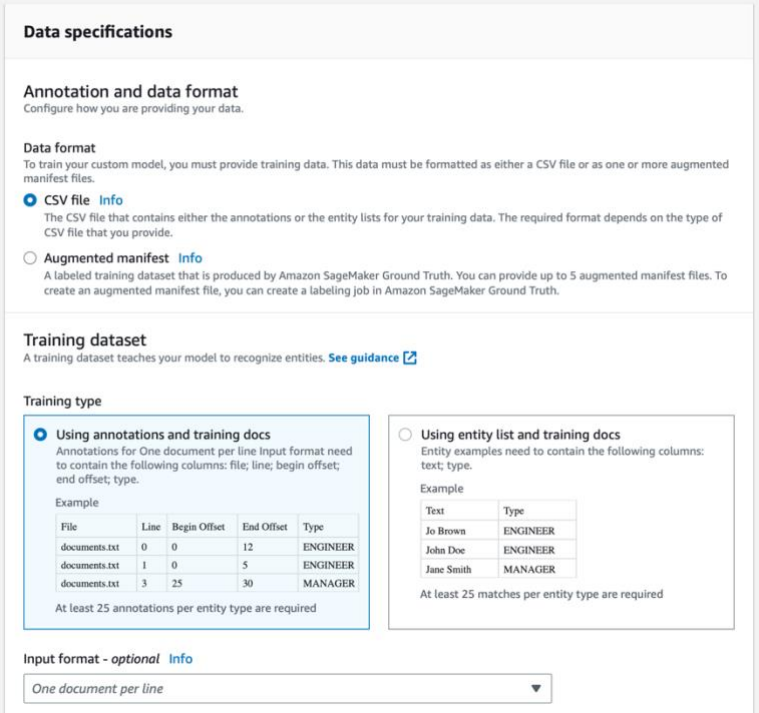


Figura 34 - Configuração dos dados de treino no AWS Comprehend

De seguida é possível submeter os documentos para treino (processo que demora algum tempo). Quando finalizado permite obter as estatísticas de *performance*, bem como possibilita que sejam feitas previsões usando o modelo.

Modelos e resultados

Foram efetuadas algumas iterações com diferentes dados para avaliar a resposta do serviço face à quantidade de exemplos fornecidos, bem como à sua natureza (gerados e reais).

O primeiro conjunto de dados fornecido foi composto por textos gerados, num volume de apenas 54 documentos (textos) dos quais 90% foram usados para treinar o modelo (correspondendo a 48 documentos) e 10% foram usados para avaliar a *performance* do mesmo.

Como resultado o modelo obteve um F1 *score* de 97,1% como se mostra na Figura 35.

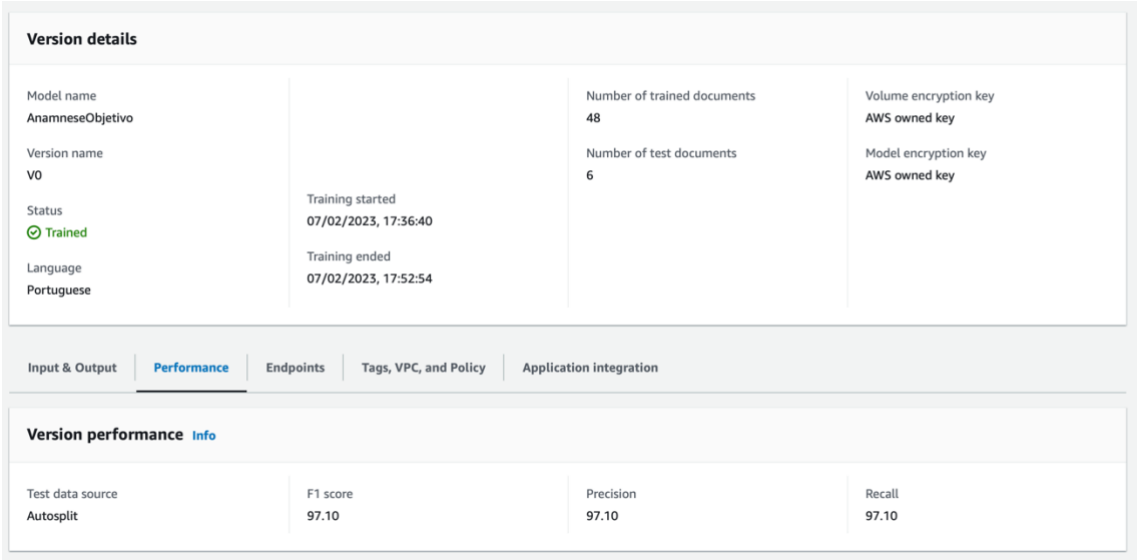


Figura 35 - Performance da primeira versão do modelo treinado usando o AWS Comprehend

Seguiram-se duas versões mantendo a percentagem de divisão entre documentos de treino e teste:

- a segunda totalizando 154 documentos, dos quais 100 eram textos reais e o restante gerado;
- a terceira contabiliza 670 documentos numa mistura de textos reais e gerados.

A cada nova versão existiu um incremento do F1 *score* do modelo, no entanto, este manteve-se cerca dos 97% (Figura 36).

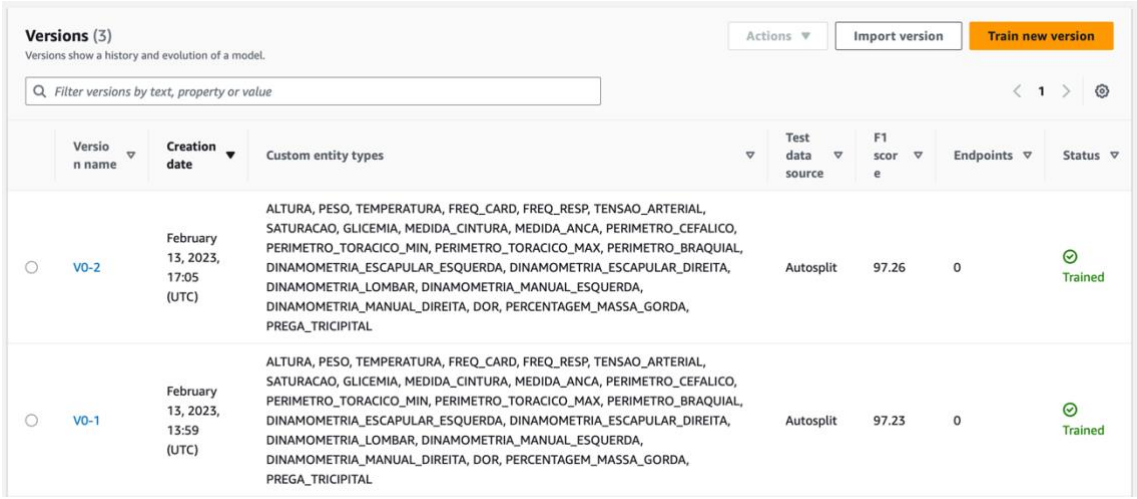


Figura 36 - Performance de duas versões do modelo treinado usando o AWS Comprehend

Este nível de *performance* é satisfatório e corrobora um dos motivos pelo qual foi escolhido um serviço de NLP na *cloud* – a sua boa prestação mesmo quando existe uma reduzida quantidade de dados.

Previsões

Depois de treinado, o modelo pode ser usado para fazer previsões de duas formas – de forma síncrona e assíncrona. Estas formas estão, não só relacionadas com o tempo de espera estimado para a resposta do serviço, mas também com o formato da resposta.

Usando o SDK da AWS para efetuar uma previsão assíncrona e de seguida obter o resultado, utilizam-se os seguintes passos:

1. O texto é enviado para processamento e é devolvido um número correspondente à tarefa iniciada neste passo;
2. Com o número da tarefa podemos fazer um pedido para verificar o seu estado (por exemplo: pendente, em processamento, concluída);
3. Após verificar o estado como concluída, o pedido inclui uma referência para um ficheiro comprimido, armazenado no serviço AWS S3 e contendo um JSONL com as entidades identificadas.

Esta forma de processamento adequa-se a um grande volume de dados nos quais pretendemos processar vários textos em simultâneo e não esperamos um resultado imediato, conquanto no caso do campo "objetivo" queremos processar apenas um texto no mínimo tempo necessário.

Em contraste, o processamento síncrono ou "em tempo real" permite fazer previsões sobre apenas um texto, sendo que este processamento é iniciado imediatamente e esperamos, na mesma chamada, uma resposta no formato JSON.

Processamento em tempo real

Tendo em conta os objetivos da aplicação e em particular do campo "objetivo", como sendo identificar e listar as biometrias a partir de um ditado feito pelo médico durante a fase de anamnese na consulta, concluiu-se que o processamento em tempo real seria a forma de processamento a usar. No uso da aplicação, o médico deve ser capaz de ditar e de seguida autorizar o processamento do texto, recebendo o mais breve possível os resultados. A possibilidade de corrigir ou descartar valores imediatamente após o processamento foi um dos fatores que pesou nesta decisão.

Esta forma de processamento exige um maior esforço na infraestrutura do provedor de serviços *cloud* visto que têm que existir recursos alocados a este modelo, prontos a iniciar uma previsão e rapidamente devolver uma resposta. Logo, para poder executar processamentos desta natureza, é necessário criar um *endpoint* do modelo e configurá-lo com uma taxa de processamento denominada "*inference units*" entre 1 e 10 unidades discretas (sendo que 1 *inference unit* corresponde a um processamento de 100 carateres de texto por segundo). Após este passo existe um tempo de espera para que sejam alocados os recursos necessários à disponibilização do *endpoint* e só a partir deste ponto podemos invocá-lo para efetuar as previsões em tempo real.

7.6 Problemas no uso do AWS Comprehend para processamento em tempo real

Na secção anterior (Secção 7.5) foi apresentado o processo de implementação do AWS Comprehend com a finalidade de ser usado na identificação das biometrias do campo "objetivo" do SOAP, bem como o raciocínio por detrás do uso da forma de processamento "em tempo real".

Este provedor foi escolhido à partida porque toda a restante *backend* foi montada usando esta mesma opção, além de que oferecia serviços comparáveis aos restantes. Também os preços de treino e inferência eram equivalentes (Secção 2.3).

O uso do processamento em tempo real implicou a criação de um *endpoint* e esta operação acarretou custos não só fixos (foi cobrada a disponibilização de um *endpoint*), como também custos variáveis associados ao número de *inference units* utilizados. No entanto, este provedor cobrou pelas *inference units* mesmo que não tivesse sido efetuada nenhuma previsão – 0,005USD por segundo por *inference unit*.

Posto isto, o custo do uso deste serviço fornecido por este provedor passou a ser incomportável e, conforme previsto no plano de contingência (Capítulo 5), foi necessário encontrar uma alternativa.

Alternativa ao AWS Comprehend

Face ao problema de custos verificado existem algumas alternativas que podem ser usadas para o resolver, sendo que das menos impactantes será a alteração do provedor dos serviços de reconhecimento de entidades.

Como foi analisado na Secção 2.3 destacam-se as opções da Google, IBM e Microsoft. Ainda assim, novamente numa perspetiva económica, verificamos que: a solução da IBM apenas é benéfica no caso do plano gratuito visto que ultrapassar este plano incorre em custos mais elevados que o AWS Comprehend; a solução da Google é semelhante à da AWS na medida em que o uso de um *endpoint* tem um valor de 0,05USD por hora (independentemente da taxa de utilização); já a alternativa da Microsoft oferece um preço muito inferior às restantes, com a disponibilização de um *endpoint* a custar 0,50USD por mês.

Deste modo, tomou-se a decisão de usar o serviço Microsoft Azure Cognitive Service for Language na vertente de *custom named entity recognition* ou *custom* NER. Sendo esta a opção mais económica e que menos impacto causa, visto que a empresa já usava outros serviços deste provedor.

7.7 Microsoft Azure custom NER

O uso do serviço de reconhecimento de entidades personalizadas da Microsoft implicou o treino de um novo modelo nesta plataforma. Este processo foi semelhante ao do treino no serviço da AWS, envolvendo apenas a conversão do ficheiro resultante do processo de *labeling*, importação dos textos e anotações para um *container* no serviço de armazenamento e início do treino de um modelo.

Primeiro modelo e resultados

No primeiro modelo treinado usando este serviço foram usados os mesmos dados (textos e anotações) que o último modelo treinado usando o serviço da AWS, contando com 670 documentos. No entanto, neste serviço foi usada uma divisão entre documentos de treino de 80% (contabilizando 429 documentos) e teste de 20% (ou 107 documentos), de notar que esta divisão foi efetuada automaticamente pelo serviço e os documentos sem anotações não foram usados para teste.

Após o processo de treino, o modelo obteve um F1 *score* de 81,25% como se apresenta na Figura 37. Este resultado é inferior ao obtido pelos modelos treinados usando a AWS.

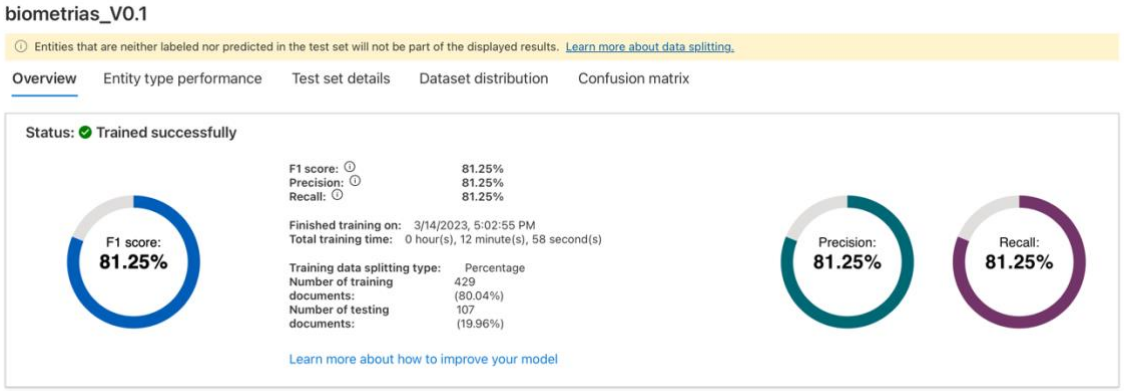


Figura 37 - Performance do primeiro modelo treinado usando o serviço da Microsoft

Uma análise à matriz de confusão gerada pelo serviço (Figura 38) e que nos indica a *performance* em relação a cada entidade, fundamenta este resultado.

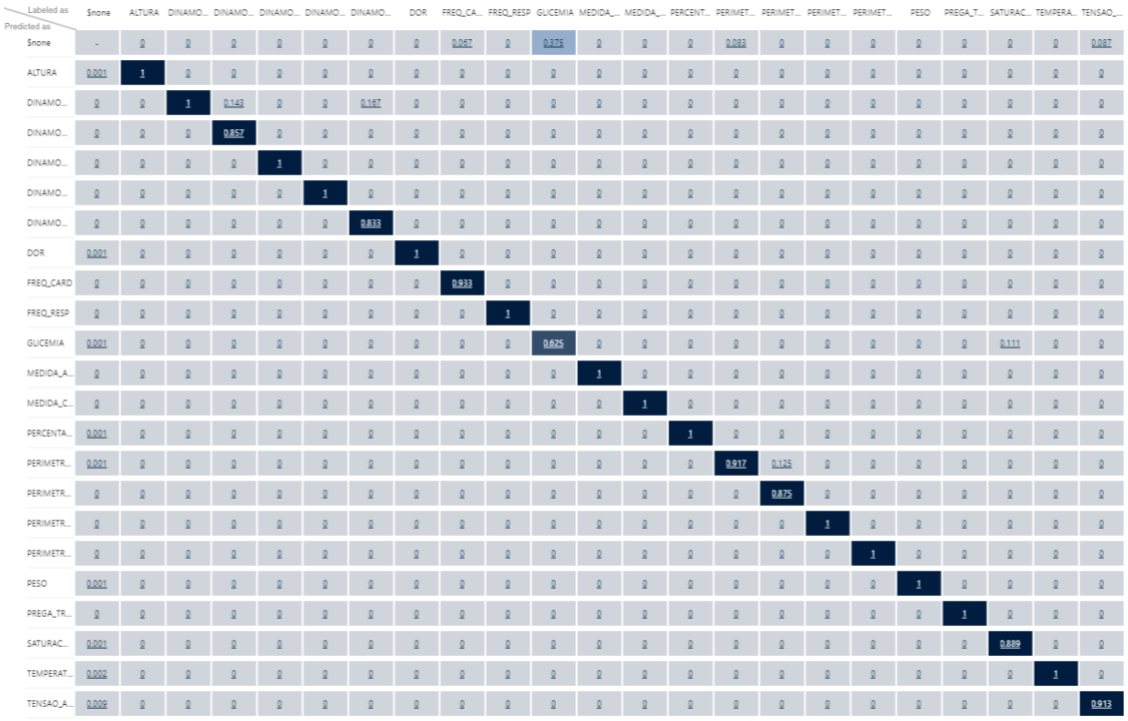


Figura 38 - Matriz de confusão do primeiro modelo treinado pelo serviço da Microsoft

Na matriz, os principais transtornos de valores ocorrem: no caso da entidade “glicémia”, noutras entidades como as que denotam uma “dinamometria” ou um “perímetro”, mas também nas várias ocorrências em que foi detetada uma entidade num valor não anotado, ou seja, casos de falsos positivos.

Numa observação dos casos de erro é percetível que as biometrias identificadas como glicémia podem ser identificadas por várias chaves: glicémia, glucose e glicose, e o número de casos de treino pode não ser suficiente para o modelo conseguir prever todos estes casos. O mesmo acontece no casos das biometrias de dinanometria e perímetro, que sendo dois grupos de entidades semelhantes (identificar “dinamometria escapular” ou “dinamometria lombar”) podem requerem mais exemplos de treino para que o modelo consiga distinguir corretamente.

Já nos casos de falsos positivos, muitas vezes o modelo detetou bem o excerto, apenas a fronteira entre a anotação e o restante texto ficou mal definida. Por exemplo, se um número anteceder um ponto final, ao invés de apenas o número ser identificado, também o ponto final o é (Figura 39). Especulando, isto acontece porque o *tokenizer* do modelo reconhece o ponto final como sendo o limitador da parte decimal do número e inclui-o no mesmo *token*. O mesmo acontece quando as unidades seguem um número sem haver um espaço entre eles, todo o texto pertence ao mesmo *token*.

Document name	Labeled text	Labeled as	Predicted as	Result type
document_104.txt	39kgf	Unlabeled in test set	DINAMOMETRIA_MANUAL_E...	False Positive
document_104.txt	24 ponto 1.	Unlabeled in test set	DINAMOMETRIA_ESCAPULA...	False Positive
document_104.txt	103 ponto 4.	Unlabeled in test set	MEDIDA_ANCA	False Positive
document_104.txt	39	DINAMOMETRIA_MANUAL_E...	No prediction	False Negative
document_104.txt	24 ponto 1	DINAMOMETRIA_ESCAPULA...	No prediction	False Negative
document_104.txt	103 ponto 4	MEDIDA_ANCA	No prediction	False Negative

Figura 39 - Exemplo de casos em que o modelo falhou a prever

Nos exemplos da figura acima verifica-se que os casos falsos positivos e falsos negativos se complementam, sendo que os excertos que o modelo identifica têm a entidade correta e contêm o valor pretendido, apenas contam com carateres adicionais.

Melhorias ao modelo de reconhecimento

Devido ao *tokenizer* do modelo não ter o comportamento esperado em relação à estratégia de *labeling*, a *performance* do modelo não atingiu um patamar ótimo (visto que as entidades identificadas não correspondem com as previstas).

Com o objetivo de melhorar a *performance* e uniformizar os textos usados para treinar o modelo foi usado um *script* de pré-processamento que separa os números das letras, retira alguma pontuação quando no limiar de um número e ajusta as *labels* de forma correspondente. Este processamento pode ser aplicado também aos textos produzidos através de ditado.

O serviço foi treinado com um número de textos aumentado para cerca de 1270 com pré-processamento aplicado, obtendo um F1 *score* de 97,91% (Figura 40).

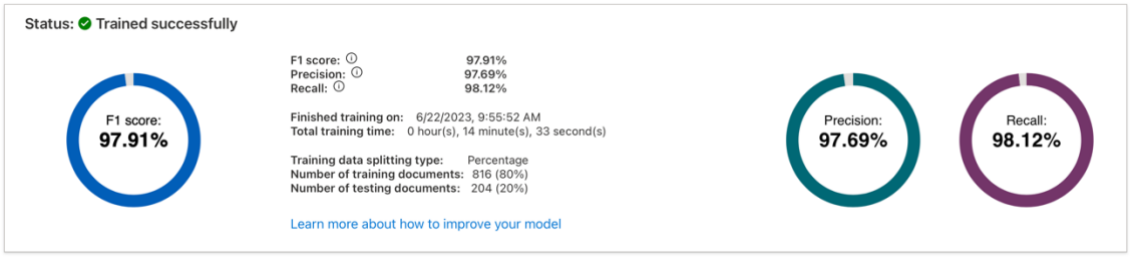


Figura 40 - Performance do modelo treinado usando textos pré-processados

Este resultado é uma melhoria em relação ao anterior modelo e encontra-se em linha com a *performance* obtida usando o serviço da AWS.

Disponibilização

De forma diferente do serviço da AWS, o serviço da Microsoft necessita que sejam alocados recursos ao modelo antes de efetuar qualquer previsão. Assim, é necessário fazer *deploy* ao modelo e depois deste processo o serviço fornece um *endpoint* o qual pode ser usado para realizar as previsões.

7.8 Função Lambda para processamento do campo "Objetivo"

Finalizada a definição e implementação de uma estratégia para reconhecer biometrias, segue-se a disponibilização da interface REST através da AWS API Gateway e coordenação do processamento usando uma função AWS Lambda.

À semelhança do anterior campo do SOAP, também o *endpoint* do campo "Objetivo" recebe um pedido HTTP com um JSON no *body* contendo um campo "text" com o texto a processar, como está no pedido de exemplo na Figura 41.

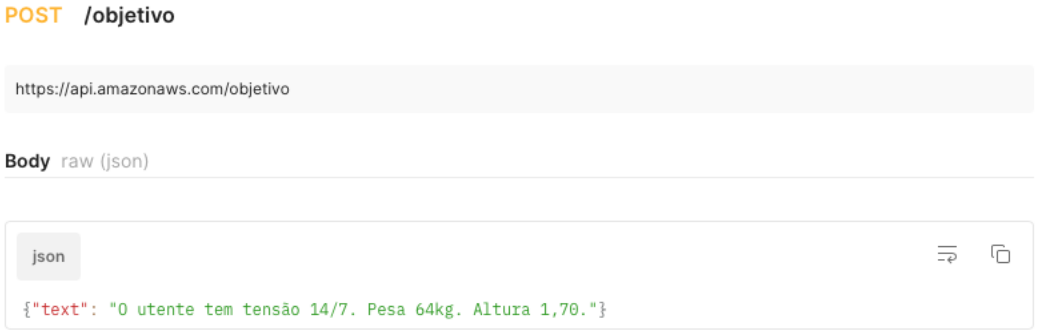


Figura 41 - Exemplo de um pedido ao *endpoint* objetivo

Depois de receber o pedido a função Lambda procede a enviar o texto para o serviço de reconhecimento de entidades. A mudança deste serviço para a Microsoft implicou que fossem importados para a Lambda o SDK necessário para fazer pedidos ao serviço, bem como as credenciais de acesso ao mesmo. Para isso, foi criada uma *layer* na função que

aloja o código das bibliotecas do Azure Cognitive Services e criadas variáveis de ambiente com a definição do *endpoint* do modelo e chave de acesso (Figura 42).

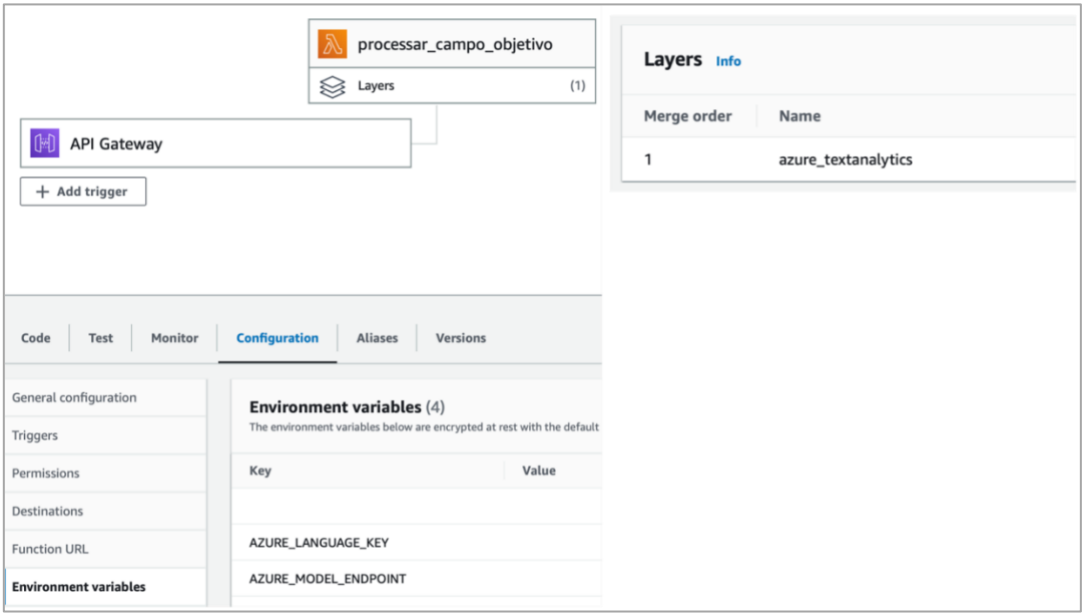


Figura 42 - Configuração da função Lambda do campo objetivo

A chamada ao serviço de reconhecimento de entidades por sua vez retorna um JSON com as biometrias identificadas, bem como o excerto correspondente.


```
{
   "kind": "EntityRecognitionResults",
   "results": {
      "documents": [
       {
         "entities": [
           {
             "text": "14/7",
             "category": "TENSAO_ARTERIAL",
             "offset": 20,
             "length": 4,
             "confidenceScore": 1,
           },
           {
             "text": "64",
             "category": "PESO",
             "offset": 31,
             "length": 2,
             "confidenceScore": 1,
           },
           {
```

```
            "text": "1,70",
            "category": "ALTURA",
            "offset": 44,
            "length": 4,
            "confidenceScore": 0.99,
          }
        ],
        "id": "1"
      }
    ],
    "errors": []
  }
}
```


Pós-processamento

Depois de obter os resultados das identificações é necessário efetuar um pós-processamento para perceber se os valores estão conforme o esperado. Por exemplo, no caso dos valores que têm os seus algarismos separados por uma forma textual, estes devem ser agregados num número decimal: o valor "37 ponto 3" deve ser convertido para "37,3".

Formato de saída

Finalmente são devolvidos os valores no *body* da resposta como uma lista contendo o tipo da biometria e respetivo valor.


```
[
 {
  "tipo": "TENSAO_ARTERIAL",
  "valor": {
     "sistolica": 14,
     "diastolica": 7,
  }
 },
 {
  "tipo": "PESO",
  "valor": 64,
 },
 {
  "tipo": "ALTURA",
  "valor": 1.7,
 }
```

```
]
```

Excecionalmente no valor das biometrias do tipo “tensão arterial” é feita a distinção entre o valor da tensão sistólica e diastólica.

7.9 Testes da API

De modo a comprovar o funcionamento da componente de processamento, foi desenvolvida uma bateria de testes usando o *software* Postman (Figura 43) que efetua

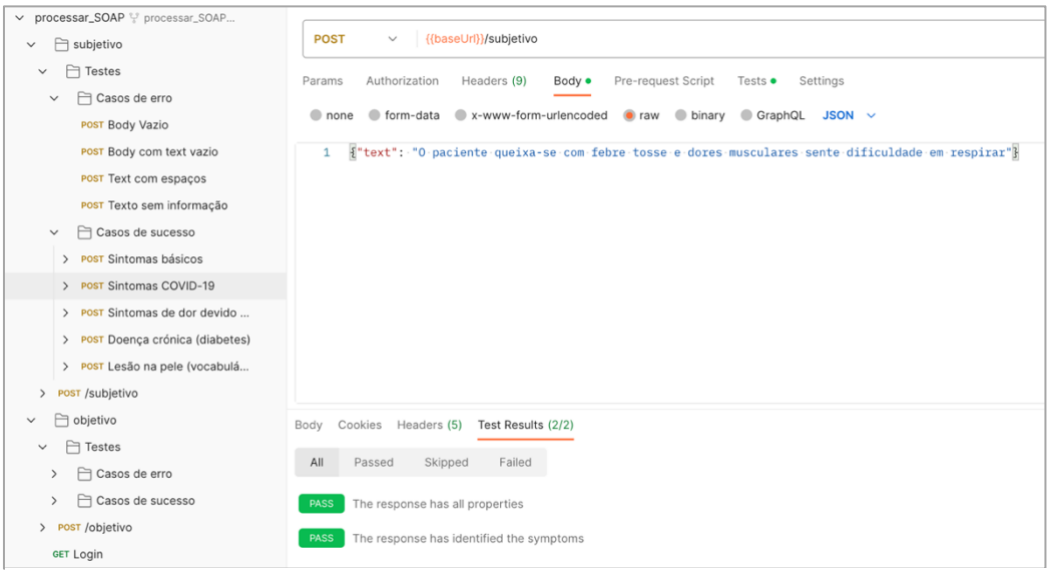


Figura 43 - Exemplo de um teste feito usando o *software* Postman

pedidos à API e compara a resposta obtida com uma resposta esperada. Esta prática de testes é semelhante à usada pela empresa noutros projetos e, por isso, foi a escolhida.

A vantagem do uso de uma ferramenta deste tipo é a automatização do processo de teste, podendo correr os testes automaticamente e ter informação de quais deles falharam. No entanto, o *software* não permite guardar estes resultados para consulta no futuro.

Ainda assim, uma versão do Postman exclusivamente para a linha de comandos “Newman” permite correr e testar todos os pedidos de uma coleção, mas também encaminhar o *output* obtido para outros módulos (referidos na documentação como *reporters*) usando a *flag* “-r”. Foi então usada esta opção em conjunto com uma biblioteca

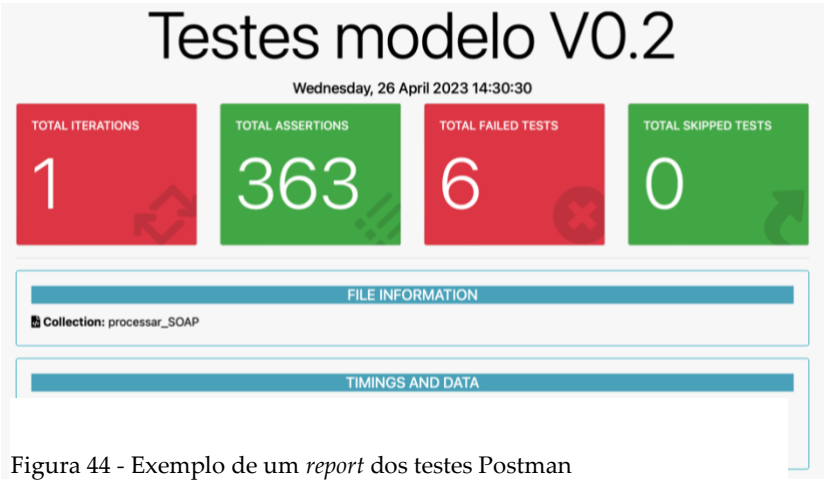


Figura 44 - Exemplo de um *report* dos testes Postman

open-source de *reporting* “newman-reporter-htmlextra” [14] e gerados relatórios com a informação dos pedidos bem sucedidos e falhados.

Capítulo 8
Desenvolvimento da aplicação

O presente capítulo apresenta o processo de desenvolvimento da aplicação e integração dos vários componentes da solução.

8.1 Mockups

De modo a objetivar o *design* esperado para a aplicação bem como os elementos que deveria incluir, a empresa desenvolveu alguns *mockups* apresentados nas figuras seguintes.

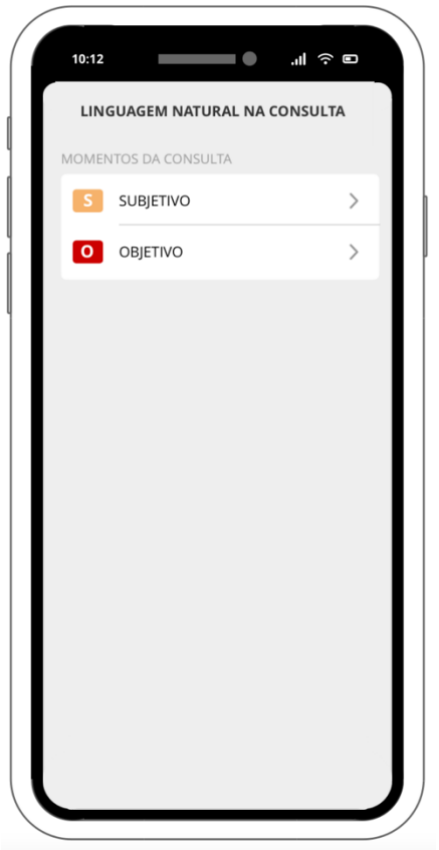


Figura 45 - Mockup do ecrã inicial

A Figura 45 representa o ecrã inicial da aplicação (após o *login*) no qual o médico tem a possibilidade de escolher o campo do SOAP que pretende preencher, cada um relativo a um momento da consulta.

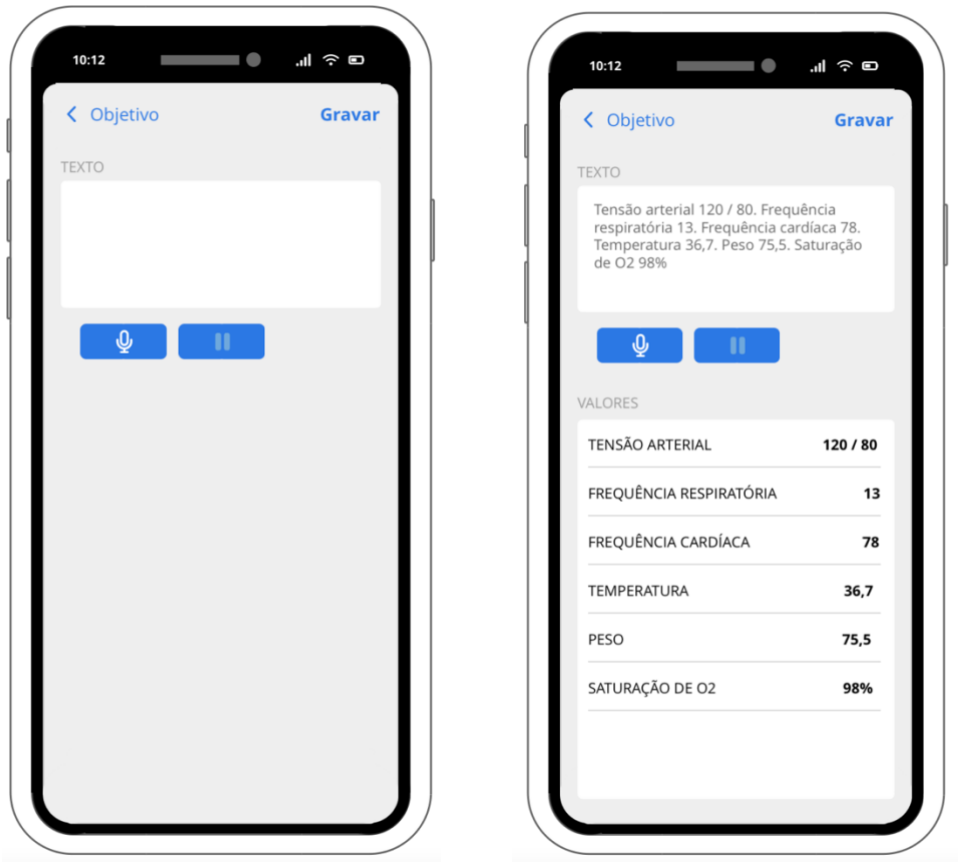


Figura 46 - Mockups do ecrã referente ao campo objetivo

A Figura 46 serve para representar o ecrã referente ao campo objetivo em dois estados diferentes:

- Antes do preenchimento, contando com uma caixa de texto, bem como um botão para iniciar o ditado e outro para o terminar;
- Depois de preenchido e processado, onde existirá uma lista com as biometrias e respetivos valores encontrados.

Figura 47 - Mockup do ecrã referente ao campo subjetivo

À semelhança do campo objetivo, também o campo subjetivo terá os mesmos elementos. No entanto, a lista será preenchida pelos sintomas e códigos identificados (Figura 47).

Os *mockups* apresentados serviram para guiar o desenvolvimento e demonstrar uma possível concretização dos objetivos. Ainda assim, existiu liberdade para fazer alterações mediante justificação.

8.2 Principais classes e componentes da aplicação

Tendo em conta os *mockups* começou-se a construir a aplicação usando a linguagem Swift e modelo MVVM, tal como apresentado na Secção 6.5.

Na presente secção serão abordadas as classes e componentes usados que permitiram a integração com a *backend* e restantes funcionalidades.

Autenticação

Como ponto de entrada na aplicação existe um ecrã de *login* onde é feita a autenticação do utilizador recorrendo ao serviço AWS Cognito, de acordo com a arquitetura da *backend.*

Para poder fazer chamadas a este serviço é necessário incluir no projeto as bibliotecas da AWS Amplify e configurar a ligação à *user pool* criada anteriormente (Secção 7.3).

Figura 48 - Configuração do AWS Amplify

Após configurada como se mostra na Figura 48, passa a ser possível utilizar os métodos disponibilizados pela biblioteca.

A lógica da autenticação está presente na classe "AuthService", incluindo os métodos de *sign-in*, *sign-out* e de obtenção/atualização do *token* JWT necessário para autenticar as chamadas REST feitas aos *endpoints* da *backend*.

API REST

De modo a efetuar o processamento dos campos subjetivo e objetivo são feitas chamadas autenticadas aos *endpoints* disponibilizados pela AWS API Gateway. Para este efeito foi criada uma classe para processar o pedido e resposta.

Os pedidos são feitos usando o método POST e incluem o texto a processar na chave "text" de um JSON presente no *body*. A resposta pode retornar um de vários códigos HTTP, caso tenha ou não ocorrido um erro no processamento do pedido, os quais são tratados nesta classe. No caso de não haver erros, o JSON no *body* da resposta é serializado para uma estrutura.

Modelo de dados

As respostas da API são devolvidas no formato JSON e como tal têm de ser serializadas para uma estrutura de modo a permitir operar com os dados nelas presentes.

A linguagem Swift tem disponível um protocolo "Codable" que permite automatizar este processo. Apenas é necessário associar a chave e o tipo de dados que cada elemento do JSON presente na chamada REST retorna, tipos estes que foram definidos em estruturas da linguagem.

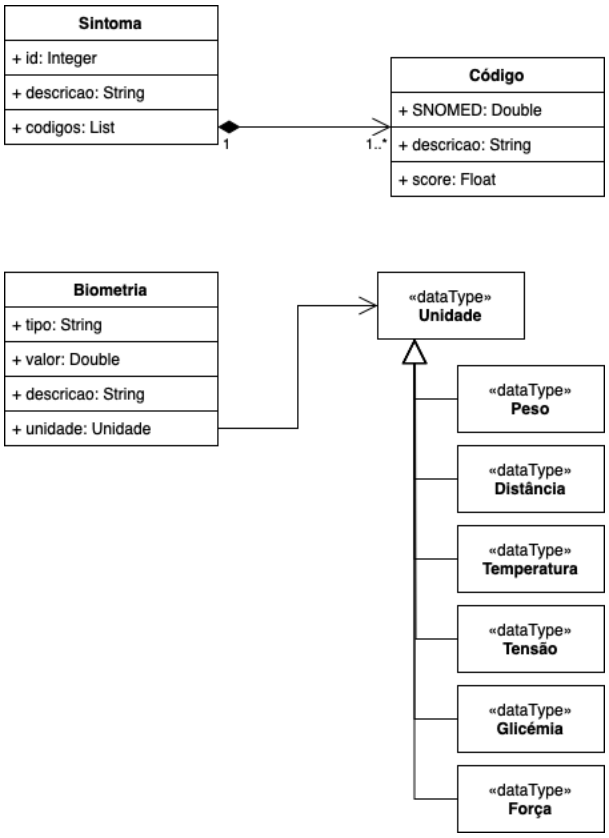


Figura 49 - Exemplo da estrutura dos dados na notação UML

São usadas estruturas diferentes para serializar os sintomas, códigos SNOMED, biometrias e unidade correspondente (Figura 49).

Reconhecimento de voz

Relativamente à componente que trata do ditado foi criada uma classe que encapsula a sua lógica.

Nela estão contemplados os métodos usados para autorizar a captação de áudio, bem como o começo e término do ditado. São também tratados quaisquer erros que possam ocorrer, pré ou pós processamento, bem como a conversão de áudio para o formato de entrada especificado pela função de processamento do *speech-to-text.*

A decisão de utilizar uma classe para este efeito tem vantagens em caso de troca do serviço ou *framework* de reconhecimento de voz.

8.3 Ecrãs da aplicação

Os ecrãs da aplicação foram desenvolvidos à semelhança dos *mockups* fornecidos, no entanto, contam com algumas alterações e edições. Serão apresentados nas páginas seguintes.

Login

O ecrã de *login* apresentado na Figura 50 foi feito segundo a identidade visual da empresa e tem dois campos nos quais serão introduzidas as credenciais para autenticação do utilizador.

Figura 50 - Ecrã de login da aplicação

Ecrã inicial

O ecrã inicial (Figura 51) conta com um menu onde é efetuada pelo médico a seleção de qual dos campos do SOAP pretende preencher.

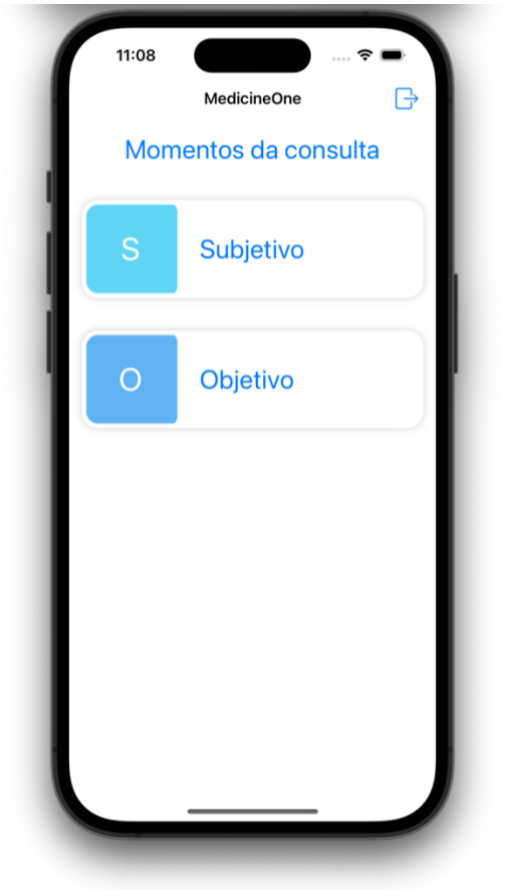


Figura 51 - Ecrã inicial da aplicação

Neste ecrã, em relação aos *mockups*, os itens do SOAP foram aumentados de tamanho para que possam ser mais facilmente selecionados. Foi também adicionado um botão de *logout* para que o utilizador possa terminar a sua sessão.

Subjetivo

O ecrã referente ao campo “subjetivo” do SOAP (Figura 52) conta com uma caixa de texto e botões para iniciar o ditado, parar e processar.

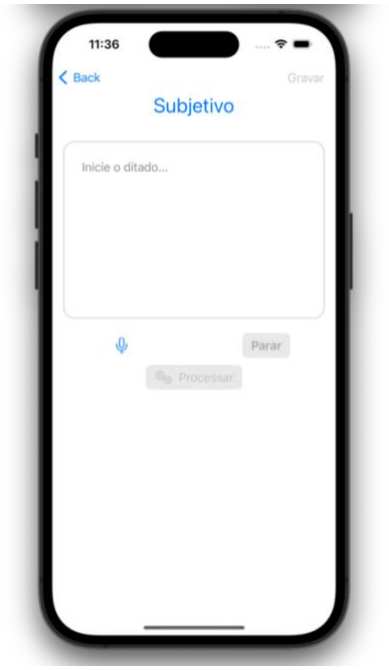


Figura 52 - Ecrã "subjetivo" por preencher

As informações ditadas irão preencher a caixa de texto e habilitar o botão de processamento. Foi tomada a opção de ter um botão de processamento para que o médico pudesse editar quaisquer erros ou informações antes de enviar o texto para análise.

Figura 53 - Ecrã "subjetivo" preenchido

Após o processamento é apresentada a lista dos sintomas identificados, bem como o respetivo código SNOMED (Figura 53).

Numa tentativa de mitigar os erros cometidos pelo serviço, (por exemplo no caso de um sintoma mal identificado ou que não fosse o pretendido) ao selecionar um sintoma da lista é apresentado um *popup* no qual podemos trocar o sintoma de entre 5 opções e um *swipe* revela a opção de removê-lo da lista (Figura 54).

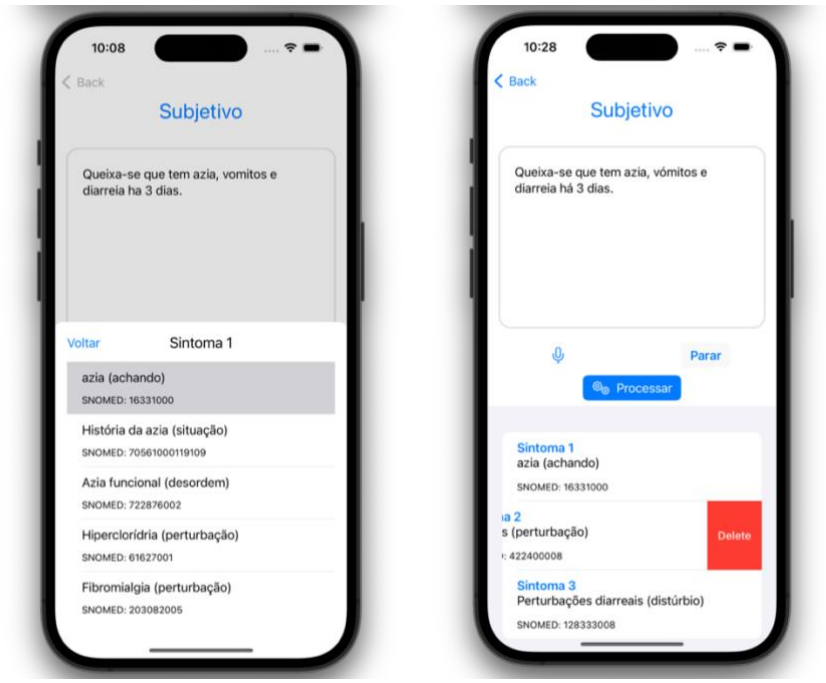


Figura 54 - Ações adicionais do ecrã "subjetivo"

Também foi adicionado um botão de gravar que, apesar de não ter implementada a lógica de gravação, imita a interação que seria feita entre a aplicação e o *software* de EHR.

Figura 55 - Ação de gravar

Objetivo

O ecrã referente ao campo “objetivo” (Figura 56) é semelhante ao do subjetivo, contando com os mesmos elementos (caixa de texto, botões de iniciar, terminar e processar).

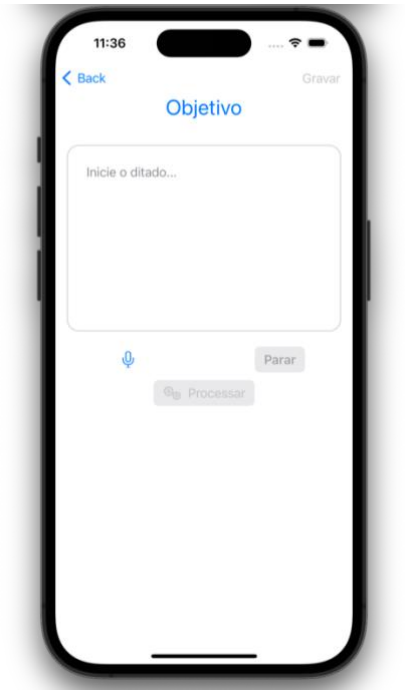


Figura 56 - Ecrã "objetivo" por preencher

O preenchimento do campo e processamento acontece do mesmo modo que no campo anterior. Depois de processado, a lista apresentada tem as biometrias, valor e unidade escolhidas (Figura 57).

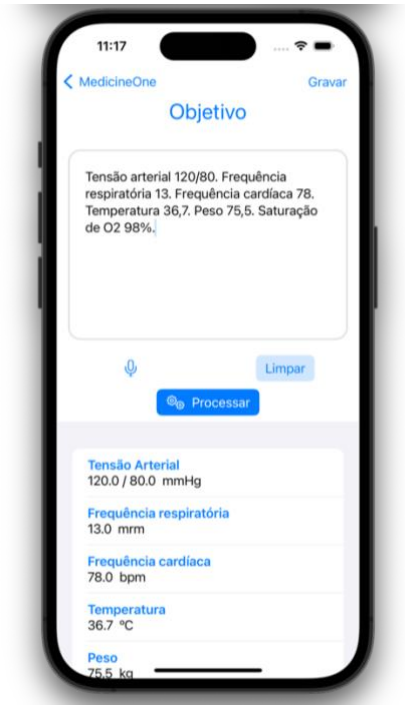


Figura 57 - Ecrã "objetivo" preenchido

Também o toque numa entrada da lista permite a edição do valor e unidade, um *swipe* permite apagá-la (Figura 58).

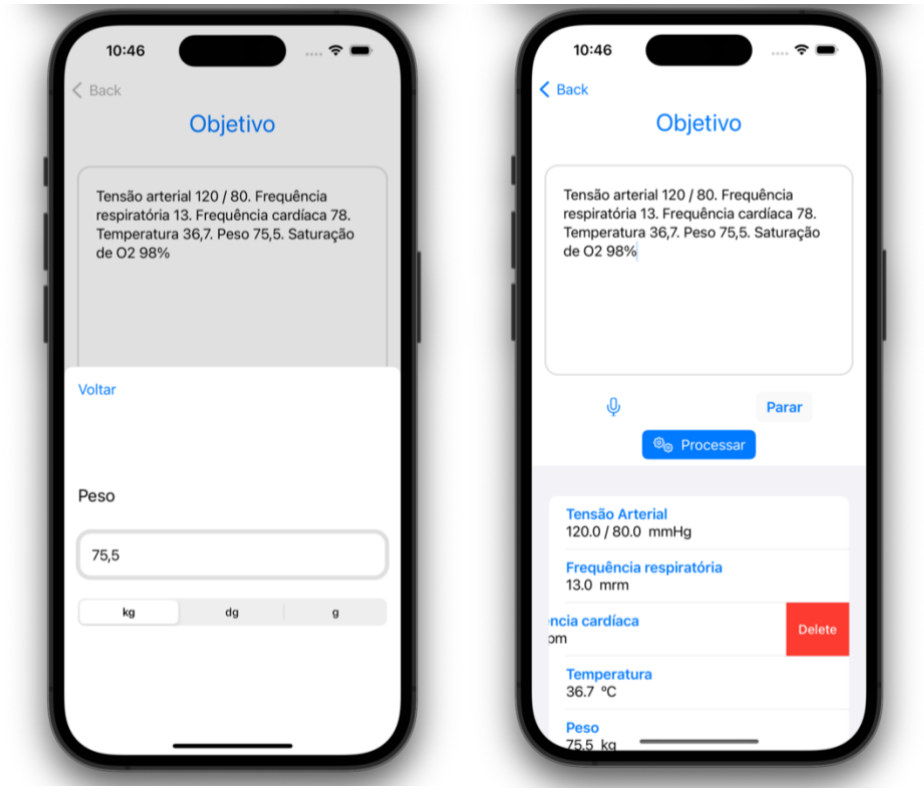


Figura 58 - Ações adicionais do ecrã "objetivo"

8.4 Motor de reconhecimento de voz

Durante o desenvolvimento foi possível testar o motor de voz nativo do iOS – SFSpeechRecognizer, motor este que apenas está disponível para dispositivos no ecossistema Apple. O seu uso revelou que, apesar de geralmente adequado, não tinha um bom desempenho quando usado em termos clínicos.

Testes

Para validar estas afirmações e comparar alternativas, foi desenvolvida uma estratégia de testagem. Consistiu na elaboração de um guião com 8 frases de exemplo baseadas em registos reais, 4 referentes a um texto expectável de ser registado no campo “subjetivo” e 4 no campo “objetivo”, tendo sido requisitado a várias pessoas (colaboradores da empresa e amigos) que gravassem áudios ditando os exemplos do guião. Os áudios serviram como *input* ao motor de reconhecimento de voz e foi analisada a transcrição produzida.

As frases de exemplo, presentes na Figura 59, contêm um diferente número de termos específicos, podendo dizer que existe uma variação na dificuldade de cada uma delas. Procurou-se que fossem usadas palavras que denotam fonemas específicos da língua portuguesa (por exemplo “ão”, “lh”).

Campo subjetivo	Campo objetivo
Exemplo 1	Exemplo 1
Texto	**Texto**
O paciente queixa-se com febre, tosse e dores musculares. Sente dificuldade em respirar.	O doente mede 1 metro e 70, pesa 63 quilos. Tensão arterial sistólica 14 diastólica 7. Temperatura 36 graus.
Exemplo 2	Exemplo 2
Texto	**Texto**
Sente cansaço devido ao trabalho. Tem dores na região lombar.	Medida da cintura 80 centímetros. Medida da anca 82 centímetros. Perímetro cefálico 58 centímetros. Prega tricipital 32 ponto 3 milímetros.
Exemplo 3	Exemplo 3
Texto	**Texto**
Diabético. O doente desmaiou devido a uma hipoglicémia.	Dinamometria lombar 112 newtons. Dinamometria escapular direita 28. Dinamometria manual esquerda 31 newtons.
Exemplo 4	Exemplo 4
Texto	**Texto**
Utente com lesões eritemato-crostosas no membro inferior direito. Refere prurido intenso.	Saturação 98 por cento. Frequência cardíaca 67 batimentos por minuto. Percentagem de massa gorda 24 por cento.

Figura 59 - Exemplos do guião de testes

Foram também registados o género e região de naturalidade para demarcar a existência de diferentes pronúncias (Figura 60), além de gerados alguns exemplos tanto usando ferramentas de *text-to-speech*, como adicionando ruído a alguns áudios.

Sujeito	Género	Naturalidade
Artificial 1	F	
Artificial 2	M	
Artificial 3	F	
Artificial 4	F	
Artificial 5	M	
Real 1	M	Coimbra
Real 10	M	Coimbra
Real 11	M	Porto
Real 12	F	Coimbra
Real 13	F	Coimbra
Real 2	M	Leiria
Real 3	M	Coimbra
Real 4	M	Coimbra
Real 5	M	Coimbra
Real 6	M	Viana do Castelo
Real 7	M	Porto
Real 8	F	Coimbra
Real 9	M	Coimbra

Figura 60 - Sujeitos de teste

Dos sujeitos reais, 5 repetiram 3 vezes a leitura dos exemplos para garantir consistência dos resultados.

No total contabilizaram-se vozes de 13 sujeitos reais, 5 artificiais e 2 adicionando ruído, ou seja, 224 áudios.

Alternativas e métricas

De modo a comparar a *performance* do motor SFSpeechRecognizer foram testados outros serviços de *speech-to-text* dos provedores Amazon (AWS Transcribe), Google (Google Cloud Speech) e Microsoft (Azure Speech to Text).

As métricas de medida usadas para efetuar a comparação são comuns na avaliação de sistemas de transcrição:

- Word Error Rate (WER) – Percentagem das palavras erradas, inseridas ou inexistentes na transcrição em comparação com o texto original.
- Character Error Rate (CER) – Percentagem dos caracteres errados, inseridos ou inexistentes na transcrição em comparação com o texto original.

Devido à pontuação das palavras na língua portuguesa, foram também comparados os exemplos descartando qualquer pontuação.

Exemplo com pontuação: Saturação 98 por cento. Frequência cardíaca 67 batimentos por minuto. Percentagem de massa gorda 24 por cento.

Exemplo sem pontuação: saturacao 98 por cento frequencia cardiaca 67 batimentos por minuto percentagem de massa gorda 24 por cento

Resultados

Os áudios foram processados pelos vários serviços de *speech-to-text* e os textos resultantes guardados num documento JSON (Figura 61).

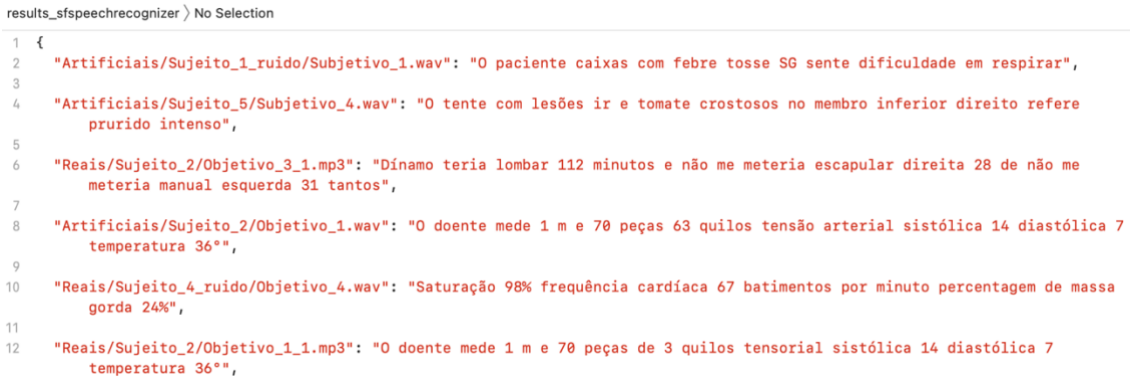


Figura 61 - Resultados do reconhecimento de voz

As transcrições foram depois analisadas, calculadas as métricas e guardadas numa folha de cálculo.

Seguem-se os gráficos e interpretação de resultados.

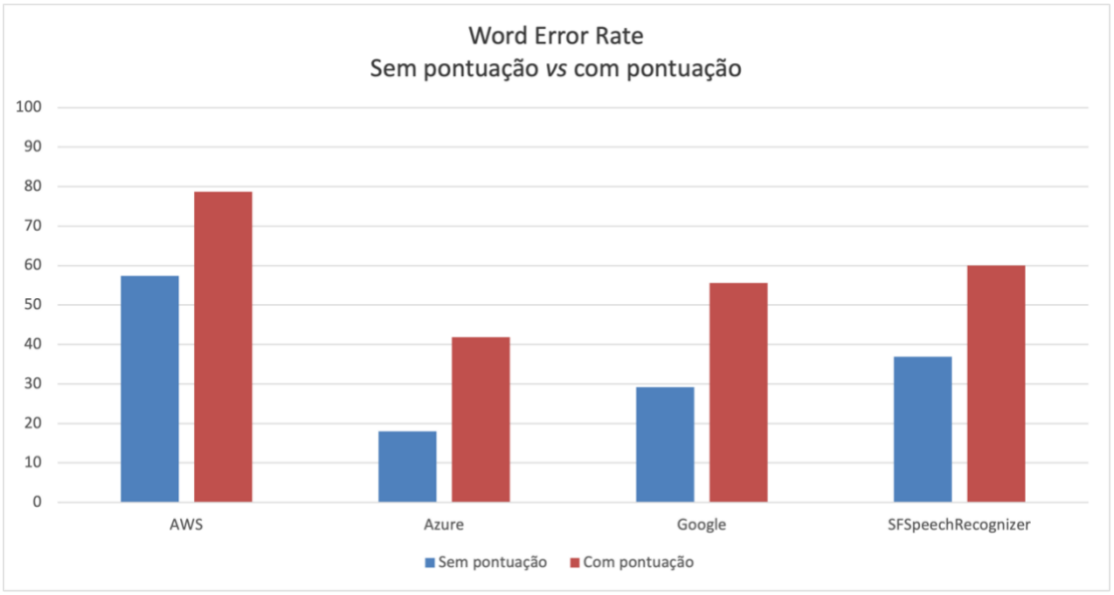


Figura 62 - Word Error Rate no speech-to-text

A partir do gráfico anterior (Figura 62) é percetível que existe uma grande diferença quando comparamos a transcrição com e sem pontuação. Isto acontece porque, por exemplo um erro de acentuação conta como uma palavra errada na estatística do WER.

No entanto, independentemente deste aspecto é notável que o serviço com menor WER é o do Azure enquanto que o com maior WER é o da AWS. Esta diferença é perto de metade quando a comparação é feita tendo em conta a pontuação e cerca de um terço descartando-a.

Em relação ao motor de *speech-to-text* da Apple (SFSpeechRecognizer), o desempenho do Azure é muito superior havendo uma redução de cerca de 19% dos erros (em exemplos pontuados).

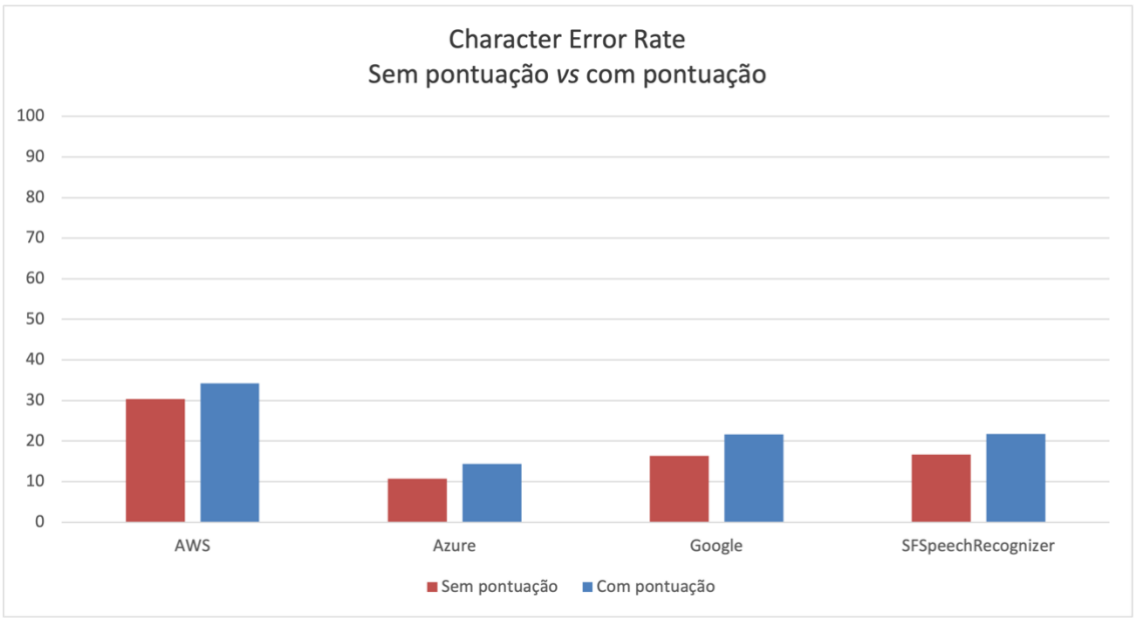


Figura 63 - Character Error Rate no speech-to-text

A métrica CER (Figura 63) confirma novamente que o serviço com melhor desempenho é o do Azure. Em todos eles existe um maior erro quando considerada a pontuação, o que indica que nenhum consegue pontuar corretamente os textos.

Faz sentido comparar também outros aspetos como o impacto dos exemplos usando vozes geradas artificialmente com as de sujeitos reais, bem como o ruído introduzido.

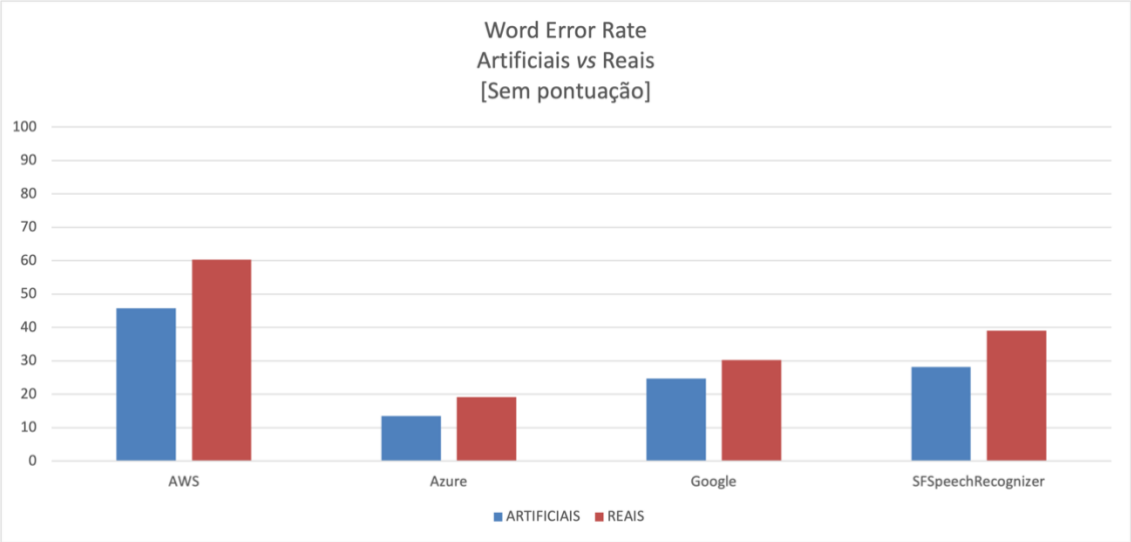


Figura 65 - Word Error Rate em exemplos artificiais vs reais

Na generalidade todos os serviços têm um pior desempenho quando confrontados com vozes de sujeitos reais (Figura 65), visto que existem diferenças principalmente de dicção e ritmo.

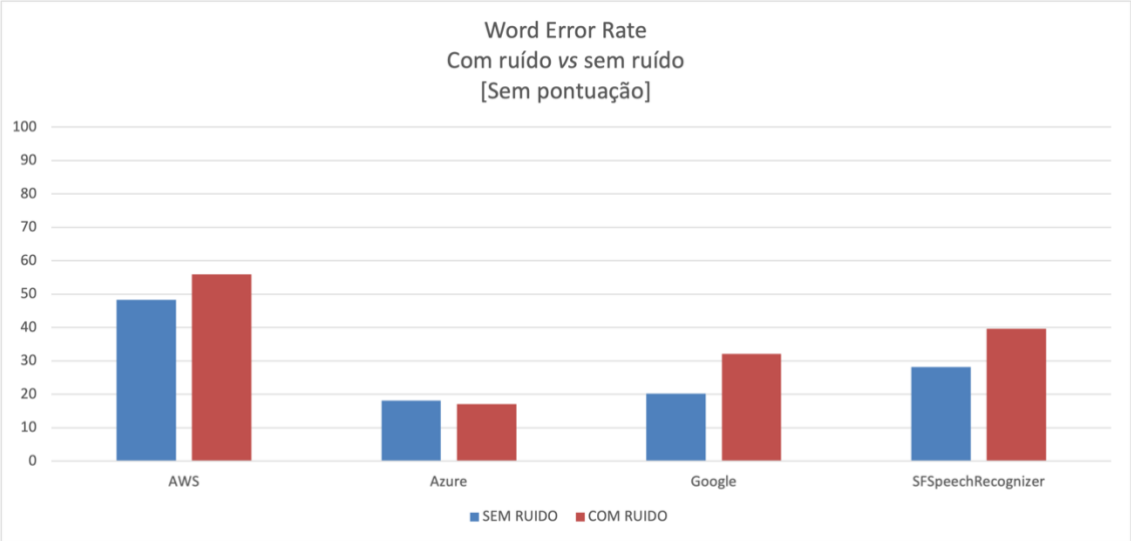


Figura 64 - Word Error Rate em exemplos com e sem ruído

A Figura 64 mostra que o ruído causa um pior desempenho em todos os serviços com exceção do Azure, que obteve um ligeiro melhor desempenho, possivelmente um *outlier*. De notar que apenas foram considerados 16 áudios com ruído para esta comparação.

Implementação do Azure Speech to text

Tendo obtido um melhor desempenho, quando comparado com os restantes serviços, o serviço da Microsoft Azure irá substituir o motor nativo de reconhecimento de voz do iOS.

Para a sua implementação foi necessário captar o som do microfone e convertê-lo para o formato de entrada do serviço por meio da *framework* AVFoundation (Figura 66).

```
private func readDataFromMicrophone() {
    let inputNode = audioEngine.inputNode

    let inputFormat = inputNode.outputFormat(forBus: 0)

    //Azure format
    let recordingFormat = AVAudioFormat(commonFormat: .pcmFormatInt16, sampleRate: Double(self.sampleRate), channels: 1, interleaved: false)

    guard let formatConverter =  AVAudioConverter(from:inputFormat, to: recordingFormat!)
    else {
        return
    }
    // Install a tap on the audio engine with the buffer size and the input format.
    audioEngine.inputNode.installTap(onBus: 0, bufferSize: AVAudioFrameCount(2048), format: inputFormat) { (buffer, time) in

        self.conversionQueue.async { [self] in
            let outputBufferCapacity = AVAudioFrameCount(buffer.duration * recordingFormat!.sampleRate)

            guard let pcmBuffer = AVAudioPCMBuffer(pcmFormat: recordingFormat!, frameCapacity: outputBufferCapacity) else {
                print("Failed to create new pcm buffer")
                return
            }
            pcmBuffer.frameLength = outputBufferCapacity

            var error: NSError? = nil
            let inputBlock: AVAudioConverterInputBlock = {inNumPackets, outStatus in
                outStatus.pointee = AVAudioConverterInputStatus.haveData
                return buffer
            }
            formatConverter.convert(to: pcmBuffer, error: &error, withInputFrom: inputBlock)
```

Figura 66 - Implementação do Azure Speech to Text

Após convertido, o processamento é feito pelo SDK disponibilizado pelo serviço usando as credencias de acesso ao mesmo, tendo sido mantido o funcionamento do mesmo modo (tocar para iniciar o reconhecimento e para parar) e mostrando resultados em tempo real.

Diferenças no reconhecimento

Comparativamente ao anterior mecanismo de efetuar *speech-to-text*, o serviço do Azure passou a conseguir reconhecer a palavra “dinamometria”, palavra componente de várias biometrias visadas no campo “objetivo” e que o SFSpeechRecognizer não conseguiu reconhecer uma única vez durante os testes efetuados.

Esta melhoria é significativa porque, integrando esta palavra várias biometrias de requisito (por exemplo “dinamometria lombar”, “dinamometria escapular esquerda”), o seu não reconhecimento na componente de ditado iria prejudicar a sua identificação por parte do serviço de processamento do campo “objetivo” ou implicava que o médico perdesse tempo a corrigir a palavra no texto.

Ainda assim, uma melhor qualidade na transcrição é um fator que se revela chave para o sucesso da prova de conceito, tendo em conta que o objetivo é agilizar o processo de registo de informações na consulta.

Capítulo 9
Arquitetura da solução implementada

Ao longo do desenvolvimento foram efetuadas alterações aos serviços usados o que impactou a arquitetura da solução, no entanto estes aspetos melhoraram o trabalho realizado.

Neste capítulo é apresentada a atualização da arquitetura refletindo a mudança dos componentes da solução, numa primeira secção. A segunda secção contém a validação das funcionalidades implementadas em linha com os objetivos. Finalmente na terceira secção são projetados futuros elementos a implementar.

9.1 Diagrama de componentes

No Capítulo 6 foi feito uso da notação C4 para definir os vários níveis de abstração de uma possível solução, de forma hierárquica, usando os diagramas de contexto, *containers* e componentes, respetivamente. A sua elaboração ocorreu numa fase anterior à implementação e neles estavam representados os serviços expectáveis de serem utilizados.

Com o decorrer da implementação e tendo em conta os testes efetuados, provou-se necessário fazer algumas mudanças nos serviços escolhidos. Estas mudanças não impactaram a arquitetura a um alto nível, mantendo-se os diagramas de contexto e *containers* atualizados em relação ao implementado, enquanto que o diagrama de componentes sofreu alterações. Na procura de representar a solução final foi corrigido o diagrama e apresenta-se na figura seguinte.

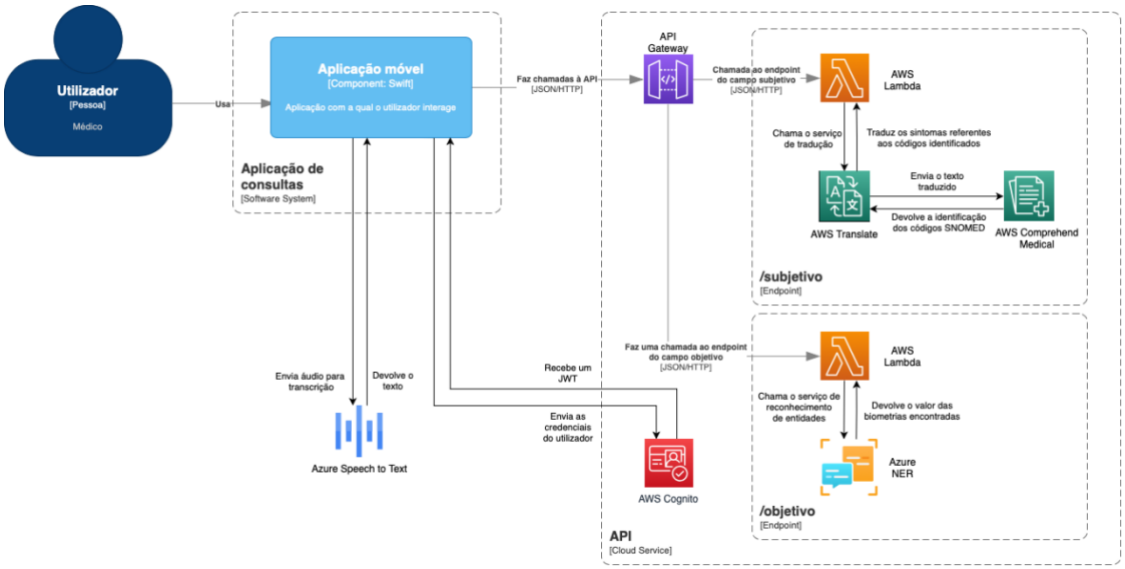


Figura 67 - Diagrama de componentes de acordo com a implementação

Pode verificar-se a troca do mecanismo de *speech-to-text*, mas também do serviço para reconhecimento de entidades no processamento do *endpoint* referente ao campo "objetivo" e a utilização do AWS Cognito para autenticação dos utilizadores.

9.2 Validação dos requisitos

Tendo finalizado o desenvolvimento, a aplicação foi testada de forma a validar o cumprimento dos requisitos funcionais. De seguida será apresentada uma tabela com a funcionalidade esperada e os resultados, bem como algumas observações.

Tabela 8 - Validação dos requisitos funcionais

Identificação	Requisito(s)	Cumprido(s)	Observações
REQ.S-1.0 e REQ.O-1.0	**Ditar texto para que o campo seja preenchido**	**Sim**	
Campo Subjetivo			
REQ.S-2.0	**Reconhecer códigos SNOMED CT a partir do texto**	**Sim**	
REQ.S-3.0	**Visualizar quais os códigos reconhecidos**	**Sim**	
REQ.S-4.0	**Eliminar um código reconhecido**	**Sim**	
REQ.S-5.0	**Editar um código reconhecido**	**Sim**	**- São apresentadas 5 alternativas das quais se pode escolher**
Campo Objetivo			
REQ.O-2.0 a REQ.O-18.0	**Reconhecer parâmetros visados a partir do texto**	**Sim**	**- Os parâmetros devem aparecer no texto como um par chave-valor**
REQ.O-19.0	**Visualizar quais os parâmetros reconhecidos**	**Sim**	
REQ.O-20.0	**Eliminar um parâmetro reconhecido**	**Sim**	
REQ.O-21.0	**Editar um parâmetro reconhecido**	**Sim**	

Quanto aos requisitos não funcionais, em questões de usabilidade são cumpridos os três passos definidos para o processamento de um campo (iniciar o ditado, terminá-lo e processar) bem como foi feito um dimensionamento dos botões para facilitar o seu uso.

Figura 68 - Passos para o processamento de um campo

A segurança foi avaliada usando a ferramenta Postman ao fazer chamadas não autenticadas à API, a qual devolve o código HTTP 401 correspondente à não autorização.

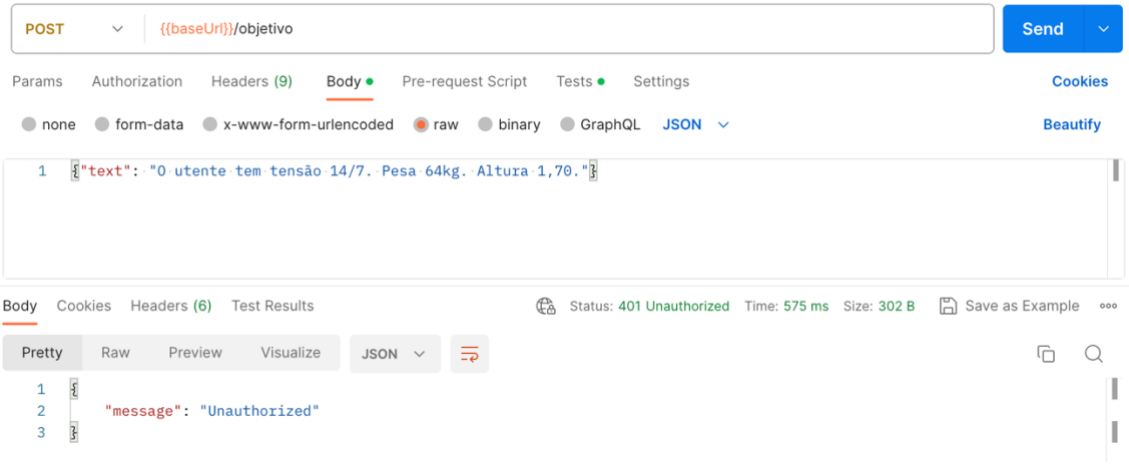


Figura 69 - Exemplo de uma chamada não autorizada

Tendo sido validados todos os requisitos pode garantir-se o cumprimento das metas definidas durante o estágio.

9.3 Objetivos futuros

Durante o estágio, partindo da prova de conceito foi implementado um protótipo que cumpre as funcionalidades pretendidas. No entanto, este não foi testado em ambiente real, mas apenas internamente na empresa. Assim, um dos objetivos futuros seria a disponibilização da aplicação a um conjunto de médicos que a iria testar e dar o seu *feedback*.

No estado de desenvolvimento atual, a aplicação permite a transcrição e processamento dos campos "subjetivo" e "objetivo" do SOAP. Como objetivos de continuação do desenvolvimento, fora do âmbito do estágio, seriam uma implementação semelhante para os restantes campos do SOAP – "avaliação" e "plano".

Atualmente, a lista de sintomas e biometrias resultantes do processamento não estão a ser usadas no contexto do processo do doente. No futuro, uma integração com o software de EHR da empresa (M1) deveria permitir preencher os campos tendo em conta o doente ao qual está a ser feita a consulta, bem como importar os dados reconhecidos de volta para o mesmo.

O reconhecimento de um leque mais alargado de biometrias e a disponibilização de uma aplicação semelhante dedicada à enfermagem de triagem que permitisse ao médico acrescentar elementos às informações já reconhecidas por estes, durante a consulta, poderiam ser complementos benéficos ao desenvolvimento atual.

Capítulo 10
Conclusão

Neste capítulo é feita uma reflexão sobre os trabalhos realizados no decorrer do estágio, seja durante a fase inicial de planeamento como do desenvolvimento e sobre a aprendizagem.

10.1 Plano de trabalhos

Tendo em conta os objetivos delineados na Secção 1.3 e a calendarização efetuada na Secção 3.4 pode-se concluir que o estágio cumpriu o plano de trabalhos.

Durante o estágio apenas o reajuste dos objetivos feito no início do estágio provocou alterações à calendarização, mas após estas mudanças tudo aconteceu conforme planeado. No segundo semestre, o planeamento de fases de teste intercaladas com fases de desenvolvimento permitiu que se contabilizassem possíveis mudanças, tais como as que aconteceram: escolha de outro serviço de reconhecimento de entidades (Seccção 7.6) alteração do motor de *speech-to-text* (Secção 8.4).

O MVP previsto esteve pronto na data especificada, tendo sido refinado até à versão atual obtida no final do estágio.

10.2 Reflexões sobre o trabalho realizado

A proposta de estágio partiu de uma ideia interna da empresa para desenvolver uma prova de conceito na forma de uma aplicação, com o objetivo de agilizar o processo de consulta de um médico. Esta ideia foi depois pormenorizada e concentrada no momento inicial da consulta, a anamnese, fazendo uso do método de registo SOAP focando apenas nos primeiros dois campos: "subjetivo" e "objetivo". O uso de ditado para o preenchimento dos campos, NLP para o seu processamento, a anotação de códigos usando o vocabulário SNOMED, uma lista de biometrias a identificar, a preferência do uso de soluções *cloud* seguindo a prática da empresa e o desenvolvimento para iOS, foram as condições colocadas.

Partindo destes pressupostos foi necessário analisar quais os produtos presentes no mercado (Secção 2.1), bem como o funcionamento da tecnologia que lhes dava suporte (Secção 2.2). Constatando que as soluções existentes no mercado não atendiam aos requisitos, principalmente porque não estavam disponíveis em português ou não permitiam o formato de anotação pretendido, entre outros, provou-se necessário desenvolver uma implementação à medida das necessidades.

Tendo em conta a constituição de uma aplicação deste tipo e atendendo às questões de *time-to-market* associadas com o desenvolvimento de uma prova de conceito, foram analisados um por um os componentes necessários para a implementação de uma aplicação deste género (Capítulo 2).

A análise dos componentes permitiu traçar um plano de desenvolvimento (Capítulo 3) e especificar os requisitos (Capítulo 4) para uma possível implementação. Dado o aspeto inovador da prova de conceito e a área de atuação visada, foi ainda necessário fazer um levantamento dos riscos associados ao projeto (Capítulo 5).

Nesta fase foi elaborada uma visão inicial da arquitetura (Capítulo 6), no entanto devido às possibilidades de escolha de entre vários serviços disponíveis, seguiu-se uma análise inicial para escolher o mais adequado à aplicação pretendida (secções 6.2 e 6.3).

Seguiu-se o desenvolvimento dos componentes de processamento (Capítulo 7), iniciando por aqueles associados ao campo "subjetivo" passando por elementos de disponibilização da API e autenticação. Ainda neste capítulo, surge a estratégia de processamento do campo "objetivo" que, não existindo serviços no mercado talhados para o reconhecimento das entidades visadas, levou a que se treinásse um modelo de inteligência artificial, ainda que usando um serviço *cloud* para esse efeito. O capítulo termina com os testes à API de *backend* criada para o processamento destes campos.

O *frontend* consiste na aplicação iOS e além de integrar com a API anteriormente desenvolvida, tem nela inserida a componente de ditado (Capítulo 8). Os ecrãs foram baseados em *mockups* fornecidos pela empresa, mas contam com melhorias resultantes das fases de testes que foram intercaladas com o desenvolvimento. Destas destaca-se a mudança do motor de ditado nativo dos dispositivos Apple para o uso de um serviço na *cloud* pela sua superior capacidade de reconhecimento (Secção 8.4).

Durante o processo de desenvolvimento foram constatadas necessidades de alteração de serviços em relação ao desenho inicial da implementação e que contribuiram para o sucesso do protótipo (MVP) que materializou a prova de conceito. Estas mudanças foram refletidas num novo diagrama da arquitetura (Capítulo 9). Verificando que o trabalho desenvolvido atendeu aos objetivos colocados, foi feita uma projeção do trabalho futuro.

Concluindo, o trabalho foi um sucesso relativamente à prova de conceito, tendo sido provado que é possível desenvolver uma solução que permita reconhecer e sintetizar registos dos campos "subjetivo" e "objetivo" do método SOAP usando técnicas de NLP. Também a aplicação mostra que é viável ter um produto que integra este processamento com mecanismos de ditado num dispositivo facilmente acessível aos médicos.

10.3 Aprendizagem

O estágio teve um peso importante na formação, na medida em que foi uma inserção num contexto de trabalho menos académico e mais empresarial. Existiu a adaptação a um meio de atuação, visto que a empresa fornece *software* para a área da saúde e com isso surge o foco para soluções expectáveis de serem utilizadas em ambiente real, por profissionais de saúde e para pessoas em necessidade. Foi com esta mentalidade que seguiu o estágio: desenvolver algo que pudésse auxiliar processos do "mundo real", a ser utilizado por "pessoas" e que satisfizesse as suas necessidades.

Além desta ambientação ao contexto do negócio, também existiu um desafio tecnológico ao utilizar serviços *cloud* de diferentes provedores, mecanismos de processamento de linguagem natural e ainda o desenvolvimento para um sistema operativo diferente (iOS) – todos eles nunca antes abordados em trabalho académico.

A identificação de possíveis otimizações para evitar custos desnecessários (por exemplo em chamadas aos serviços *cloud*) mas também ao nível da usabilidade e manutenção

posterior dos serviços foram tidos em conta ao longo do estágio. Também a justificação das escolhas dos serviços disponíveis no mercado numa perspetiva de negócio, contando com o tempo de implementação, preço e funcionalidade.

Para além destes, também os processos acessórios ao desenvolvimento como reuniões, documentação e organização do plano de trabalhos fizeram parte do processo e ditaram o ritmo do projeto.

Em resumo, foram alcançados os objetivos da proposta de estágio feita pela empresa, num processo que contribuiu para a aprendizagem de aspetos de trabalho num contexto de negócio, mas também de novas tecnologias. A experiência enriqueceu não só a formação académica, mas também a formação profissional do estagiário.

Referências

[1] *Advanced Artificial Intelligence API*. (2022). NLP Cloud. Consultado a 3 de novembro de 2022. Disponível em https://nlpcloud.com/

[2] Ajayi, D. (2020, Dezembro 3). *How BERT and GPT models change the game for NLP*. Watson Blog. Disponível em https://www.ibm.com/blogs/watson/2020/12/how-bert-and-gpt-models-change-the-game-for-nlp/

[3] Apple. (2023). *Apple Developer Documentation*. Consultado a 15 de Janeiro de 2023. Disponível em https://developer.apple.com/documentation/speech/sfspeechrecognizer

[4] Apple. (2018). *Xcode - Apple Developer*. Apple.com. Consultado a 29 de dezembro de 2022. Disponível em https://developer.apple.com/xcode/

[5] *Assess CT*. Assess-Ct.eu. Consultado a 29 de dezembro de 2022. Disponível em https://assess-ct.eu/start0/

[6] Atlassian. (2023). *Welcome to Jira Software. Atlassian*. Consultado a 15 de Janeiro de 2023. Disponível em https://www.atlassian.com/software/jira/guides/getting-started/introduction#what-is-jira-software

[7] baeldung. (2021, Agosto 2). *Difference Between MVC and MVP Patterns*. Baeldung. Consultado a 3 de novembro de 2022. Disponível em https://www.baeldung.com/mvc-vs-mvp-pattern

[8] Berger, A., Della, V., & Della, S. (1996). *A Maximum Entropy Approach to Natural Language Processing*. Disponível em https://aclanthology.org/J96-1002.pdf

[9] Berkeley University Extension. (2020, Dezembro 16). 11 Most In-Demand Programming Languages in 2021. Berkeley Boot Camps. Consultado a 15 de Janeiro de 2023. Disponível em https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/

[10] Chacon, S. (2012). *About Git*. Git. Consultado a 10 de Janeiro de 2023. Disponível em https://git-scm.com

[11] *Clinician solutions | 3M Health Information Systems*. (2020). 3M. Consultado a 22 de dezembro de 2022. Disponível em https://www.g2speech.com/solutions/core-products/ https://www.3m.com/3M/en_US/health-information-systems-us/create-time-to-care/clinician-solutions/

[12] *Cloud Natural Language*. (2022). Google Cloud. Consultado a 3 de novembro de 2022. Disponível em https://cloud.google.com/natural-language

[13] *Core Products*. (2022). G2 Speech. Consultado a 22 de dezembro de 2022. Disponível em https://www.g2speech.com/solutions/core-products/

[14] Dainton, D. (2023, Julho 7). *newman-reporter-htmlextra*. GitHub. Disponível em https://github.com/DannyDainton/newman-reporter-htmlextra

[15] *DeepScribe - AI-Powered Medical Scribe*. (2022). Deepscribe AI. Consultado a 22 de dezembro de 2022. Disponível em https://www.deepscribe.ai/

[16] DeepTalk. *History and present of Natural Language Processing*. Deep-Talk.ai. Consultado a 10 de janeiro de 2023. Disponível em https://www.deep-talk.ai/post/history-and-present-of-natural-language-processing

[17] DSA. (2022, Dezembro 12). Capítulo 48 - Redes Neurais Recorrentes. Deep Learning Book. Disponível em https://www.deeplearningbook.com.br/redes-neurais-recorrentes/

[18] EHR/EMR, RCM Software for Ambulatory Practice| NextGen Healthcare. (2019). NextGen Healthcare. Disponível em https://www.nextgen.com/

[19] *Empresa - MedicineOne*. Medicine One. Consultado a 10 de janeiro de 2023. Disponível em https://www.medicineone.net/empresa

[20] *Extrair dados de saúde - Amazon Comprehend Medical – Amazon Web Services*. Amazon Web Services, Inc. Consultado a 3 de novembro de 2022. Disponível em https://aws.amazon.com/pt/comprehend/medical/

[21] Fonseca, C. (2021, Janeiro 23). *Word Embedding: fazendo o computador entender o significado das palavras*. Turing Talks. Disponível em https://medium.com/turing-talks/word-embedding-fazendo-o-computador-entender-o-significado-das-palavras-92fe22745057

[22] Galuh, R. T. (2022, Agosto 11). *MVC, MVP, MVVM: Which One to Choose?* MUO. Disponível em https://www.makeuseof.com/mvc-mvp-mvvm-which-choose/#why-do-we-need-architectural-design-patterns

[23] Google. (2023, Abril 5). Using the Healthcare Natural Language API | Cloud Healthcare API. Google Cloud. Disponível em https://cloud.google.com/healthcare-api/docs/how-tos/nlp#including_licensed_vocabularies

[24] Google Cloud. (2022, Agosto 31). NLP on Google Cloud. App.pluralsight.com. Disponível em https://app.pluralsight.com/library/courses/natural-language-processing-google-cloud/table-of-contents

[25] Granja, M., & Outeirinho, C. (2018). Registo médico orientado por problemas em medicina geral e familiar: atualização necessária. *Revista Portuguesa de Clínica Geral*, *34*(1), 40–44. Disponível em https://doi.org/10.32385/rpmgf.v34i1.12362

[26] *Healthcare Natural Language API | Cloud Healthcare API*. (2022). Google Cloud. Consultado a 3 de novembro de 2022. Disponível em https://cloud.google.com/healthcare-api/docs/concepts/nlp

[27] *Inteligência Artificial na Saúde*. (2019, Abril 18). SPMS. Disponível em https://www.spms.min-saude.pt/2019/04/inteligencia-artificial-na-saude/

[28] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022). Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications. Disponível em https://doi.org/10.1007/s11042-022-13428-4

[29] Kumar, Y., Koul, A., Singla, R., & Ijaz, M. F. (2022). Artificial intelligence in disease diagnosis: a systematic literature review, synthesizing framework and future research agenda. *Journal of Ambient Intelligence and Humanized Computing*. Disponível em https://doi.org/10.1007/s12652-021-03612-z

[30] *Language - Language Modeling for Apps | Microsoft Azure*. Azure.microsoft.com. Consultado a 11 de janeiro de 2023. Disponível em https://azure.microsoft.com/en-us/products/cognitive-services/language-service/

[31] Latysheva, N. (2019, Setembro 10). *Why do we use word embeddings in NLP?* Medium; Towards Data Science. Disponível em https://towardsdatascience.com/why-do-we-use-embeddings-in-nlp-2f20e1b632d2

[32] *Medical Speech Recognition that Recognizes How You Work | Nuance*. Nuance Communications. Consultado a 22 de dezembro de 2022. Disponível em https://www.nuance.com/healthcare/provider-solutions/speech-recognition.html

[33] Meireles, A. (2014, Fevereiro 6). Sistema informático está a roubar tempo aos doentes. Diário de Notícias. Disponível em https://www.dn.pt/portugal/sistema-informatico-esta-a-roubar-tempo-aos-doentes-3673056.html

[34] Merritt, R. (2022, Março 25). *What Is a Transformer Model?* NVIDIA Blog. Disponível em https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/

[35] Miñarro-Giménez, J. A., Martínez-Costa, C., Karlsson, D., Schulz, S., & Gøeg, K. R. (2018). Qualitative analysis of manual annotations of clinical text with SNOMED CT. *PLOS ONE*, *13*(12), e0209547. Disponível em https://doi.org/10.1371/journal.pone.0209547

[36] Morsch, J. A. (2022, Janeiro 11). *Anamnese médica: como fazer e qualificar a atenção ao paciente*. Pt.linkedin.com. Disponível em https://www.linkedin.com/pulse/anamnese-m%C3%A9dica-como-fazer-e-qualificar-aten%C3%A7%C3%A3o-ao-paciente-morsch/?originalSubdomain=pt

[37] Nakayama, H., Kubo, T., Kamura, J., Taniguchi, Y., & Liang, X. (2018). *doccano*. GitHub. Disponível em https://github.com/doccano/doccano

[38] *NLP - overview*. Cs.stanford.edu. Consultado a 4 de novembro de 2022. Disponível em https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/nlp/overview_history.html

[39] Nunnikhoven, M. (2019, Junho 28). *Analyzing Text on AWS with Amazon Comprehend*. Pluralsight. Disponível em https://app.pluralsight.com/library/courses/aws-comprehend-analyzing-text

[40] Postman. (2022). Postman API Platform | Tools. Postman API Platform. Disponível em https://www.postman.com/product/tools/

[41] *Processamento de linguagem natural – Amazon Comprehend – serviço da Web Amazon*. Amazon Web Services, Inc. Consultado a 3 de novembro de 2022. Disponível em https://aws.amazon.com/pt/comprehend/

[42] Project Jupyter. (2022). *Project Jupyter*. Jupyter. Consultado a 2 de dezembro de 2022. Disponível em https://jupyter.org/about

[43] Ravi, J. (2019, Agosto 5). *Natural Language Processing with PyTorch*. Pluralsight. Disponível em https://app.pluralsight.com/library/courses/natural-language-processing-pytorch/table-of-contents

[44] *Reconhecimento de voz ao serviço da saúde*. (2019, Maio 26). Glintt. https://www.glintt.com/pt/o-que-somos/noticias/Paginas/reconhecimento-voz-saude.aspx

[45] Redman, T. (2018, Abril 3). *If Your Data Is Bad, Your Machine Learning Tools Are Useless*. Harvard Business Review. Disponível em https://hbr.org/2018/04/if-your-data-is-bad-your-machine-learning-tools-are-useless

[46] *Robot Virtual Medical Scribe*. S10.Ai. Consultado a 22 de dezembro de 2022. Disponível em https://s10.ai/

[47] Scrum.org. *What is Scrum?* Scrum. Consultado a 16 de janeiro de 2023. Disponível em https://www.scrum.org/resources/what-is-scrum

[48] Shah, K. (2020, Novembro 18). *Architecture Presentation Patterns: MVC vs MVP vs MVVM | Thirdock Techkno*. Third Rock Techkno. Disponível em https://www.thirdrocktechkno.com/blog/architecture-presentation-patterns-mvc-vs-mvp-vs-mvvm/

[49] *SNOMED CT e Interoperabilidade Semântica nos Sistemas de Informação da Saúde*. (2014, Abril 15). SPMS. Disponível em http://www.spms.min-saude.pt/2014/04/snomed-ct-e-interoperabilidade-semantica-nos-sistemas-de-informacao-da-saude/

[50] Sommerville, I. et. al. (2011). SOFTWARE ENGINEERING Ninth Edition (p. 85). Disponível em https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf

[51] *Suki*. (2019). Suki.ai. Disponível em https://www.suki.ai/

[52] *Text Analysis*. MonkeyLearn. Consultado a 3 de novembro de 2022. Disponível em https://monkeylearn.com/

[53] *The case for investment*. (2021). Snomed. Disponível em https://www.snomed.org/SNOMED/media/SNOMED/documents/SNOMED-CT-Case-for-Investment-About-SNOMED-CT.pdf

[54] Visual Studio Code. (2022). *Documentation for Visual Studio Code*. Microsoft. Consultado a 2 de dezembro de 2022. Disponível em https://code.visualstudio.com/docs

[55] *Watson Natural Language Understanding - Overview*. IBM. Consultado a 3 de novembro de 2022. Disponível em https://www.ibm.com/cloud/watson-natural-language-understanding

Apêndices

Apêndice A

A.1 Planeamento inicial para o primeiro semestre

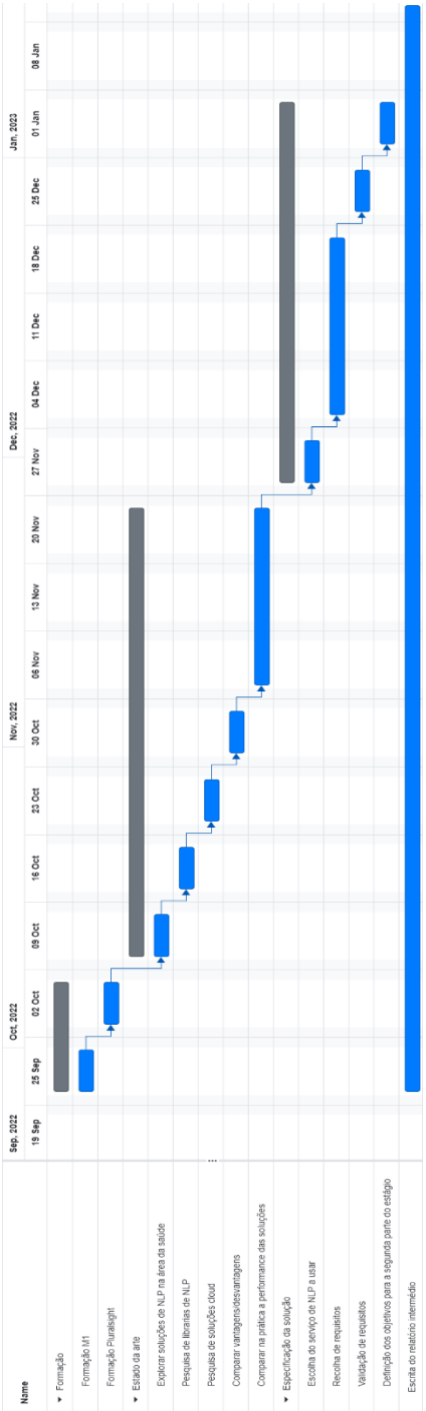


Figura 70 - Diagrama de Gantt referente ao planeamento inicial do primeiro semestre

A.2 Planeamento para o primeiro semestre após alteração dos objetivos

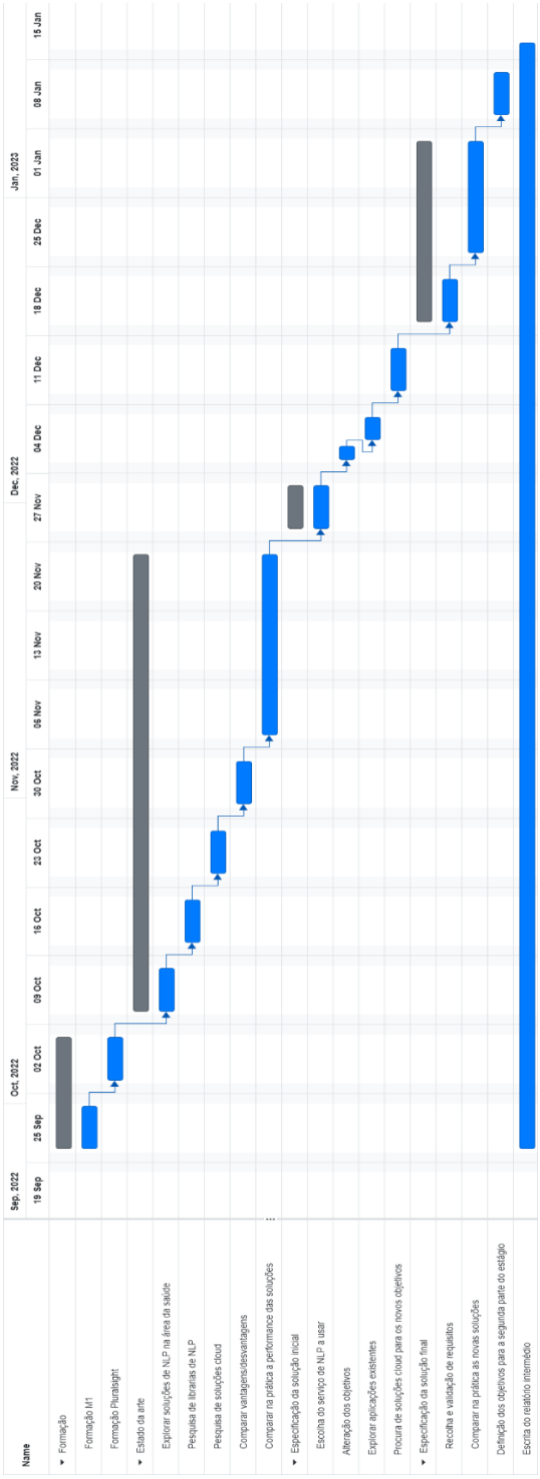


Figura 71 - Diagrama de Gantt referente ao planeamento do primeiro semestre após alteração dos objetivos

A.3 Planeamento para o segundo semestre

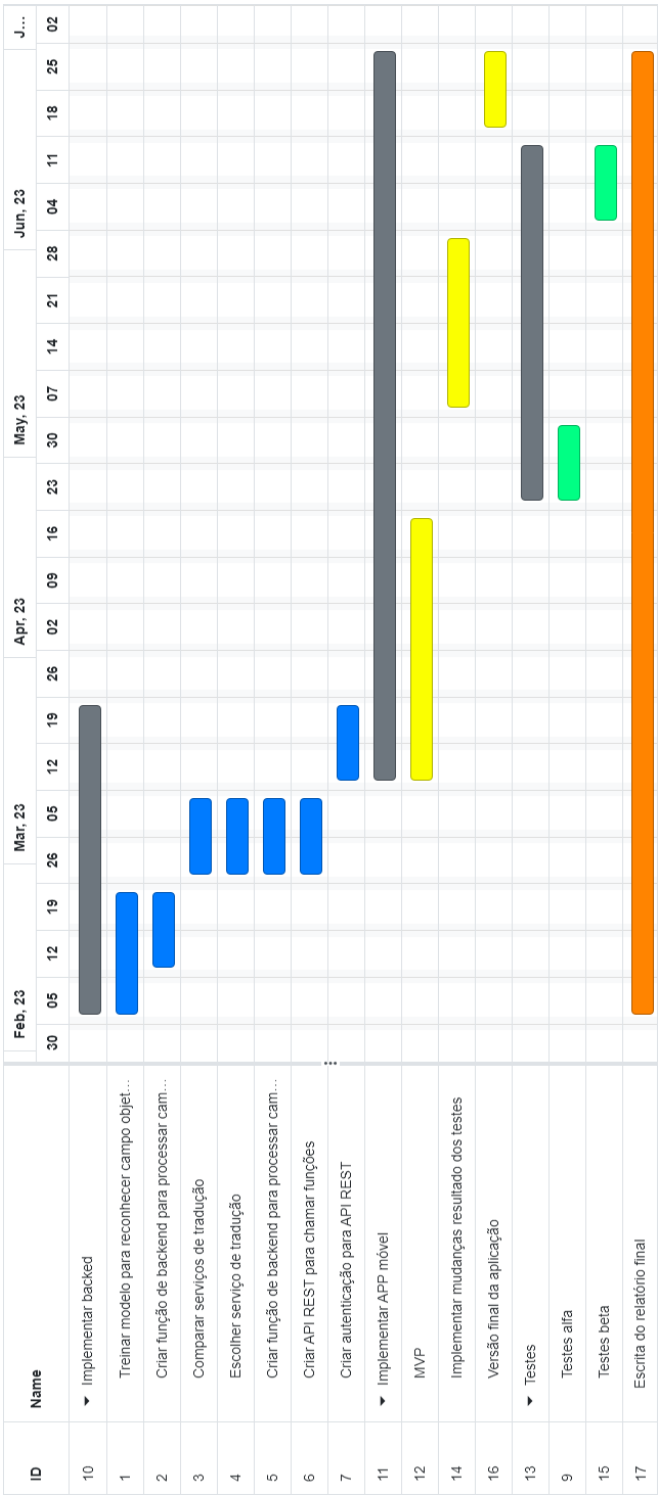


Figura 72 - Diagrama de Gantt referente ao planeamento para o segundo semestre

Apêndice B

B.1 Medições e biometrias a serem anotadas no campo “Objetivo”

- Altura
- Peso
- Temperatura
- Frequência cardíaca
- Frequência respiratória
- Tensão arterial sistólica
- Tensão arterial diastólica
- Saturação
- Glicémia
- Cintura
- Anca
- Perímetro cefálico
- Perímetro torácico mínimo
- Perímetro torácico máximo
- Perímetro braquial
- Dinamometria escapular esquerda
- Dinamometria escapular direita
- Dinamometria lombar
- Dinamometria manual esquerda
- Dinamometria manual direita
- Dor
- Percentagem de massa gorda
- Prega tricipital

Todos estes parametros são numéricos, maiores ou iguais a zero.

Apêndice C

C.1 Termos identificados pelo serviço da AWS

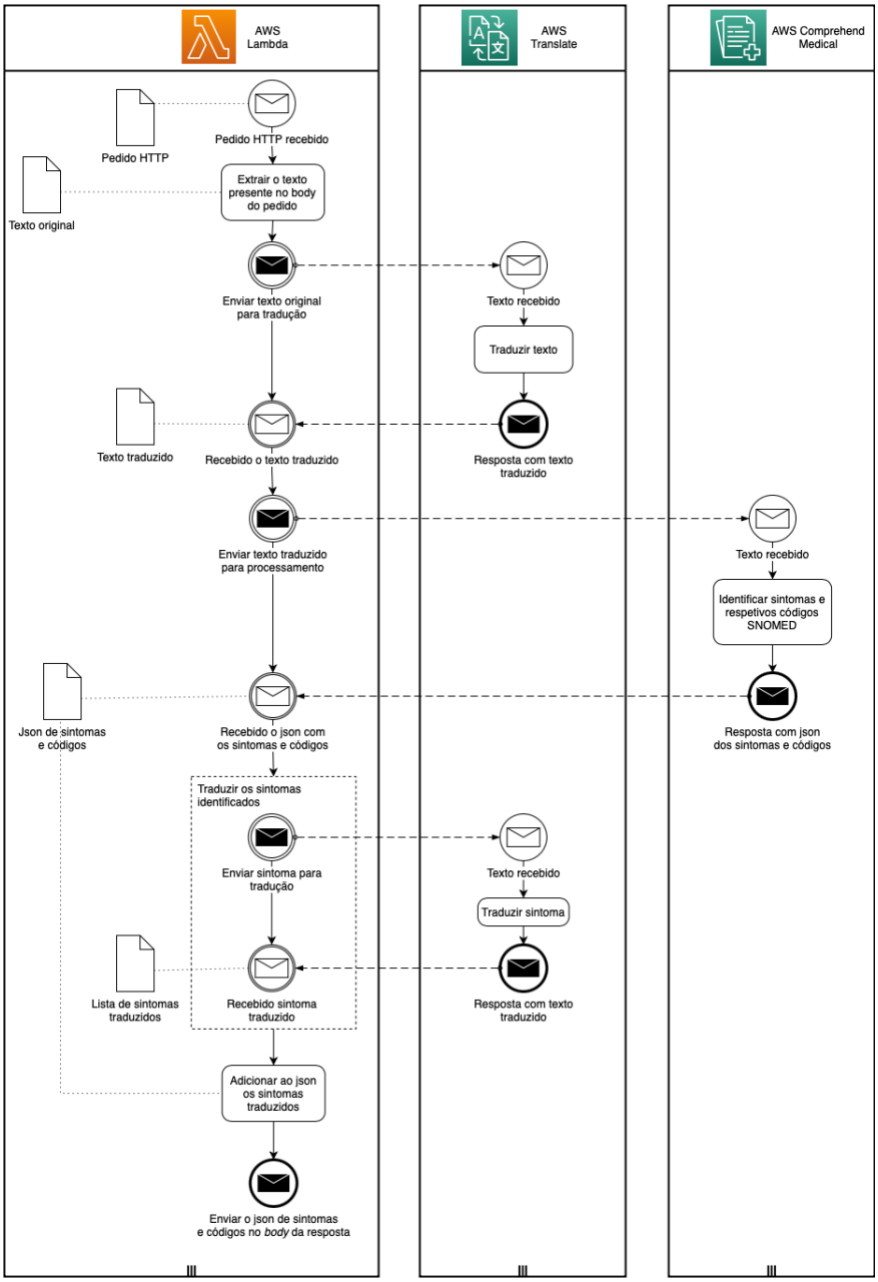
AWS	**66**
Termo bem identificado	**66**
Anamnese pre vacinal	1
Anca	1
Células do colo do útero	1
Ciatalgia	1
Comichão	1
Covid19	1
Dejecoes diarreicas	1
Diarreia	1
Dor de garganta	1
Dores de estomago	1
Edemas palpebrais	1
Hérnia discal	1
Holocraneana	1
Leucorreia	2
Lupus	1
Maléolo interno	1
Mau estar	1
Maxilo facial	1
Menorragia	1
Odinofagia	1
Ovários micropoloquisticos	1
Penso	34
Perímetro cefálico	1
Persistencia de dor	1
Pieira	1
Polimialgia; Odontalgia	1
Pré-vacinação	1
Regiao mamaria	1
Rinorreia	1
Sacrococigea; Amputação; Hemiparesia esq; Picossulfato de sodio	1
Síndrome vertiginoso	1
Taquicardia	1

C.2 Motivos que levaram a uma tradução insatisfatória

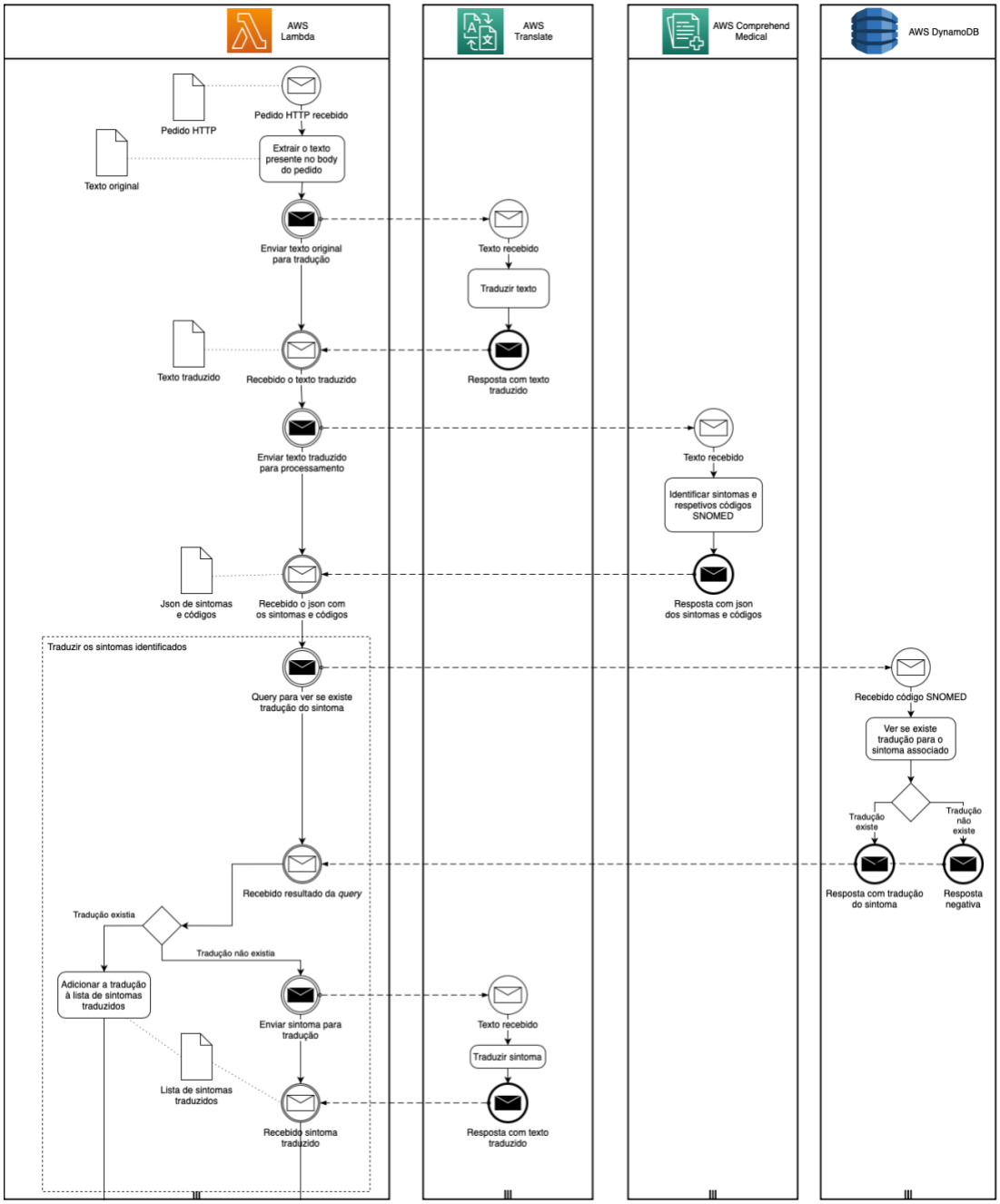
Nenhuma	**47**
Má tradução	1
Palavra mal traduzida	5
Termo não reconhecido	20
Texto mal estruturado	3
Várias inconsistências	17
Termo mal traduzido	1

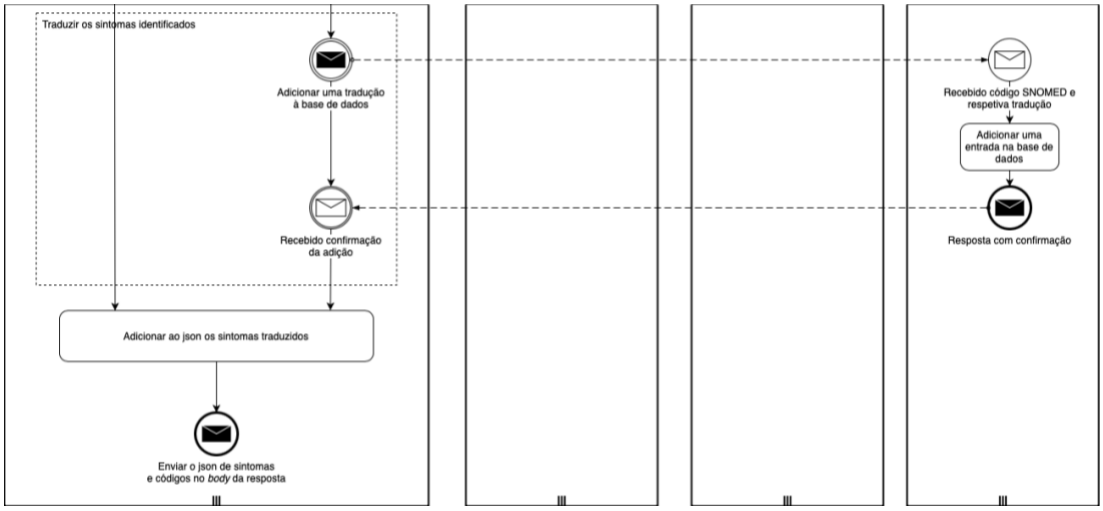
Apêndice D

D.1 Fluxo do processo referente à Lambda do campo "Subjetivo"



D.2 Fluxo do processo referente à Lambda do campo “Subjetivo” refletindo a otimização




Traduzir os sintomas identificados
Adicionar uma tradução à base de dados
Recebido confirmação da adição
Adicionar ao json os sintomas traduzidos
Enviar o json de sintomas e códigos no *body* da resposta
Recebido código SNOMED e respetiva tradução
Adicionar uma entrada na base de dados
Resposta com confirmação