

# Speech to Action, em ambiente empresarial - Conversão de informação verbal em ações contextuais

Nelson Palmas<sup>1</sup>, Rui Filipe Marques<sup>2</sup>

<sup>1 2</sup> Instituto Superior de Engenharia do Porto, Porto, Portugal

<sup>1</sup>[nelsonpalmas2009@gmail.com](mailto:nelsonpalmas2009@gmail.com), <sup>2</sup>[rfm@isep.ipp.pt](mailto:rfm@isep.ipp.pt)

**Resumo.** Muita da informação no ambiente empresarial é falada (analógica), desorganizada e complexa de tratar. A obtenção desta informação é trabalhosa de gerir e dada a erros. Nestes últimos anos, têm vindo a surgir no mercado vários assistentes de casa com diversas funções que incluem ações sobre comandos de voz. De modo a inovar a relação Pessoa-Máquina, a ideia base é a de aplicar o conceito de “assistente de voz”, ao ambiente empresarial e industrial, construindo para o efeito um conjunto de métodos e acessos a um sistema de Palavra-Ação, que tome partido de informação de uma dada conversa, e ofereça ao utilizador informação contextual dessa conversa. O assistente de voz selecionado foi o da Google. As razões da sua escolha serão abordadas no estado da arte. O sistema está preparado para perceber construções frásicas com principalmente 3 palavras associadas: produtos, encomendas, faturas. Através de agentes do DialogFlow (Machine Learning) foi possível associar o discurso a uma ação a despoletar pelo mesmo. Os testes efetuados tiveram uma boa percentagem de sucesso e mais informações sobre os mesmos serão debatidas posteriormente. A aplicação foi desenvolvida no contexto de prova de conceito. Este documento relata o seu franco potencial de bom funcionamento em ambiente profissional e empresarial.

**Palavras-Chave:** Interação Pessoa-Máquina, Comandos de voz, Sistema de Palavra-Ação, Speech-to-Action.

## 1 Introdução

O atual artigo descreve uma solução empresarial, que facilite o tratamento e consulta de faturas, encomendas e produtos de um qualquer cliente, integrado com um software *ERP* (Enterprise Resource Planning). A aposta contínua na eficiência, inovação e simplificação de tarefas empresariais, leva as organizações a pensar em novas estratégias e procedimentos. O intuito deste projeto foi o de agilizar tarefas e de converter sugestões de ação ou ações verbais em ações “*de facto*”. Normalmente, as atividades manuais realizadas no sentido de desempenhar uma ação demoram mais tempo, e podem originar em erros, ao automatizarmos tarefas, conseguimos chegar a um resultado final previsível, com menos erros, e, de uma forma geral, mais rapidamente. Com base neste pensamento, foi desenhado um *Sistema* constituído por uma *App* (Aplicação Móvel), um *Website* e um *ERP*. A *App* recebe o discurso do utilizador (que pode ocupar o papel de comercial ou agente de suporte) e, através desse

ISBN: 978-989-54758-6-5

discurso, apresenta num dashboard no *Website* (um tipo de painel de bordo), a ação executada pelo seu discurso, permitindo a manipulação das ações em estilo mãos-livres. Existem fundamentalmente duas vertentes nesta solução:

1. Speech To Text: Através do reconhecimento de voz pela *App*, conseguir a sua reprodução textual e, posteriormente, interpretá-la com a utilização do DialogFlow, ferramenta fundamental na interação máquina-homem e que faz uso do conceito de Machine Learning, através dos seus agentes.

2. Text To Action: Após o tratamento do texto ter sido feito, este é enviado da *App* para o servidor. No dashboard, a mensagem será escutada e executada, consoante o seu conteúdo.

A área de negócio na qual este projeto se focou, foi a área da ourivesaria. A evolução tecnológica leva a ponderar formas de modernizar as suas plataformas e de otimizar processos. Da otimização e automatização dos processos resulta uma redução de falhas humanas e um aumento da eficiência. Neste contexto, o método escolhido neste projeto, por proximidade tecnológica e simplicidade de implementação, foi o ambiente mobile. A *App* foi criada de forma ser intuitiva e simples, agilizando processos de login, permitindo ampla usabilidade e acessibilidade no contexto empresarial.

Para que a *App* funcione, necessita de uma API (Application Programming Interface) de um assistente de voz. Uma API consiste num conjunto de ferramentas e blocos de código fornecidos para utilizar no nosso sistema. São definidas funcionalidades independentes da implementação de forma a ser compatível e fácil de integrar num programa [1]. No ponto seguinte será abordado mais detalhadamente quais os assistentes estudados e, por fim, considerados. Os assistentes de voz têm um papel preponderante na implementação desta solução, pois estão encarregues por transformar o input (voz) em output (texto). O objetivo deste tipo de *Sistema* é o de providenciar suporte no atendimento ao cliente, com a intenção de melhoria de nível de serviço, em termos de qualidade, relevância e velocidade da informação resultante. Nesse sentido, investir em apps, como esta, é investir na satisfação do cliente, na melhoria da relação da empresa com os seus diferentes públicos e potenciar o aumento de produtividade e competitividade. [2]

A última versão da aplicação foi desenvolvida em Java, por ser das linguagens mais usadas [3] e com maior documentação. Como tal há a limitação de apenas ser desenvolvida para Android. A expansão da *App*, de forma ser compatível com iPhone é inquestionavelmente uma melhoria. Este ponto podia ser concretizado migrando de Java para *Ionic* (que juntamente com *Cordova* permite uma linguagem de programação de uso híbrido – Android e iOS - e com diversas bibliotecas pertinentes que se adaptam ao caso prático como a “cordova-plugin-audioinput” – responsável por obter o input do microfone e a “cordova-plugin-api” – responsável por fazer a integração com o DialogFlow). A arquitetura do *Sistema* desenvolvido será posteriormente exposta neste artigo, assim como a sua implementação.

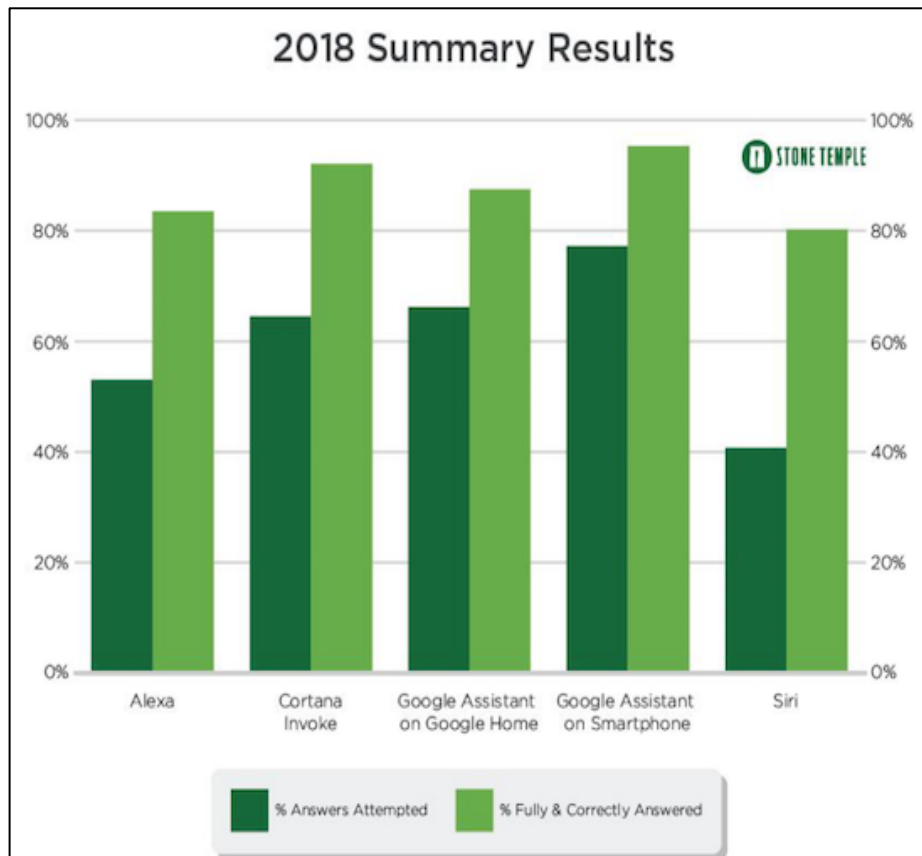
O impacto da aplicação foi estimado através da realização de questionários, com o intuito de avaliar o *Website* e a *App*. O resultado dos mesmos mostrou um impacto positivo na celeridade dos processos (produtos, encomendas e faturas), assim como permitiu obter um bom grau de satisfação da população amostra.

## 1.1 Assistentes de voz

Atualmente, os três assistentes de voz mais populares são a Alexa (Amazon), a Assistente da Google (Google) e a Siri (Apple).

A agência Stone Temple realizou um estudo profundo em 2018 [4] onde foram efetuadas 4942 questões, gastas 80 horas de pesquisa, avaliando cinco API's: Alexa, Cortana Invoke, Google Assistant on Google Home, Google Assistant on a Smartphone, Siri. O objetivo desta investigação foi o de expor os pontos fracos e fortes de cada uma.

Os resultados, representados na Figura 1, mostram que a Google Assistant no smartphone superou significativamente todos os outros assistentes de voz em termos de tentativas de resposta a perguntas e respostas corretas. Este é um aspeto crítico no desenvolvimento da *App*, pois uma má interpretação de qualidade ou quantidade de informação num processo comercial, pode ser danoso tanto para a empresa como para o seu cliente, tornando em imperativa, a necessidade de assertividade no conteúdo obtido.



**Fig. 1.** Comparação do número de perguntas respondidas entre 2017 e 2018 [4]

## 1.2 Arquitetura e tecnologias usadas

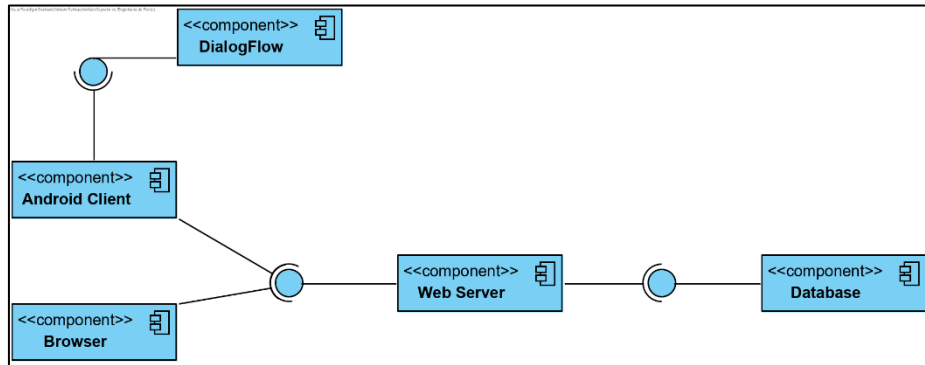


Fig. 2. Diagrama de Componentes do Sistema

No Sistema existem, fundamentalmente cinco componentes: o cliente android (*App*), o browser, o servidor web, a base de dados (que compõem o *Website*), e a *API* do DialogFlow (*API*), como pode ser observado na Figura 2.

O primeiro componente referido (*App*) necessita de comunicar com o servidor web em vários casos, através de pedidos HTTP. Além disso, precisa, igualmente, de fazer pedidos HTTP (POST e GET) à *API* do DialogFlow.

Por sua vez, o servidor necessita de comunicar com a base de dados de forma a consultar ou modificar dados. Esta comunicação é feita através de queries SQL.

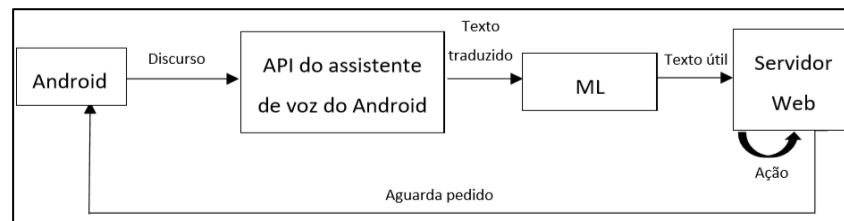
Para a concretização deste estudo, foram utilizadas sete tecnologias. Ao nível da *App*, o seu core foi construído em Java, com a integração da *API* do DialogFlow, de forma a traduzir o discurso feito pelo utilizador num discurso “útil”. Ao nível do *Website* a linguagem de programação escolhida foi o PHP. HTML e Bootstrap, framework utilizada para o desenvolvimento web responsivo, foram usadas naquilo que é a disposição gráfica do conteúdo. Para o backend do *Website* foi utilizado JavaScript, para incluir bibliotecas como o AJAX (Asynchronous JavaScript And XML) e jQuery, de forma a poder trabalhar com informação em real-time e preencher a informação no *Website*. O AJAX é uma técnica de desenvolvimento para a criação de aplicações web interativas. A intenção é fazer com que as páginas web se sintam mais responsivas através da troca de pequenas quantidades de dados com o servidor, para que a página web inteira não tenha de ser recarregada cada vez que o utilizador solicita uma alteração. Isto destina-se a aumentar a interatividade, velocidade e usabilidade da página web. O jQuery é uma biblioteca JavaScript open-source que simplifica a interação entre HTML e JavaScript. É ideal para prototipagem, é completamente discreta, utiliza CSS e tem um comportamento fácil de separar. [5]

## 2 Análise e desenho da solução

### 2.1 Domínio do problema

Esta solução, apresentada na Figura 3 (abaixo), consiste em receber um discurso através de uma aplicação móvel, com acesso ao microfone e à internet, utilizando a API da Google Assistant de forma a traduzir o discurso em texto. Depois, deste texto era necessário retirar o fundamental para desencadear a ação. Isso foi possível através do

uso do DialogFlow e dos seus agentes de Machine Learning, obtendo o texto útil. Este era posteriormente enviado para o servidor Web que executava as ações consoante o que foi pedido no discurso e o utilizador podia acompanhar tais ações num dashboard online.



**Fig. 3.** Esquema da solução implementada

Um cenário deste *Sistema* seria o de adicionar produtos a uma encomenda em aberto. O utilizador deve fazer o login no *Website* e depois usar o código de emparelhamento que lhe foi dado para se autenticar na *App*. Depois, o utilizador faria um pedido verbal como “Adicionar 2 produtos 412HZ à encomenda 1”, sendo “2” a quantidade a encomendar, “412HZ” a referência do produto e “1” o número da encomenda. Após ter feito o discurso, este é interpretado pelos agentes do DialogFlow e é feita uma query à base de dados onde é criada uma linha com a informação transmitida (quantidade, número do produto e número da encomenda). Por último é dado um alerta ao utilizador em como os produtos foram adicionados à encomenda.

## 2.2 Desenho da solução implementada

A solução desenvolvida segue uma arquitetura 3-Tier Cliente/Servidor.

O modelo Cliente/Servidor separa os processos em plataformas independentes que interagem, permitindo que os recursos sejam compartilhados enquanto se obtém o máximo de benefício de cada dispositivo diferente. A arquitetura 3-Tier é constituída por três camadas: a camada do cliente, a do servidor e a da base de dados. A Tabela 2 apresenta as vantagens e as desvantagens do uso desta arquitetura.

**Tabela 1.** Vantagens e Desvantagens da arquitetura 3-Tier

Vantagens	Desvantagens
<ul style="list-style-type: none"> <li>- É uma estrutura simples. Ter 3 níveis faz com que não haja dificuldade em compreendê-la</li> <li>- Sendo uma estrutura simples também se torna fácil de configurar e de fazer manutenção</li> <li>- Permite que equipas de desenvolvimento diferentes trabalhem em cada uma das suas próprias áreas de especialização</li> </ul>	<ul style="list-style-type: none"> <li>- A separação física entre o servidor e base de dados pode afetar moderadamente o desempenho</li> </ul>

Com base neste estilo arquitetural, foram construídos 3 componentes: o cliente (android ou browser), o servidor web e a base de dados.

O primeiro componente referido necessita de comunicar com o servidor web em vários casos, através de pedidos HTTP (GET e/ou POST). Por exemplo, para verificar o código de emparelhamento e para enviar a mensagem do utilizador para o servidor. Por sua vez, o servidor necessita de comunicar com a base de dados de forma a consultar ou modificar dados. Esta comunicação é feita através de queries SQL.

### 3 Implementação da solução

#### 3.1 Descrição da implementação

##### 3.1.1 Website

O *Website* foi criado com a linguagem de programação PHP. As principais áreas são o Login e a Ação do discurso.

O Login tinha HTML embebido no ficheiro PHP, pois necessitava de ter uma componente visual. A construção do código HTML fez-se juntamente com a framework Bootstrap. Esta ferramenta fez com que não fosse necessário criar código CSS e tornou o sistema responsivo.

Em algumas áreas, como as Encomendas, implementou-se AJAX e jQuery, de forma a tirar partido de ações em tempo real. Para que os dados de negócio pudessem ser atualizados devido a alterações feitas pelos pedidos do utilizador, foi usado jQuery. O AJAX foi utilizado para escutar periodicamente pedidos do utilizador com alterações aos dados.

A informação nas tabelas apresentadas ao utilizador é obtida através da execução de código associado à área da Ação do discurso. Inicialmente, este ficheiro começa por escolher a ação que vai executar. Existem 5 ações possíveis: “produtos”, “encomendas”, “faturas”, “encomenda” e “fatura”. Cada ação implica a execução de queries à base de dados cujo retorno será código HTML ou campos simples, apresentados no browser.

##### 3.1.2 App

A *App* foi desenvolvida em Java, usando como IDE (Integrated Development Environment) o Android Studio. Esta necessita de ter acesso à Internet e ao Microfone. O seu objetivo era o de o utilizador poder fazer um discurso para a mesma e esta poder captá-lo, traduzi-lo e enviá-lo para o servidor, para depois este executar a ação correspondente.

Para poder fazer uso da aplicação, o utilizador necessita de estar autenticado. Para tal, seria necessário a introdução de 4 dígitos no formato OTP (One-Time-Password). Optou-se por esta estratégia por ser mais prática e popular, como pode ser observado na figura seguinte. A realocação do hardware mobile a qualquer posto tem como principal vantagem a portabilidade e a facilidade na realização de processos.

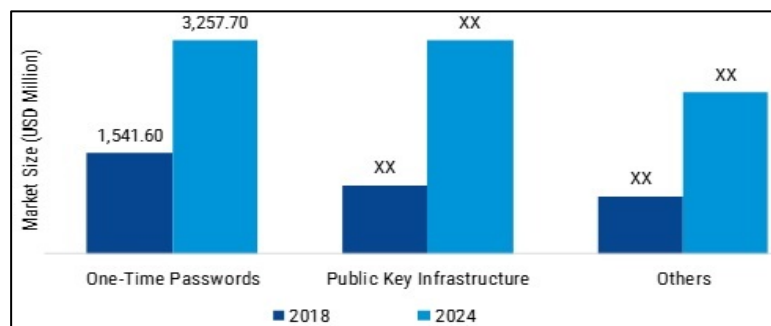


Fig. 4. Market share de OTP [6]

Uma vez feito o login no *Website*, o utilizador pode consultar um código de 4 dígitos na página principal e pode utilizá-lo para se autenticar e emparelhar com a *App*. Esse código será usado para fazer a autenticação na *App*. Com a versão final deste projeto deveria ser possível obter o papel (role) do utilizador através do *ERP* e, com base nele, permitir que ele fizesse uma navegação na *App*, consoante as suas permissões.

O login e a captação do discurso na *App* são *AsyncTasks*. Uma *AsyncTask* é uma classe abstrata de Android que ajuda as aplicações Android a lidar de forma eficiente com a Main UI (User Interface) thread. A classe *AsyncTask* permite executar tarefas/operações em segundo plano e mostrar o resultado na thread da UI sem afectar o comportamento da thread principal. Estas foram concebidas para permitir uma utilização adequada e fácil da thread do UI e devem ser usadas para operações curtas.

### 3.1.3 *API* do DialogFlow

A captação do discurso é feita através da API da Google Assistant, que transforma a voz em texto. A tradução do discurso acontece graças à integração da *API* do DialogFlow. Nesta plataforma foi construída uma base de dados com inputs, semelhantes aos dos utilizadores, que seriam traduzidos num conjunto de palavras-chave. Os seus agentes seriam os responsáveis por escolher as melhores palavras-chave, com base no input.

## 3.2 Testes e resultados obtidos

Para avaliar a solução criada, foram feitos medidos tempos e precisão de resposta.

Os tempos de resposta foram medidos desde que o utilizador acaba de efetuar o discurso até ser notificado que uma ação foi efetuada. Foram feitos 288 pedidos, POST e GET. A média de tempo de resposta foi de 1,08 segundos. O que se verificou foi que à medida que a base de dados ia ficando mais composta, os tempos iam aumentando ligeiramente. É, portanto, espectável que, quando existir uma quantidade de dados considerável na base de dados, os tempos de respostas aumentem gradualmente. A tradução do discurso e a precisão da resposta foram muito boas. Dos 288 pedidos, apenas 14 foram mal traduzidos, o que se revela numa percentagem de 4,86%. A precisão do discurso irá sempre depender de variáveis como a qualidade do microfone e do ruído ambiente. Os testes foram efetuados em ambientes fechados, sem cuidado acústico, mas com pouco ruído, embora contextualizado num ambiente empresarial. Sendo assim, como melhoria, dever-se-ão fazer mais testes em ambientes mais ruidosos e com a introdução de outras condicionantes como por exemplo, uma menor receção de rede Wi-Fi.

## 4 Conclusões

Este projeto focou-se em desenvolver uma solução modular que permitisse a expansibilidade da ferramenta e uma fácil integração com as restantes aplicações.

O *Sistema* foi concluído com sucesso embora numa versão embrionária e em ambiente de PoC (Prova de Conceito). Neste contexto, esta versão permitiu avaliar a possível aplicação em produção desta nova metodologia de trabalho. A interligação ao *ERP* necessita de um trabalho mais denso de integração, com uso dos recursos existentes, que não foi o âmbito deste projeto. Ao nível da *App*, poder-se-ia ter optado por programar com Ionic, o que tornaria possível aos utilizadores com iPhone usufruir

da mesma. No entanto, o Java foi a linguagem escolhida por ser familiar ao desenvolvedor. O uso de uma framework PHP, como Laravel, em vez de programação “pura” em PHP, teria facilitado e organizado a codificação do *Website* e do servidor. A escolha de não usar uma framework PHP residiu no facto do desenvolvedor já ter tido contacto com PHP “puro”. A integração com a *API* do DialogFlow foi uma mais valia para este projeto e, cada vez mais, a aplicação de conceitos como Inteligência Artificial e Machine Learning, é necessária. Numa versão seguinte desta PoC, poderá ser adicionado contexto de conversa comercial/empresarial, opções de navegação, adição de informação de suporte ao utilizador ou até mesmo ações de texto livre, entre outros melhoramentos.

Em suma, para que se torne numa aplicação de negócio, é necessário melhorar alguns aspetos como: segurança, autenticação e interface com o utilizador, além da realização dos testes necessários.

## Referências

- [1] K. Hanif e D. Bosia, *Design Engineering Refocused*, Wiley, 2016 .
- [2] WayNext, “A PESQUISA POR VOZ É INCONTORNÁVEL NA ESTRATÉGIA DE OTIMIZAÇÃO DE SITES PARA OS MOTORES DE BUSCA,” [Online]. Available: <https://www.waynext.com/waytrends/pesquisa-por-voz-na-otimizacao-para-motores-de-busca-seo/>. [Acedido em 1 Junho 2019].
- [3] B. Putano, “A Look At 5 of the Most Popular Programming Languages of 2019,” 30 Agosto 2019. [Online]. Available: <https://stackify.com/popular-programming-languages-2018/>.
- [4] B. Kinsella, “Google Assistant is Still the Smartest Voice Assistant According to New Study,” voicebot.ai, 24 Abril 2018. [Online]. Available: <https://voicebot.ai/2018/04/24/google-assistant-is-still-the-smartest-voice-assistant-according-to-new-study/>. [Acedido em 5 Junho 2019].
- [5] G. Cornelia, R. Gyorodi, G. Pecherle, L. Tamas e A. Roşu, “Web 2.0 Technologies with jQuery and Ajax,” *Journal of Computer Science and Control Systems*, 2009.
- [6] Market Research Future, “ Two-Factor Authentication Market,” Agosto 2019. [Online]. Available: <https://www.marketresearchfuture.com/reports/two-factor-authentication-market-3772>.
- [7] Developers Android, “AsyncTask,” [Online]. Available: <https://developer.android.com/reference/android/os/AsyncTask>. [Acedido em 7 Junho 2019].