Desenvolvimento Front-End e Back-End de aplicações que solucionam problemas reais com a integração de Inteligência Artificial

Afonso Filipe Garcia Aires

(Licenciado em Engenharia Informática e Multimédia)

Relatório de Estágio para obtenção do grau de Mestre em Engenharia Informática e Multimédia

Orientadores:

Doutor Rui Jesus

Licenciado Pedro Duarte

Júri:

Presidente: Doutor Carlos Gonçalves

Vogais: Doutor Pedro Santos

Doutor Rui Jesus

Outubro de 2024

Desenvolvimento Front-End e Back-End de aplicações que solucionam problemas reais com a integração de Inteligência Artificial

Afonso Filipe Garcia Aires
(Licenciado em Engenharia Informática e Multimédia)

Relatório de Estágio para obtenção do grau de Mestre em Engenharia Informática e Multimédia

Orientadores:

Doutor Rui Jesus, ISEL

Licenciado Pedro Duarte, Make The Shift

Júri:

Presidente: Doutor Carlos Gonçalves, ISEL

Vogais: Doutor Pedro Santos, ISEL

Doutor Rui Jesus, ISEL

Outubro de 2024

Declaração de integridade

Declaro que esta(e) dissertação / trabalho de projeto / relatório de estágio é o resultado da minha investigação pessoal e independente. O seu conteúdo é original e todas as fontes listadas nas referências bibliográficas foram consultadas e estão devidamente mencionadas no texto. Mais declaro que todas as referências científicas e técnicas relevantes para o desenvolvimento do trabalho estão devidamente citadas e constam das referências bibliográficas.

Assinado por: Afonso Filipe Garcia Aires
Identificação: BI15662236
Data: 2024-09-30 às 14:58:51

_____

Lisboa, …… de ……. de ……

Resumo

Nos dias que correm a evolução tecnológica avança exponencialmente, e nunca antes a necessidade de abstração foi tão crítica para acompanhar esse ritmo. Com o aumento da entropia dos sistemas informáticos, através da sua crescente complexidade e volume de informações a processar, é fundamental que as tecnologias se tornem mais acessíveis, para que o foco esteja na lógica e nas soluções, enquanto as máquinas assumem a gestão dos detalhes técnicos. O contínuo avanço da inteligência artificial, acentua cada vez mais este padrão, caminhando para um cenário onde o papel humano se encarrega de orientar e ajustar o comportamento das máquinas, enquanto estas assumem tarefas técnicas e decisivas de forma cada vez mais autónoma.

Importante é o papel de empresas inovadoras como a Make The Shift, que se foca no desenvolvimento de soluções aplicacionais utilizando a plataforma *low-code* Microsoft Power Platform, por forma a impulsionar a sua investigação e desenvolvimento. A Make The Shift pretende ainda superar as limitações das tecnologias *low-code* com a integração de tecnologias tradicionais. Esta abordagem visa aumentar a sua eficiência no desenvolvimento de software, garantindo soluções robustas que melhoram a satisfação dos clientes.

No âmbito do estágio, serão realizadas aplicações e automatizações utilizando a plataforma *low-code* da Microsoft e integrando tecnologias tradicionais. Com a tecnologia Power Apps será realizada uma aplicação móvel para gestão de cartões de negócio digitais, com a integração de serviços da plataforma Microsoft Azure, como Azure Functions e Azure Web Apps. Realizar-se-á ainda a automatização para criação dinâmica de relatórios, através das tecnologias Power Automate, HTML e CSS. Também irá proporcionar a oportunidade de explorar a integração de inteligência artificial, enquanto se automatiza o agendamento periódico de reuniões dentro da empresa. Por fim, será desenvolvido um sistema dinâmico de notificações e gestão de tarefas utilizando que pode ser administrado através da plataforma SharePoint.

Palavras chaves: Microsoft Power Platform, Low-code, Aplicações Web, Automação de Processos, Inteligência Artificial

Abstract

Nowadays, technological evolution is advancing exponentially, and never before has the need for abstraction been so critical in order to keep up with this pace. With the increasing entropy of computer systems, through their growing complexity and volume of information to process, it is essential that technologies become more accessible, so that the focus is on logic and solutions, while machines take over the management of technical details. The continuous progress of artificial intelligence increasingly accentuates this pattern, moving towards a scenario where the human role is responsible for guiding and adjusting the behavior of the machines, while they take on technical and decisive tasks in an increasingly autonomous way.

Important is the role of innovative companies such as Make The Shift, which focuses on developing application solutions using *low-code* platform Microsoft Power Platform, in order to boost their research and development. Make The Shift also aims to overcome the limitations of *low-code* technologies by integrating traditional technologies. This approach aims to increase its efficiency in software development, guaranteeing robust solutions that improve customer satisfaction.

As part of the internship, applications and automations will be carried out using Microsoft's *low-code* platform and integrating traditional technologies. Using Power Apps technology, a mobile application will be created for managing digital business cards, integrating services from the Microsoft Azure platform, such as Azure Functions and Azure Web Apps. There will also be automation for the dynamic creation of reports, using Power Automate, HTML and CSS technologies. It will also provide the opportunity to explore the integration of artificial intelligence, while automating the periodic scheduling of meetings within the company. Finally, a dynamic notification and task management system will be developed using the SharePoint platform.

Keywords: Microsoft Power Platform, Low-code, Web Applications, Process Automation, Artificial Intelligence

Agradecimentos

Primeiramente, agradeço ao meu orientador da empresa Make The Shift, Pedro Duarte, pela orientação valiosa, pela paciência e pelo apoio constante ao longo deste período. As suas orientações foram fundamentais para o desenvolvimento das soluções aplicacionais e para minha evolução profissional.

Agradeço também aos membros da equipa da Make The Shift, que me receberam de forma calorosa e colaboraram diretamente no desenvolvimento dos projetos. Agradeço particularmente à Vanda Aragão, pela sua compreensão e suporte contínuo durante o estágio.

Sou grato ao Instituto Superior de Engenharia de Lisboa, que me acolheu desde a licenciatura até ao fim do mestrado, pela oportunidade de realizar o estágio e pelo apoio fornecido ao longo deste percurso. A todos os docentes pela partilha da sua paixão pela área, aos colegas pela inclusão e apoio, e a toda a infraestrutura pelos recursos disponibilizados, foram essenciais para o sucesso dos projetos e para minha formação. Em particular, quero agradecer ao coordenador do mestrado em que se insere o estágio e o meu orientador por parte da instituição de ensino, Rui Jesus, pelos seus ensinamentos, ideias e suporte inabalável durante todo o processo.

Agradeço ainda aos meus pais, irmã, avós, tios, namorada e amigos pelo apoio, incentivo e compreensão durante todo o processo. A sua confiança em mim foi uma fonte constante de motivação e encorajamento.

Finalmente, agradeço a todos que, de alguma forma, contribuíram para a minha jornada académica e profissional. O apoio e a colaboração de cada um foi imprescritível para a conclusão deste estágio e para o desenvolvimento deste relatório.

Índice de Conteúdos

Índice de Figuras

Lista de Acrónimos

API Application Programming Interface. xi, 15, 18, 51, 56–60, 72, 78, 113, 123

CSS Cascading Style Sheets. v, 2, 6, 16, 18, 19, 41, 49, 51–54, 85, 87, 94, 122, 123

HTML Hypertext Markup Language. v, ix, xiii, 2, 6, 16–19, 41, 45, 49, 51–53, 55, 60–64, 66, 67, 71, 77, 84, 85, 87–90, 92–94, 96–100, 118, 122, 123

HTTP Hypertext Transfer Protocol. viii, ix, xii, xiii, 14, 15, 20, 40, 50, 62, 74–79, 112–114, 122

HTTPS Hypertext Transfer Protocol Secure. xiii

JSON JavaScript Object Notation. 12, 75–77

MVC Model-View-Controller. viii, 3, 56, 57, 60–62, 78, 122

PDF Portable Document Format. ix, 12, 19, 27, 28, 45, 87, 89, 94, 96–100

PNG Portable Network Graphics. viii, 74, 75, 77

QR Quick Response. viii, 25, 40, 41, 50, 51, 74–77, 79, 80, 83, 122

SDLC Software Development Life Cycle. 31

SVG Scalable Vector Graphics. 18, 75

UAT User Acceptance Testing. 33, 34, 38, 39, 45, 80, 100

XHTML Extensible Hypertext Markup Language. 18

XML eXtensible Markup Language. 12, 16–19, 91

XPath XML Path Language. ix, 13, 91

ZIP Zone Improvement Plan. 94

Glossário

- **API** - sigla em inglês para Application Programming Interface, mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos.
- **App** - abreviatura de "Aplicação", um programa de software que visa facilitar a realização de uma tarefa através do dispositivo de um utilizador.
- **Backend** - parte da aplicação que não está visível para o utilizador mas é indispensável. Esta é responsável pela lógica, processamento de dados, comunicação com o servidor e dá apoio às ações do utilizador.
- **Boleano** - em ciências da computação, é um tipo de valor lógico que pode ter um de dois valores possíveis: verdadeiro ou falso.
- **Dropdown** - é um elemento de uma interface de utilizador, semelhante a uma lista. Quando o utilizador interage com este elemento, o elemento expande-se para apresentar várias opções, permitindo ao utilizador escolher uma.
- **Feedback** - expressão inglesa que significa reação, retorno ou avaliação. Termo que se refere a informações ou opiniões fornecidas como resposta a uma ação ou desempenho.
- **Framework** - coleção de componentes e ferramentas de software reutilizáveis que que facilita e agiliza o desenvolvimento de novas aplicações.
- **Frontend** - parte da aplicação que compõe a interface gráfica visível para o utilizador, onde este pode interagir.
- **Hardware** - componentes eletrônicos e físicos constituintes de um computador, como processador, placa-mãe, memória RAM, placa de vídeo, cabos internos, fonte, entre outros. Estes, ao se conectarem, fazem o equipamento funcionar.

- **Hiperligação** - Elemento num documento que permite a navegação para outro documento ou recurso na Internet, ou para uma seção específica dentro do mesmo documento.
- **HTTP** - abreviatura de “Hypertext Transfer Protocol” descreve um protocolo de comunicação utilizado para a troca de informações na Web, permitindo a transmissão de dados entre servidores e clientes.
- **Input** - expressão da língua inglesa que significa entrada.
- **Interface de utilizador** - espaço de interação entre o humano e a máquina, onde o utilizador pode enviar comandos e receber respostas, facilitando a comunicação e operação do sistema.
- **Link** - abreviatura para *Hiperlink*. Elemento num documento que permite a navegação para outro documento ou recurso na Internet, ou para uma seção específica dentro do mesmo documento.
- **Low-code** - filosofia digital que permite a qualquer pessoa criar aplicações e programas sem a necessidade de saber programação.
- **Mockups** - representações visuais estáticas, próximos ao aspecto final do projeto, geralmente em tamanho real, utilizados para apresentar e avaliar o seu design e *layout*.
- **Navegador** - aplicação informática para visualização de páginas Web e navegação na Internet.
- **Output** - expressão da língua inglesa que significa saída.
- **Serverless** - modelo de computação em que a gestão da infraestrutura de backend é abstraída dos programadores. A própria plataforma providencia um ambiente escalável e flexível, automaticamente ajustando os recursos conforme a demanda.
- **Sintaxe** - em ciências da computação, é o conjunto de regras que determina quais combinações de símbolos e palavras-chaves que podem ser utilizadas numa linguagem de programação, definindo a estrutura correta das expressões e comandos.
- **Sistema** - conjunto de componentes relacionados e interdependentes, criado para atingir um objetivo específico, com uma operação que é mais eficaz do que a soma de seus componentes.
- **Site** - nome mais curto utilizado para referir Website.

- **Software** - conjunto de programas e dados com instruções que como o computador deve executar tarefas específicas.
- **String** - sequência de caracteres, geralmente utilizada para representar palavras, frases ou textos de um programa.
- **Substring** - parte ou segmento de uma string.
- **Tags** - palavras-chave ou etiquetas utilizadas para categorizar e organizar informações, facilitando a busca e o gestão de dados em diversos contextos, como por exemplo em HTML ou sistemas de gestão de conteúdos.
- **Template** - expressão na língua inglesa que significa padrão ou molde. Em computação, refere-se a um modelo predefinido utilizado para criar documentos, páginas ou outros conteúdos de maneira consistente e eficiente.
- **Tenant** - expressão na língua inglesa para inquilino. No contexto das ciências da computação representa um grupo de utilizadores que partilham o acesso com privilégios específicos para uma instância de um sistema.
- **Trigger** - expressão na língua inglesa que significa desencadear algo. Geralmente, refere-se a um mecanismo que inicia automaticamente uma ação ou processo em resposta a um evento específico.
- **Web** - nome mais curto utilizado para referir World Wide Web.
- **Website** - conjunto de páginas na rede conectadas por hipertextos, acessíveis na Internet geralmente através dos protocolos HTTP ou HTTPS. HTTPS é a versão segura do HTTP, utiliza criptografia para proteger a comunicação entre o navegador e o servidor.
- **Wireframes** - representações visuais básicas de uma interface de utilizador, utilizadas no processo de design para ajudar a definir a disposição dos elementos, sem preocupação com os detalhes do design.
- **World Wide Web** - Refere-se a um sistema de documentos interligados por hipertexto e hipermédia, acessíveis na Internet através do navegador.

Capítulo 1

Introdução

Este capítulo explora a motivação e o contexto do estágio, destacando os avanços rápidos nas tecnologias *low-code* e a crescente demanda por soluções rápidas e flexíveis no desenvolvimento de aplicações. Este capítulo, proporciona ainda ao leitor uma descrição da empresa em que se insere o estágio e a compreensão dos objetivos deste, oferecendo uma visão da necessidade da integração de tecnologias de desenvolvimento tradicionais nas soluções *low-code* da empresa. Por fim, o capítulo apresenta a estrutura detalhada do documento, permitindo ao leitor entender como o relatório está organizado e como cada secção contribui para a compreensão global do estágio.

1.1 Motivação

Este estágio surgiu em resposta da crescente demanda por digitalizar os processos manuais de forma inovadora e eficiente, tanto em aplicações de computador quanto móveis. Com o aumento da necessidade de soluções rápidas e flexíveis, a adoção de plataformas de software *low-code* tem crescido significativamente. De acordo com um relatório da Forrester [1], a utilização de tecnologias *low-code* aumentou cerca de 21% desde 2019, refletindo uma tendência crescente no mercado onde a agilidade e a flexibilidade são altamente valorizadas. A integração destas plataformas com inteligência artificial está a acelerar cada vez mais esta rápida ascensão, comenta a Gestora de Marketing de Produtos da empresa OutSystems [2].

Durante a formação no curso, adquire-se competências em diversas áreas, incluindo desenvolvimento full stack (front-end e back-end), análise de bases de dados, inteligência artificial, aprendizagem automática e interação pessoa-máquina. Contudo, não proporciona a oportunidade de trabalhar com tecnologias da Microsoft, em particular com a sua plataforma *low-code* Microsoft Power Platform nem com a sua plataforma de computação na nuvem Mi-

crosoft Azure, incluindo com as suas ferramentas de desenvolvimento Web.

O estágio na empresa Make The Shift proporciona não só a oportunidade de explorar estas novas tecnologias como de integrar os conhecimentos, tanto em desenvolvimento de aplicações como em inteligência artificial, adquiridos ao longo do curso com as soluções *low-code* desenvolvidas pela empresa. Este é um dos aspetos que mais motiva a realização deste estágio.

A aplicação de conhecimentos especializados permite contribuir significativamente para a empresa, ajudando a desenvolver soluções inovadoras que a destacam no mercado das tecnologias *low-code* de forma criativa.

1.2 Contexto

O estágio foi realizado na empresa Make The Shift, uma Microsoft Golden Partner com experiência em desenvolver e entregar soluções *low-code* que atendem às necessidades específicas dos seus clientes. A equipa na qual estive inserido é responsável pelo desenvolvimento de soluções tecnológicas utilizando as tecnologias da Microsoft 365. Este ambiente proporcionou uma oportunidade única de aplicar e integrar conhecimentos adquiridos durante o curso de Engenharia Informática e Multimédia, com a plataforma de desenvolvimento *low-code* Microsoft Power Platform e a plataforma de computação na nuvem Microsoft Azure.

A empresa utiliza a plataforma Power Apps, para desenvolver aplicações de computador e móveis, assim como a plataforma Power Automate para automação de processos. Todavia, a empresa pretende tornar as suas soluções mais flexíveis através da integração de tecnologias Web tradicionais como HTML, CSS, Javascript, assim como recorrer à plataforma Microsoft Azure para criar e gerir páginas Web de forma a mostrar e partilhar eficientemente conteúdos. A colaboração com profissionais experientes também permitiu a expansão de competências técnicas e a capacidade de enfrentar desafios práticos, essenciais para a adaptação ao contexto profissional.

1.3 Empresa - Make The Shift

A empresa Make The Shift, uma Microsoft Golden Partner, tem desenvolvido e entregue soluções para empresas de mais de 20 países, utilizando as tecnologias da Microsoft 365, incluindo a plataforma *low-code* Microsoft Power Platform. Através da criação de aplicações móveis e de desktop, a empresa atende às mais diversas necessidades dos seus clientes,

garantindo um elevado grau de excelência nas suas soluções. Isso reforça o seu prestígio no mercado e contribui para a expansão do portfólio de clientes, fortalecendo a sua presença global.

O desenvolvimento das aplicações utilizando as tecnologias habituais da empresa e a integração, nas suas soluções, de tecnologias tradicionais e inteligência artificial permite oferecer soluções inovadoras que a destacam no mercado das tecnologias *low-code*. Esta abordagem criativa garante não só que a empresa se mantenha atualizada nas suas soluções, como também aumenta a satisfação dos clientes, ao assegurar que as entregas estejam alinhadas com as exigências modernas e as necessidades específicas de cada projeto.

1.4 Objetivos

O principal objetivo do estágio é o desenvolvimento de soluções aplicacionais utilizando a plataforma Microsoft Power Platform, com ênfase nas tecnologias Power Apps e Power Automate, integradas com as tecnologias Web tradicionais HTML e CSS. Para além disso tem-se como objetivo expandir as capacidades dessas soluções através da plataforma Azure App Service, nomeadamente a utilização das Azure Web Apps para suportar a criação e gestão de aplicações Web e as Azure Functions que permitem correr trechos de código através de eventos. Ao longo do estágio, várias tarefas surgiram para atender às necessidades específicas dos clientes da empresa, proporcionando as melhores soluções possíveis.

Em detalhe, os objetivos deste estágio são distribuídos em cinco projetos distintos:

Aplicação de Gestão de Cartões de Negócio Digitais - Cartões Horizontais

A aplicação móvel foi desenvolvida em Power Apps num momento anterior ao estágio. O foco do estágio são as caraterísticas complementares à aplicação. Especificamente, é necessário a digitalização dos cartões digitais de negócios do cliente em HTML e CSS. É necessário ainda o desenvolvimento de uma aplicação Web, *front-end* e *back-end*, através da plataforma Azure Web Apps, visando um modelo MVC, para a apresentação dos cartões virtuais sem a necessidade de instalar a aplicação. Através desta aplicação terá de ser possível o *download* das informações do cartão digital no formato vCard. É essencial o desenvolvimento de algoritmos para apresentar as informações nos cartões digitais de forma dinâmica, na linguagem de back-end do Website C# e na linguagem típica das aplicações em Power Apps, Power Fx. A integração na aplicação móvel de um código QR, através dos serviços das Azure Functions, para facilitar a partilha dos cartões.

Aplicação de Gestão de Cartões de Negócio Digitais - Cartões Verticais

Nesta aplicação é necessário uma cópia da aplicação anterior e o desenvolvimento totalmente livre dos *templates* dos cartões de negócios digitais, num formato vertical, incluindo todo o processo de design e implementação, para a demonstração da aplicação a um novo cliente.

Aplicação de Gestão de Auditorias

A aplicação de gestão de auditoras foi realizada em Power Apps num momento anterior ao contexto do estágio. Esta visa facilitar a realização de auditorias, permitindo aos utilizadores no terreno registar ocorrências, através de observações, comentários e imagens, e partilhar essas informações com contatos através de relatórios. O estágio concentra-se na caraterística de criação automática de relatórios dinâmicos com os tópicos relatados na aplicação, através de automatizações, utilizando a ferramenta de fluxos de trabalho Power Automate e com recurso às tecnologias HTML e CSS.

Automatização de Agendamento de Reuniões Internas

O estágio objetiva continuar a utilizar a tecnologia Power Automate, desta vez para realizar a automatização do agendamento de reuniões semanais internas na empresa, sem o trabalho repetitivo de um agendamento manual. Esta solução propõe ainda a investigação da integração de inteligência artificial na concepção do fluxo de trabalho.

Sistema Dinâmico de Gestão Notificações e Tarefas

A última aplicação no âmbito do estágio tem como objetivo concretizar um sistema dinâmico de notificações e de criação e gestão de tarefas através de eventos. Esta solução visa ser desenvolvida através de automatizações, utilizando a tecnologia Power Automate e terá uma interface em Sharepoint, onde são geridos os templates das notificações enviadas e tarefas a serem criadas e concluídas, assim como os eventos que acionam estas ações.

Outro objetivo importante é o aperfeiçoamento de competências em metodologias ágeis, aplicada num contexto empresarial. Esta abordagem foca-se na melhoria contínua e na adaptação às mudanças de requisitos dos clientes, otimizando o ciclo de desenvolvimento e garantindo a entrega de um produto final mais robusto.

Além do conhecimento técnico, o estágio também tem como objetivo a participação em sessões internas e reuniões com clientes e o acompanhamento de membros seniores, o que permite desenvolver competências na gestão de projetos, resolução de problemas e trabalho em equipa, fundamentais para o contexto profissional.

1.5 Estrutura do documento

Este relatório está organizado em **seis capítulos**. O primeiro capítulo da **Introdução** aborda os aspectos iniciais e fundamentais do estágio. Começa com a secção de Motivação, onde é explicado o que motivou a realização do estágio e a relevância do projeto. Em seguida, é apresentado o Contexto, integrando o estágio e o seu impacto no contexto da empresa. A secção de Objetivos define os principais propósitos do estágio, tanto em termos técnicos quanto em desenvolvimento profissional. Após isso, o capítulo também descreve a Empresa, explicando sua atuação e relevância no mercado. Por fim, é apresentada a Estrutura do Documento, oferecendo uma visão geral da organização do relatório.

No segundo capítulo, intitulado **Conceitos, Tecnologias e Trabalho Relacionado**, são apresentados os fundamentos teóricos e as tecnologias que sustentam os projetos que ocorreram no âmbito do estágio, assim como uma revisão das soluções semelhantes disponíveis no mercado.

O terceiro capítulo, **Metodologia**, introduz o ciclo de vida do desenvolvimento de software e apresenta alguns modelos de desenvolvimento utilizados atualmente. De seguida, descreve o modelo de desenvolvimento adotado pela empresa. Este capítulo ainda, apresenta o contexto de cada uma das soluções em que o estágio teve efeito, os seus objetivos, requisitos funcionais e não funcionais e a gestão adotada para cada tarefa.

No quarto capítulo, **Implementação**, é detalhada a implementação das diferentes soluções. Este capítulo começa por filtrar e enumerar os requisitos das soluções que foram atribuídos ao estágio. Apresenta uma abordagem ao desenvolvimento, com uma breve descrição das tecnologias utilizadas e como se relacionam com as tarefas a desenvolver. Após isso, explica detalhadamente o processo de implementação e, por fim, discute os resultados e propõe um avaliação crítica, onde as tarefas desenvolvidas são analisadas tendo em conta os requisitos atribuídos.

No quinto capítulo, **Conclusões e Trabalho Futuro**, são discutidas as conclusões do estágio e apresenta sugestões para futuros aprimoramentos e possíveis novos desenvolvimentos.

Capítulo 2

Conceitos, Tecnologias e Trabalho relacionado

Neste capítulo, irá analisar-se as **tecnologias, plataformas e práticas utilizadas durante o estágio** de modo a contextualizar a sua importância e funcionalidades no ambiente de trabalho.

Esta análise é crucial para entender o cenário atual das ferramentas e serviços adotados, permitindo uma melhor compreensão do seu impacto nas atividades desenvolvidas e a tomada de decisões informadas, garantindo que os avanços mais recentes das tecnologias sejam aproveitados para criar soluções modernas e eficazes.

Dado que a empresa Make The Shift é uma **Microsoft Golden Partner**, é fundamental compreender a relevância das tecnologias fornecidas pela **Microsoft** no contexto das atividades desenvolvidas durante o estágio. A Microsoft é reconhecida como uma das maiores empresas de tecnologia do mundo, tendo sido fundada por Bill Gates e Paul Allen em 1975. Ao longo dos anos, a Microsoft cresceu para se tornar uma das empresas mais valiosas do mercado, influenciando profundamente o mundo da tecnologia e dos negócios. As suas soluções abrangem uma ampla gama de serviços e aplicações fundamentais para o sucesso dos projetos envolvidos, incluindo o **Microsoft 365**, que oferece ferramentas de produtividade e colaboração, e a **Microsoft Azure**, uma das plataformas líderes em computação na nuvem. Para a análise destas tecnologias foi consultado a plataforma oficial de documentação e cursos da Microsoft, Microsoft Learn [3].

O estágio na Make The Shift concentra-se no desenvolvimento de aplicações Web e móveis, tornando essencial a compreensão de tecnologias como **HTML, CSS** e **Javascript** [4], as quais desempenham um papel crucial na criação e design de interfaces atraentes e funcionais. Para além disso é necessário perceber como os utilizadores interagem com as aplicações, es-

tudando a área de **Iteração Pessoa-Máquina** [5].

Ainda neste capítulo, são mencionados e revistos **trabalhos relacionados** e semelhantes que exploram abordagens semelhantes às desenvolvidas durante o estágio.

2.1 Microsoft 365

Microsoft 365 é um vasto **grupo de aplicações e serviços baseados na nuvem** que visa aprimorar a produtividade e a colaboração nas organizações. Com ferramentas amplamente conhecidas e utilizadas no mercado, como SharePoint Online, Teams, Outlook, OneDrive for Business, Word, Excel e PowerPoint, além da poderosa Microsoft Power Platform com as tecnologias de Power Apps e Power Automate, o Microsoft 365 facilita a comunicação, a gestão de documentos, processos empresariais e o desenvolvimento de aplicações personalizadas.

2.1.1 OneDrive for Business

OneDrive for Business é um serviço de armazenamento na nuvem da Microsoft projetada especialmente para utilização empresarial. Esta tecnologia é integrante do ecossistema Microsoft 365 e oferece ferramentas avançadas de armazenamento, colaboração e segurança para ajudar as empresas a gerir os seus dados e documentos de forma eficiente. Estes arquivos podem ser acedidos e sincronizados em qualquer dispositivo conectado à Internet.

É possível partilhar arquivos com colegas, clientes e parceiros através da gestão dos administradores que controlam as permissões de edição, políticas de partilha e proteção por senha. Várias pessoas podem colaborar no mesmo documento em tempo real visualizando as alterações instantaneamente e permite ainda rastrear e restaurar versões antigas do documento. Toda a atividade e auditoria de utilização do documento pode ser monitorizada em relatórios gerados automaticamente para esse propósito.

A integração nativa com outras ferramentes do Microsoft 365 for Business, como o Teams, Sharepoint, Outlook, Word, Excel, PowerPoint, Power Apps e Power Automate proporcionam uma experiência do utilizador robusta e coesa de produção, revisão e automatização das mais diversas aplicações e fluxos de trabalho aumentando a eficiência operacional.

2.1.2 Sharepoint Online

O **Sharepoint Online** é uma plataforma de **colaboração e gestão de conteúdos baseada na Web**. Este é designado como um local seguro na nuvem onde os utilizadores podem

armazenar, organizar, partilhar e aceder a informação a partir de qualquer dispositivo ligado à Internet.

Com o SharePoint Online, é possível construir **sites intranet** personalizados, adaptados às necessidades específicas da organização. Esses sites podem incluir páginas personalizadas, onde os utilizadores podem partilhar informações e conteúdos, que podem incluir documentos, imagens, vídeos, notícias e *links* para atualizações relevantes.

Além disso, o SharePoint Online permite a criação de **bibliotecas de documentos**, que funcionam como repositórios centralizados para armazenar e organizar documentos importantes nas empresas. Estas bibliotecas oferecem recursos avançados, como controlo de versão, metadados e permissões de acesso, garantindo a segurança e a integridade dos dados.

Outro recurso essencial são as **listas**, que proporcionam um meio flexível de armazenar e gerir dados estruturados. As listas podem ser personalizadas para responder a uma variedade de necessidades, desde o rastreamento de tarefas e projetos até a gestão de ativos e processos de fluxo de trabalho. Estas listas são constituídas por itens horizontais com vários campos verticais, como se de uma base de dados tabular se tratasse. Os campos abrangem textos, escolhas *dropdown*, datas, textos multi-linhas, pessoas ou grupos baseados em e-mails, números, *boleanos*, *links*, moedas, localizações, imagens, metadados, *lookups* e avaliações. Com recursos avançados de personalização e automatização, estas listas são muitas vezes utilizadas para armazenar dados e tornam mais fácil a colaboração e a tomada de decisões.

Os recursos versáteis do SharePoint Online e sua integração com outras aplicações do Microsoft 365 tornam esta plataforma uma escolha popular para empresas que procuram melhorar a colaboração e a produtividade da sua equipa.

Quando o OneDrive for Business é acedido, é através de um site Sharepoint Online o que significa que tecnicamente os arquivos são armazenados numa biblioteca de documentos dentro do Sharepoint. O Sharepoint e o OneDrive for Business são muitas vezes utilizados em conjunto de modo a oferecer uma solução mais completa de gestão de documentos e colaboração empresarial.

2.1.3 Microsoft Power Platform

A tecnologia **Power Platform**, como o nome indica é um plataforma composta por um conjunto de ferramentas que permitem a criação de soluções empresariais personalizadas sem a necessidade de um extenso desenvolvimento de código, geralmente referidas como ferramentas *low-code*. A plataforma é composta por várias ferramentas integradas que facilitam a **automação de processos** (com a tecnologia Power Automate), a **construção de aplicações**

personalizados (com a tecnologia Power Apps), **análise de dados** (com a tecnologia Power BI), a **criação de páginas da web** (com a tecnologia Power Pages) e **soluções baseadas em Inteligência artificial** (com a tecnologia Microsoft Copilot Studio).

A Microsoft desenvolveu a linguagem de programação *low-code* **Power Fx** de modo a expor toda a lógica na sua plataforma Power Platform. Esta tecnologia fornece ainda integrações com o GitHub e o Microsoft Teams, entre outras aplicações da Microsoft.

A Power Platform possibilita a criação de vários **ambientes** onde se pode armazenar, gerir e partilhar dados empresariais, aplicações, *chatbots* e fluxos de automatização. Estes ambientes podem servir como contentores para a diferenciação de versões de teste e de produção das aplicações, *chatbots* e/ou fluxos de automatização, como podem também podem servir para separar os diferentes componentes que correspondam diferentes equipas ou departamentos numa empresa.

Cada ambiente é criado sob um ***tenant*** Microsoft Entra, e os seus recursos só podem ser acedidos por utilizadores dentro desse *tenant*. Cada ambiente está vinculado a uma localização geográfica.

Existem ainda o conceito de **soluções** na Power Platform. Estas são utilizadas para transportar aplicações e componentes de um ambiente para outro ou para aplicar um conjunto de personalizações a aplicações existentes. São constítudas por vários componentes como aplicações, mapas de sítios, tabelas, processos, recursos Web, escolhas e fluxos de automatização.

As soluções podem ser **não geridas** (unmanaged) ou **geridas** (managed). As primeiras permitem que os seus componentes sejam modificados livremente e por isso são utilizadas em ambientes de desenvolvimento com o objetivo de criar e personalizar aplicações e mais tarde podem ser exportadas como geridas ou não geridas. As segundas são utilizadas em ambientes de produção onde as personalizações são restritas e controladas. Uma vez implementadas, as soluções geridas não podem ser exportadas nem modificadas diretamente.

O **ciclo de vida de uma solução** passa pelo seu desenvolvimento inicial com todos os componentes necessário denominado como sua **criação** (“*create*”). Nesta fase é onde é criado a entidade **“publisher”** da solução, onde se encontram o prefixo dos componentes para evitar nomes repetidos, onde se especifica quem pode desenvolver e quem detém propriedade sobre a mesma. A solução pode sofrer um **“update”** através de uma solução mais recente, ou seja os componentes que as duas soluções possuam em comum, serão substituídos na versão mais antiga pelos mais recentes, mas os que não foram substituídos não são eliminados, mantendo-se na nova solução implementada. A solução pode também sofrer

um **"upgrade"**, que de forma contrária ao "update", todos os componentes da antiga solução serão eliminados e manter-se-ão apenas os componentes da nova versão pela qual se fez upgrade. Se não se pretende eliminar de imediato a solução antiga há a hipótese de realizar um **"stage for upgrade"**. Este irá fazer um upgrade à solução mas irá guardar a solução antiga até que o upgrade se realize de facto. Se as alterações à solução não forem significativas existe a opção de efetuar um **"patch"** que são pequenos *updates* de rápida instalação.

Os diferentes componentes das soluções podem ter **dependências** entre si dentro da mesma solução e/ou entre soluções diferentes. Uma boa gestão destas dependências é crucial para garantir que as soluções funcionem corretamente quando transportadas entre ambientes. Por exemplo, se uma solução contém dados e uma segunda solução tem fluxos de automatização que utilizam esses dados é necessário instalar primeiro a solução detentora dos dados e só depois a solução que contém os fluxos.

Power Apps

Power Apps é um ambiente de desenvolvimento que permite aos utilizadores criar aplicações personalizados sem a necessidade de codificação avançada. Esta oferece dois tipos principais de aplicações: **Canvas Apps** e **Model Driven Apps**.

Canvas Apps é uma plataforma que permite criar aplicações de forma rápida e intuitiva. Nesta é possível **arrastar e largar controlos** (tais como caixas de texto, botões, listas, imagens, entre outros) numa **tela em branco**. O utilizador pode **ajustar cada propriedade** da interface (altura, comprimento, cor, espaçamento, etc) de forma personalizada, sem a utilização de código.

Porém a plataforma permite também ajustar as propriedades dos controlos utilizando a linguagem da Microsoft, **PowerFx**. Através desta linguagem, ainda é possível definir comportamentos na aplicação de acordo com eventos, tais como o evento "OnVisibile" despertado quando o utilizador navega para um dado ecrã, ou o evento "OnSelect", disparado quando o utilizador clica num dado controlo. A linguagem PowerFx, viabiliza de forma simples a **lógica**, **dinamismo** e **interatividade** da aplicação.

Esta tecnologia possibilita a **integração de dados** de uma grande variedade de fontes da Microsoft e externas como SharePoint, Excel, SQL Server, OneDrive e serviços Web.

Estas aplicações pode ainda ser **partilhadas** para que os utilizadores possam executá-las num browser ou num dispositivo móvel.

Se não for necessário um design personalizado e se o pretendido é uma aplicação baseada num modelo de dados, pode-se optar por uma abordagem mais automatizada através das

Model Driven Apps. Estas são ideais para cenários em que se tem uma estrutura de dados complexa e precisa de ser rigorosamente controlada.

Neste tipo de aplicações é possível modelar formulários, vistas, gráficos, tabelas entre outros componentes, e a interface de utilizador predefinida ajusta-se automaticamente aos mais diversos dispositivos como telemóveis, computadores, tablets, etc. Para além disso, as relações entre tabelas permitem a navegação entre registos relacionados e garantem que os dados não são repetidos desnecessariamente.

Este tipo de aplicações utilizam um modelo de dados alojado no **Microsoft Dataverse**. Este serviço de suporte de dados da Microsoft permite ao utilizadores armazenar, partilhar e analisar dados de forma escalável e dinâmica num ambiente altamente seguro.

Do ponto de vista da **experiência do utilizador**, todas as Model Driven Apps oferecem uma experiência semelhante, simples e direta, acessível a maior parte dos utilizadores e dispositivos.

Ambas as formas de aplicação tem objetivos diferentes. Enquanto a Canvas Apps oferece mais personalização e detalhe as Model Diven Apps apostam na análise e robustez de dados. Ambas podem ser usadas simultaneamente em diferentes cenários no mundo empresarial.

Power Automate

O **Power Automate** é uma ferramenta versátil e poderosa de **fluxos de automatização de trabalho**. O seu processamento é baseado na nuvem o que permite aos seus utilizadores criar, gerir e executar fluxos a qualquer momento de forma eficiente, sem a necessidade de uma grande infraestrutura.

Para iniciar um fluxo é necessário um ***trigger***. Este *trigger* pode ser accionado automaticamente pelo sistema por exemplo ao receber um e-mail ou uma atualização num sistema de base de dados, pode ser executado manualmente por uma utilizador ou executado em intervalos regulares, como diariamente ou semanalmente.

Após o *trigger*, os fluxos são baseados em **ações** ligadas entre si. Cada ação é original de um **conetor**, estes podem ser do tipo padrão ou *premium*, ao contrário do primeiro tipo, os conetores do segundo tipo têm um custo adicional mesmo para a subscrição do Microsoft 365 for Business. Os conetores integram-se com centenas de serviços e aplicações, incluindo:

- **Control** que funciona como os controlos do fluxo (lógica condicional e ciclos de repetição).
- **Variables** permite armazenar e manipular variáveis temporárias durante a execução de

fluxos de trabalho, facilitando a organização e a gestão de dados dinâmicos.

- **Data Operations** oferece ferramentas para manipulação e transformação de dados, permitindo realizar operações como filtragem, ordenação e combinação de dados, facilitando a análise e processamento de informações em fluxos de trabalho automatizados.
- **Approvals** que permite automatizar processos de aprovação, enviando notificações e solicitações de aprovação tanto por e-mail quanto pelo Microsoft Teams, garantindo eficiência e rapidez na tomada de decisões.
- **SharePoint** que integra fluxos de trabalho com sites e bibliotecas de documentos do SharePoint, permitindo automatizar tarefas como criar, modificar ou gerir conteúdos, colaboração e processos empresariais.
- **Adobe PDF Services** permite automatizar a manipulação de documentos PDF, incluindo conversão, edição e extração de dados, agilizando processos e melhorando a eficiência na gestão de documentos.

Estes conetores podem ainda ser baseados em APIs personalizadas ampliando a capacidade de automatização e permitindo que os fluxos de trabalho incluam praticamente qualquer serviço que disponha de uma API.

É possível aceder facilmente às informações das conexões, *triggers* e ações anteriores com o **"Dynamic content"**. Este refere-se aos dados dinâmicos disponíveis dentro de uma ação ou condição no Power Automate, que podem ser diretamente selecionados e utilizados numa ação distinta.

Para uma manipulação ainda mais avançada e flexível dos dados dentro dos fluxos tem-se disponível as **"Expressions"** que permitem manipular e transformar os dados usando uma linguagem de expressões específica, com o Power Automate Expression Language, para realizar tarefas como manipulação de texto, cálculos matemáticos, controlo de fluxo e muito mais.

O **resultado de cada ação** normalmente é disponibilizado num formato estruturado como texto, JSON ou outro formato específico, dependendo da natureza dos dados envolvidos (XML, tabelas, imagens ou anexos). Através destes o utilizador pode analisar, decidir e encadear a próxima ação no fluxo.

Esta ferramenta permite também a integração com **modelos de inteligência artificial** a partir do conetor AI Builder. Através deste é possível o processamento de texto, reconhecimento de imagem e análise de sentimentos.

Alguns **casos de utilização** comuns são a automatização de tarefas manuais e repetitivas, como o envio de e-mails ou a atualização de registros em sistemas de base de dados. Sincronização de dados entre diferentes aplicações e serviços. Configuração de alertas e notificações automáticas baseadas em eventos específicos, como é exemplo a submissão da folha de horas no final de cada semana. Criação de fluxos de aprovação que podem ser geridos e rastreados de forma fácil. É importante notar que estes fluxos de trabalho complexos podem crescer com as necessidades da organização.

A **interface** é intuitiva com mecanismos de arrastar e soltar, que permitem aos utilizadores criar fluxos de trabalho sem necessidade de grande conhecimento em programação. Existem porém métodos que permitem lidar com casos de utilização mais complexos com uma maior eficiência e por isso tem uma maior complexidade técnica, como filtros ou expressões avançadas muitas vezes podendo utilizar XPath ou JSONPath.

Com a utilização desta ferramenta é possível notar uma redução do tempo gasto em tarefas manuais, permitindo que os funcionários se concentrem em atividades de maior valor.

Dataverse

O Dataverse é uma **plataforma de dados que permite gerir e armazenar modelos de dados personalizados** através da definição de entidades, campos, relações e regras de validação.

Este pode ser utilizado como uma **base de dados tabular**. Ao criar entidades, que são essencialmente tabelas, é possível armazenar dados de forma estruturada. Cada entidade pode ter campos específicos, semelhante às colunas de uma tabela, onde pode ser definido relações entre diferentes entidades, assim como se de uma base de dados relacional se tratasse.

É uma plataforma de dados que **faz parte do ecossistema do Microsoft Power Platform**, o que significa que é possível criar aplicações (através das Power Apps), fluxos de trabalho automatizados (através do Power Automate), relatórios e painéis (através o Power BI), tudo com base nos dados armazenados no Dataverse.

Além de integrar-se perfeitamente ao Power Platform, o Dataverse também pode ser integrado a uma variedade de outros serviços e aplicações, como o Microsoft Teams, SharePoint e Dynamics 365, permitindo uma experiência de trabalho integrada e eficiente.

O Dataverse oferece **recursos avançados de segurança e controlo de acesso** para proteger os dados presentes. Podem ser definidas permissões para controlar quem pode aceder, visualizar, editar ou excluir dados em diferentes entidades.

2.2 Microsoft Azure

Lançada em 2010, a Microsoft Azure tornou-se rapidamente uma das **principais plataformas de serviços na nuvem** do mercado, permitindo a implantação e gestão de aplicações e serviços por meio de uma rede central de dados. A plataforma oferece três modelos de serviços, incluindo:

- IaaS (Infrastructure as a Service): Oferece infraestrutura como servidores virtuais, armazenamento e redes.
- PaaS (Platform as a Service): Oferece uma plataforma que permite aos profissionais de desenvolvimento de software criar, gerir e implantar aplicações sem se preocupar com a infraestrutura subjacente.
- SaaS (Software as a Service): Oferece software pronto para utilização, acessível pela Internet.

Com um **modelo de pagamento por utilização**, as empresas podem escalar recursos conforme o necessário, beneficiando de uma infraestrutura global que garante alta escalabilidade, flexibilidade, gestão e segurança na nuvem para impulsionar a inovação e melhorar a eficiência operacional. Esta plataforma **integra-se na perfeição com outras ferramentas da Microsoft**, como o Microsoft 365, além de suportar diversas linguagens de programação e sistemas operativos, o que a torna uma escolha robusta e versátil para atender às necessidades dos modernos negócios digitais.

A empresa Make The Shift, irá dar utilização a alguns **serviços de computação** oferecidos pela plataforma PaaS **Azure App Service**, especificamente a plataforma **Azure Web Apps** para desenvolver e gerir aplicações Web e **Azure Functions** para executar tarefas de forma automatizada.

2.2.1 Azure Web Apps

Azure App Service oferece um ambiente completo e confiável para a criação, implantação e gestão de aplicações Web, APIs e back-ends móveis. Baseada no protocolo HTTP, esta plataforma oferece funcionalidades avançadas como escalabilidade automática, integração contínua e suporte para uma ampla variedade de linguagens e frameworks, incluindo .NET, Java, Ruby, Node.js, PHP e Python.

Com o Azure Web Apps, os profissionais de desenvolvimento de software podem criar e dimensionar rapidamente suas aplicações, beneficiando-se de uma infraestrutura totalmente

gerida e que garante grande disponibilidade, segurança robusta e escalabilidade automática com base na demanda das aplicações.

Oferece ainda integração contínua com tecnologias de controlo de versões utilizando ferramentas como Azure DevOps, GitHub e Bitbucket. Isto permite aos seus utilizadores realizar atualizações de código com rapidez, segurança e eficácia. Com suporte a ambientes de desenvolvimento e produção, a Azure Web Apps permite a criação de ambientes de teste isolados e a implementação de estratégias de desenvolvimento ágeis.

Esta plataforma é ideal para organizações que procuram uma solução de implantação confiável, escalável e de alto desempenho, capaz de suportar desde pequenos websites pessoais até grandes aplicações corporativas.

2.2.2 Azure Functions

Azure Functions são "micro-serviços" de computação *serverless*, hospedados pela Microsoft Azure, que permitem aos profissionais de desenvolvimento de software executar pequenas partes de código, chamadas **funções**, em resposta a eventos sem a necessidade de gerir a infraestrutura que os suporta.

Lançado em 2016, as Azure Functions facilitam a criação de aplicações escaláveis e reativas, suportando diversas linguagens de programação, como C#, JavaScript, Python, Java, e PowerShell. As funções intengram-se diretamente com ferramentas de desenvolvimento como Visual Studio, Visual Studio Code, Maven, entre outras de modo a promover um desenvolvimento e *debug* eficientes.

A principal vantagem das Azure Functions é a sua capacidade de **escalabilidade automática**. As funções são ativadas em resposta a uma variedade de ***triggers***, como alterações em bases de dados, eventos em fila ou pedidos HTTP como se de uma API Web se tratasse. Os programadores escolhem a linguagem de programação que mais os agrada, concentrando-se no código e na lógica do negócio. De seguida, lançam a função na plataforma Azure e deixam que a mesma realize, de forma responsiva, a gestão de alocação de recursos de computação conforme a exigência dos seus "micro-serviços".

Os **custos** estão inerentes às opções de hospedagem orientadas por eventos. Estas variam de totalmente sem servidor, onde apenas se paga pelo tempo de execução (plano de consumo), a instâncias sempre prontas para tempos de resposta mais rápidos (plano *Premium*). Quando há a previsão de que os recursos de alojamento são de uma alta demanda, é possível alojar as funções num plano fixo da Azure. Se o pretendido é ter controlo total sobre o ambiente de execução e as dependências das suas funções, pode implementá-las

em contentores totalmente personalizáveis.

Este modelo de computação garante que os custos sejam incorridos de acordo com a execução das funções, proporcionando uma solução económica, eficiente e ajustada para cargas de trabalho variáveis.

As Azure Functions podem ser ainda integradas diretamente em aplicações do Microsoft Power Platform. Estas podem ser usadas para realizar operações complexas ou que exigem acesso a serviços externos que o Power Apps ou Power Automate não oferecem nativamente.

2.3 HTML, CSS e JavaScript

HTML (HyperText Markup Language), **CSS** (Cascading Style Sheets) e **JavaScript** são as três tecnologias principais da **World Wide Web**. Primeiro, o **HTML** fornece a estrutura base dos conteúdo Web com elementos como títulos, parágrafos e imagens. De seguida, o **CSS** assume o controlo da apresentação visual, definindo o estilo e a disposição das páginas com cores, fontes e posicionamento. Finalmente, a interatividade e dinamismo são adicionados às páginas com recurso ao **JavaScript**, permitindo a manipulação do comportamento de cada página em resposta às ações do utilizador. Estas tecnologias criam uma experiência de utilizador personalizada e interativa: o HTML estrutura as páginas, o CSS fornece-lhe estilo e o JavaScript torna-as interativas.

2.3.1 HTML

HTML, abreviatura para **HyperText Markup Language**, traduzido Linguagem de Marcas de Hipertexto, não é considerada uma linguagem de programação, já que não pode criar funcionalidades dinâmicas.

Ao invés disso, como o nome indica **"Linguagem de marcas"**, é uma linguagem composta por diversas *tags*, estas definem elementos que dão forma às diferentes partes das página Web, como títulos, parágrafos, imagens, *links* e outros componentes de multimédia. Por sua vez, estes elementos são interpretadas pelos navegadores Web e correspondem ao conteúdo e estrutura de um documento. Outra linguagem de marcas muito conhecida é o XML, abreviação para Xtensible Markup Language, que é utilizada para definir regras de formatação de documentos de forma estruturada e hierárquica, facilitando a troca de dados entre sistemas.

Um **"Hipertexto"** é um texto usado para fazer referência a outros documentos ou partes do mesmo documento, chamadas hiperligações. Esta é a linguagem mais utilizada na construção

de páginas na Web, compondo a maior parte das páginas da Internet e das aplicações *online*.

É possível controlar o comportamento intrínseco de alguns elementos utilizando **atributos**, estas são palavras especiais definidas dentro da *tag* de abertura de cada elemento. Com estas *tags*, é possível identificar melhor um elemento, informar qual arquivo aquela *tag* deve utilizar, indicar o tipo de um campo de texto, etc.

Há **dois tipos de atributos** no HTML, os globais, aceites por todas as *tags*, como, por exemplo: class, id, lang, style, entre outros; e existem os específicos, que somente algumas *tags* possuem, como: src, disabled, href, label, etc.

Um documento HTML tem necessariamente **três *tags* principais**, que se traduzem em blocos:

- **Definição do documento** - <!DOCTYPE html>, é uma instrução para informar o navegador em que versão HTML o documento está escrito, de modo a este apresentar as páginas Web da melhor forma possível. Esta definição é obrigatória e deve ser a primeira linha de todos os documentos HTML.
- **Cabeça** - dentro da *tag* <head>, é onde se encontram as informações que não são transpostas visualmente para o utilizador do documento HTML. São dados de utilização e controlo do documento: título do documento, vinculação com outros arquivos, aplicação de *scripts* de lógica e metadados, codificação de caratéres.
- **Corpo** - dentro da *tag* <body>, coloca-se qualquer informação que se deseja apresentar, como imagens, sons, conteúdo multimédia, entrada de dados (formulários). É neste bloco onde se encontram as marcações para a construção da página Web e ainda elementos semânticos, como cabeçalhos, rodapé, conteúdo principal, etc.

A **criação do HTML é atribuída a Tim Berners-Lee**, o inventor da World Wide Web. A sua ideia era criar um ambiente em que os cientistas pudessem divulgar as suas pesquisas para que colegas pudessem consultá-las com facilidade. Berners-Lee acabou por criar o HTML como linguagem padrão para elaborar esses documentos.

Desde a sua implementação, em 1993, o **HTML passou por diversas melhorias** e hoje é o alicerce de *blogs*, *e-commerces*, redes sociais e todo o tipo de página acessível através de um navegador Web.

Uma das reformas da tecnologia foi o **XHTML**, eXtensible HyperText Markup Language. Este combina a flexibilidade do HTML com a sintaxe rigorosa do XML, garantindo maior compatibilidade e interoperabilidade entre diferentes dispositivos e navegadores.

Uma das **melhorias mais célebres**, publicada em 2014, **foi o HTML5**. Esta atualização trouxe a incorporação nativa de vídeos e arquivos de áudios. Ao invés de ser necessário utilizar o Flash Player, com a nova atualização é possível incorporar vídeos e áudios com as novas *tags* <audio></audio> e <video></video>. Esta versão também possui compatibilidade com SVG (vetor gráfico escalável) e MathML para fórmulas científicas e matemáticas.

Estas melhorias e reformas são da **autoria do World Wide Web Consortium (W3C)** que desenvolve e mantém especificações do HTML, para além de providenciar atualizações regulares.

Para apresentar um documento como uma página HTML, a maior parte dos navegadores utilizam um modelo internacional denominado **DOM**, Document Object Model que, mais uma vez fiscalizada pela entidade World Wide Web Consortium (W3C), é uma convenção multiplataforma e independente de linguagem de programação, para representar e interagir com objetos em documentos HTML, XHTML e XML. Esta convenção organiza os elementos de um documento numa estrutura em árvore, chamada de Árvore DOM e por sua vez associa-os a nós/objeto. Deste modo é possível endereçar e manipular através de uma API sobre os objetos.

Assim o HTML é essencial para a construção de qualquer página Web, servindo como a espinha dorsal que organiza e apresenta o conteúdo de forma estruturada, lógica e acessível.

2.3.2 CSS

O CSS é a abreviatura para **Cascading Style Sheet**, traduzido Folha de Estilo em Cascata e é utilizada para conceder estilo a elementos escritos em linguagens de marcas como é o exemplo do HTML.

O CSS separa o conteúdo da representação visual do Website. Com este é possível alterar a cor do texto e do fundo, fonte e espaçamento entre parágrafos, para além de ser possível realizar variações de *layouts* e ajustar imagens. O navegador processa as regras de estilo do CSS e aplica-as aos objetos correspondentes na árvore DOM, alterando a apresentação visual dos elementos indicados.

Foi desenvolvida pelo W3C (World Wide Web Consortium) em 1996, já que o HTML não foi projetado para ter *tags* que ajustam e formatam as páginas.

Este mecanismo de **formatação é aplicado** diretamente nos atributos *style* das *tags* HTML, chamado de *inline* CSS, ou inserido dentro de um bloco com *tags* próprias <style>, dentro do cabeçalho do HTML. Também é possível, incorporar estilos num documento HTML adicionando um *link* para um folha/documento CSS que contém estilos. Assim, quando se

pretende alterar a aparência dos documentos vinculados a este arquivo CSS, basta modifica-lo e todos os documento irão seguir as alterações.

Tem uma **sintaxe simples** utilizando várias de palavras da língua inglesa para especificar os nomes das propriedades de estilo de uma página. Uma instrução CSS consiste num seletor que referência um elemento HTML (pode ser o conteúdo do atributo “class” ou “id” num elemento, tipo de elemento, pseudo-classes ou pseudo-elementos, no caso do inline CSS não há necessidade desse seletor já que o estilo se aplica ao próprio elemento que contém o atributo *style*) e um bloco de declaração. Cada bloco de declaração contém uma propriedade e um valor, separados por dois pontos (:). Cada declaração é separada por ponto e vírgula (;).

O CSS pode ser complementado por **frameworks**, como Bootstrap e Foundation. Estas são bibliotecas que facilitam o design e *layout* das páginas Web seguindo os padrões CSS. São utilizados tanto para uma prototipagem rápida quanto para a concessão de Website completos.

2.3.3 JavaScript

JavaScript foi implementada como **parte dos navegadores Web para que as páginas Web fossem interativas** com o utilizador. Isto acontece através da execução de *scripts* do lado do cliente sem a necessidade dos mesmos serem executados pelo servidor, utilizando técnicas como AJAX (Asynchronous JavaScript and XML), pode controlar o navegador de forma assíncrona e alterar o conteúdo do documento exibido. A grande maioria dos Websites utiliza esta linguagem, e todos os navegadores mais conhecidos têm um mecanismo JavaScript dedicado para executá-lo. É atualmente a principal linguagem de programação *client-side* em navegadores Web.

Porém os mecanismos JavaScript estão agora incorporados em muitos outros tipos de software, incluindo servidores (e.g. node.js), bases de dados (e.g. Apache CouchDB) e mesmo programas que não são da Web, como processadores de texto e PDF (e.g. Adobe Acrobat).

O ecossistema JavaScript é vasto, com inúmeras **bibliotecas e frameworks** que facilitam o desenvolvimento Web, como React, Angular, Vue.js para front-end, e o anteriormente referido Node.js para back-end.

É uma **linguagem de programação estruturada e interpretada**, o primeiro conceito significa que utiliza funções, blocos de código, e outros elementos promovendo a organização lógica e hierárquica do código enquanto que o segundo diz que o código é executado diretamente, linha por linha, por um interpretador, sem a necessidade de ser compilado para código máquina antes da execução, permitindo uma execução mais flexível que facilita o *debug* e

testes rápidos.

Esta é conhecida como a **linguagem de *script* de alto nível**, ou seja é uma linguagem utilizada para escrever ***scripts*** que automatizam tarefas dentro de grandes programas, neste caso grandes sistemas Web como Websites e aplicações Web complexas. Considera-se de **alto nível** já que abstrai muitos detalhes do hardware do computador, permitindo aos programadores escrever código mais intuitivo e focado em resolver problemas lógicos ou relacionados com a aplicação em si, invés de lidar com detalhes técnicos do computador.

Considera-se que o JavaScript possui uma **tipagem dinâmica fraca**. **Tipagem dinâmica** siginifica que os tipos de dados são determinados em tempo de execução, e não é necessário declarar explicitamente o tipo das variáveis. Para além disso, é uma linguagem de **tipagem fraca**, o que significa que permite conversões implícitas entre tipos diferentes (por exemplo, somar um número com uma string).

O JavaScript suporta ainda múltiplos paradigmas de programação, incluindo ser uma **linguagem orientada a objetos**, o que permite a definição de classes e objetos, encapsulando dados e comportamentos. E também como uma **linguagem imperativa** dado que é baseada em declarações que mudam o estado do programa, como por exemplo utilizando *loops* e condições. Para além destes paradigmas, o JavaScript é uma **linguagem funcional**, pois utiliza funções de primeira classe (são tratadas como uma variável, ou seja, podem ser atribuídas a variáveis, passadas como argumentos e retornadas por outras funções), funções de ordem superior (funções que podem receber outras funções como argumentos ou retornar funções como resultado) e métodos como map, filter e reduce. Suporta também **programação declarativa**, onde se descreve o que se deseja que aconteça e a biblioteca ou framework (por exemplo React) cuida dos detalhes de implementação. É também **orientado a eventos**, tratando dos eventos vindos do utilizador e do sistema.

Com a introdução de ES6 (ECMAScript 2015) e versões posteriores, o JavaScript ganhou novos recursos significativos, como classes, módulos, arrow functions, let/const, e promises, tornando a linguagem mais poderosa e versátil.

Para um desenvolvimento Web seguro vários **aspectos de segurança** são críticos, como o tratamento de XSS (Cross-Site Scripting) prevenindo que *scripts* maliciosos sejam injetados em páginas Web, levando ao roubo de dados ou execução de ações indesejadas. E CORS (Cross-Origin Resource Sharing), um mecanismo de segurança que permite que os recursos de uma página Web sejam acedidos através de domínios diferentes, usando cabeçalhos HTTP específicos para evitar ataques.

O JavaScript tem uma das maiores e mais ativas **comunidades**, com muitos recursos

disponíveis, incluindo documentação, tutoriais, fóruns e conferências como JSConf.

2.4 Interação Pessoa-Máquina

No âmbito do estágio na empresa Make The Shift, a grande maioria dos objetivos estão associados ao **desenvolvimento de aplicações**, tanto para dispositivos móveis quanto para computador. Neste contexto, é fundamental destacar os princípios de design e usabilidade que garantem interfaces de utilização intuitivas, eficientes e que proporcionem uma experiência satisfatória a quem as utiliza. O campo que estuda estes conceitos tem como nome **Interação Pessoa-Máquina** (IPM).

Este é um campo é multi-disciplinar que se foca na interação entre humanos e computadores, em particular na concepção, avaliação e implementação de sistemas informáticos interativos para utilização humana, considerando a engenharia de computação, ciências cognitivas e condições humanas.

Considerando a diversidade de dispositivos e utilizadores, a implementação de boas práticas estudadas por este ramo é crucial para o sucesso das aplicações desenvolvidas, assegurando que atendem às necessidades e expectativas dos utilizadores de forma eficaz.

Para a construção de um **sistema computacional interativo** é preciso ter presentes alguns conceitos essenciais.

Um sistema destes, por norma, tem uma **interface** na qual o seu utilizador pode interagir. A concepção desta interface é um dos pontos mais críticos de toda a interação pessoa-máquina e é denominado de **Design da Interface do Utilizador**, em inglês User Interface Design (UID). Este processo inclui a análise e escolha de cores, dados de entrada, botões, listas, disposição dos componentes, tipografia e multimédia.

Este conceito faz parte de um grupo maior relacionado com a experiência de utilização do produto, por isso pode dizer-se que quando se desenvolve qualquer sistema interativo é necessário ter em atenção um design centrado na experiência do utilizador - **Design da Experiência do Utilizador**, em inglês User Experience Design (UXD). Este é o processo de design utilizado para criar produtos que fornecem experiências significativas e relevantes aos utilizadores.

Porém este processo de Design da Experiência do Utilizador não abrange apenas o Design da Interface de Utilizador, abrange também todo o processo de aquisição e integração do produto, incluindo os aspetos da **marca**, **usabilidade** e **função**.

A **usabilidade** é uma medida realizada ao longo de todo processo de desenvolvimento e

que mede o quão bem o utilizador pode usar um produto para atingir um objetivo. Existem seis metas de usabilidade:

- **Efetiva**, mede a qualidade da produto em fazer o que é suposto fazer, por exemplo através da percentagem de utilizadores que completaram tarefas específicas com sucesso
- **Eficiente**, mede a forma como a aplicação suporta os utilizadores, por exemplo estimando o tempo necessário que o utilizador leva para completar tarefas
- **Segura**, mede a segurança que é dada ao utilizador ao utilizar o produto, deve precaver os utilizadores para não provocarem danos
- **Útil**, mede se as caraterísticas do produto são úteis para o contexto que foi criado
- **Fácil de aprender a usar**, mede o quão fácil um utilizador inexperiente executa rapidamente tarefas, através, por exemplo do cálculo por exemplo a relação entre o tempo gasto por um utilizador inexperiente e um experiente a realizar a mesma tarefa
- **Fácil de memorizar com usar**, mede o quão fácil é reutilizar o produto contabilizando por exemplo, o número de erros cometidos a executar a mesma tarefa ao longo do tempo

Porém um produto com **boa usabilidade não garante necessariamente um boa experiência ao utilizador**, já que esta experiência é concetualmente holística e depende de vários fatores como ser visualmente atraente, ir de encontro às necessidades e expectativas específicas dos utilizadores (criando a necessidade da compreensão mais profunda dos mesmos), relevância e a utilidade do conteúdo e das funcionalidades oferecidas, suporte ao cliente e integração com outros sistemas.

De modo a realizar aplicações digitais centradas na experiência de utilizador pode seguir-se um **modelo iterativo** de **Análise**, **Design**, **Implementação** e **Avaliação**. Este começa por identificar as necessidades dos utilizadores e definir as tarefas numa primeira fase de análise. De seguida é necessário desenvolver modelos conceptuais (como *Wireframes* e Mockups) e protótipos na fase de design. Após esta fase implementa-se os protótipos funcionais em computador e/ou smarphone. Diz-se que este modelo é **iterativo** já que após as fases de design e implementação devem existir avaliações das ideias e dos protótipos de modo a calcular a sua usabilidade e melhorar constantemente o produto. É muito comum depois de uma primeira fase de implementação ter que se passar de novos pelas fases anteriores de modo a aperfeiçoar o produto.

Uma forma de avaliar o design das interface dos sistemas interativos, identificar falhas e erros são as Heurísticas de Nielsen [6], estas são seguidas de modo a corrigir e a otimizar

experiência do utilizador. Estas orientações são “evergreen” no sentido de que são apontamentos relevantes que não tem validade, ou seja não se fecham em conceitos específicos mas sim lembretes para garantir a melhor experiência do utilizador.

As 10 heurísticas de Nielsen são:

1. Visibilidade do status do sistema - Deve-se sempre informar o utilizador sobre o progresso de determinado processo, permitindo que este tenha o máximo de controlo. Por exemplo a barra de progresso num vídeo do YouTube.

2. Correspondência entre o sistema e o mundo real - Para que o utilizador tenha uma interação natural, fácil e intuitiva com o sistema é fundamental que a sua interface seja compatível com o mundo real. Por exemplo a semelhança entre a troca de correios físicos e eletrónicos.

3. Liberdade e controlor do utilizador - Quando o utilizador realiza alguma ação por engano deve ter a liberdade e controlo para desfazer a ação ou refazer uma ação desfeita, sem muito esforço. Por exemplo a opção de desfazer que os correios eletrónicos normalmente oferecem quando o utilizador apaga um e-mail.

4. Consistência e padrões - Deve ser construído um guia de estilo para um determinado produto que facilite a criação de novos ambientes mantendo o padrão visual já definido, com a padronização de cores, utilização de grupo tipográfico, escolha de imagens e vetores e características dos botões e *links*. Por exemplo a consistências entre as páginas das redes sociais.

5. Prevenção de erros - Esta heurística parte da premissa que os acidentes acontecem e que o design de interação deve ser desenhado para reduzir esse tipo de situações, melhorando também a terceira heurística que se foca no controlo por parte do utilizador. Por exemplo todas as ações com consequências que não podem ser desfeitas precisam de um aviso prévio, como caixas de confirmação antes de eliminar um arquivo.

6. Reconhecimento em vez de memorização - Aqui destaca-se a importância de minimizar a carga cognitiva dos utilizadores, facilitando o reconhecimento de opções e informações em vez de exigir da sua memória. Por exemplo a utilização de ícones com rótulos descritivos que ajudam os utilizadores a reconhecer facilmente a função de cada botão ou opção invés de se lembrarem.

7. Eficiência e flexibilidade de utilização - A interface deve ser eficiente e flexível para que tanto utilizadores inexperientes quanto aqueles mais experientes possam utilizá-la de forma eficaz. Por exemplo permitir que os utilizadores ajustem e personalizem a interface de acordo com suas preferências, ou criar atalhos que permitam aos utilizadores mais experiente

realizar tarefas mais rapidamente.

8. Estética e design minimalista - Quanto maior a quantidade de informação mais análise e decisões o utilizador precisa de fazer, por isso, apostar na simplicidade da composição visual facilita a navegação do utilizador. Por exemplo, manter apenas informações relevantes em primeiro plano e as informações menos importantes podem ser deixadas em segundo plano (menus, abas, etc.).

9. Auxiliar utilizadores a reconhecer, diagnosticar e recuperar erros - É importante que o utilizador tenha acesso a diagnósticos de problemas e situações, sendo possível este entender o aconteceu e como é possível solucioná-lo. Por exemplo - avisos dos formulários nos campos que não foram preenchidos corretamente.

10. Ajuda e documentação - Seguindo as heurísticas acima descritas é provável que o sistema seja fácil de usar, porém é fundamental que a ajuda seja sempre de fácil acesso. Por exemplo criar uma página de FAQ com as perguntas mais frequentes que auxiliem o utilizador.

2.5 Trabalho relacionado

Para fornecer um contexto mais amplo sobre as soluções propostas pela empresa, é importante analisar aplicações e soluções semelhantes já existentes no mercado. Esta análise ajuda a identificar características, falhas e inovações que podem informar e servir de inspiração para o desenvolvimento das soluções.

2.5.1 Aplicação de Gestão de Cartões de Negócio Digitais

Esta solução visa fornecer uma aplicação móvel para a criação e gestão de cartões de negócios digitais. Com o objetivo de compreender melhor o mercado e as opções disponíveis, foram pesquisadas algumas aplicações semelhantes.

A primeira aplicação é a “HiHello” [7]. Esta é uma aplicação móvel para a criação e gestão de cartões de negócios digitais. Entre as suas principais funcionalidades, destaca-se a introdução de dados personalizados para a criação de um ou vários cartões digitais para o mesmo utilizador, oferecendo uma variedade de *templates* à escolha. É possível adicionar fotos, logotipos, e incluir uma ampla gama de informações, como nome, detalhes de contato, site, endereço e perfis de redes sociais.

Esta aplicação tem ainda a capacidade de digitalizar os detalhes do contatos com a câmera do smartphone, utilizando inteligência artificial, o que permite extrair informações de cartões de papel integrando essas informações diretamente na aplicação. Para este processo

não requer uma ligação à Internet e garante a privacidade e segurança dos dados, que são armazenados localmente no smartphone. Apresenta a partilha dos cartões digitais através de NFC e um código QR.

A segunda aplicação é a "KADO" [8]. Esta aplicação móvel oferece as mesmas funcionalidades da aplicação descrita anteriormente à excepção de oferecer a integração com CRMs e ferramentas de produtividade, facilitando o acompanhamento e gestão de contatos.

Ambas as aplicações mencionadas permitem o *scan* de um código QR para a partilha dos cartões digitais, porém apenas a aplicação "Popl", [9] permite o scan e a partilha sem necessidade de que o receptor tenha a mesma aplicação instalada.

Caraterísticas como a introdução automática das informações presentes no cartão digital, a possibilidade de escolher entre vários modelos de cartão e a partilha através de código QR, sem a necessidade de ter a aplicação instalada, serão fundamentais durante o desenvolvimento da solução da empresa.

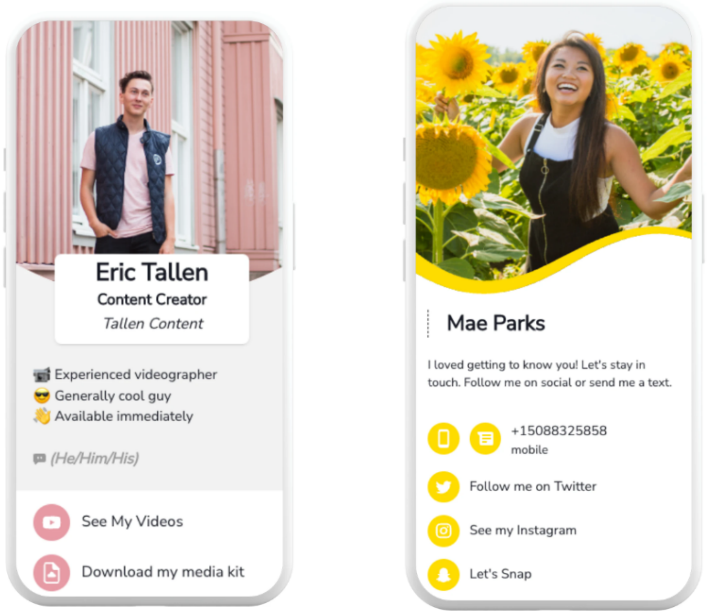
O design do cartão digital é crucial para criar uma impressão duradoura e impactante. De acordo com a aplicação "HiHello" em [10], um bom design não só chama a atenção, como também pode aumentar significativamente as oportunidades de interação profissional. A plataforma fornece algumas sugestões para criar cartões digitais:

- Realizar pesquisa: antes de iniciar o design, pesquisar diferentes opções e selecionar o estilo e conteúdos mais adequados.
- Debater ideias: definir o propósito do cartão e pensar como se pode se destacar em diferentes contextos.
- Selecionar informações: escolher as informações que serão incluídas no cartão. Além dos detalhes básicos, considerar adicionar informações adicionais relevantes consoante o propósito do cartão, como *links* para redes sociais ou vídeos.
- Escolher design e template: utilizar alguns *templates* já disponíveis para criar um cartão que se alinha com as suas necessidades.

A "HiHello" oferece quatro ideias de designs para cartões digitais de negócios:

- Design Elegante: escolher um esquema de cores neutras, texto minimalista com uma tipografia sofisticada, logotipo discreto, foco nas informações essenciais e garantir uma apresentação limpa e profissional. Este estilo é ideal para profissionais que desejam transmitir uma imagem séria e requintada. Um exemplo deste modelo consta na Figura 2.1 (a).

- Design Divertido: utilizar cores vibrantes, elementos gráficos chamativos, imagens e texto que representem a personalidade ou a área de atuação. Este estilo é perfeito para profissionais criativos ou para eventos informais. Um exemplo deste modelo encontra-se na Figura 2.1 (b).
- Design Simples: adotar um esquema básico com uma palete de cores suaves, formas e texto simples e espaço adequado para todas as informações. Este estilo é prático e eficiente, adequado para as áreas onde a clareza é valorizada. Um exemplo deste modelo está representado na Figura 2.2 (a).
- Design Moderno: estabelecer uma palete de cores contemporâneas, linhas limpas, tipografia moderna, elementos gráficos modernos. Este estilo é ótimo para transmitir inovação, é especialmente eficaz em setores tecnológicos. Um exemplo deste modelo pode ser visualizada na Figura 2.2 (b).



(a) Design Elegante (b) Design Divertido

Figura 2.1: Exemplos de modelos dos cartões da autoria 'HiHello' - parte 1

(a) Design Simples (b) Design Moderno

Figura 2.2: Exemplos de modelos dos cartões da autoria 'HiHello' - parte 2

Estes modelos serão considerados no desenvolvimento do *template* a ser apresentado aos clientes.

2.5.2 Aplicação para Gestão de Auditorias

Com esta solução, a empresa tem como objetivo fornecer uma aplicação móvel que permita aos empregados de uma empresa realizar auditorias no terreno, registando comentários e observações com recurso a texto e imagens e partilhar essas informações com contatos designados, através de relatórios gerados automaticamente. Para entender melhor o mercado e as soluções existentes, foram analisadas duas aplicações semelhantes.

A primeira aplicação é a "Incident Report" [11]. Esta aplicação permite aos utilizadores registar incidentes e perigos diretamente a partir de um smartphone. As informações recolhidas podem ser enviadas para um contato preferido, facilitando a comunicação em situações de emergência ou no terreno. O foco principal desta aplicação é a sua simplicidade.

A segunda aplicação analisada é a "Report Generator" [12]. Esta aplicação móvel é mais complexa, pois permite não só o registo de incidentes, como também a criação de relatórios de auditoria completos, que podem incluir perguntas personalizadas, fotografias e ações a serem tomadas. A aplicação destaca-se pela sua utilização offline, notificações por e-mail, geração de relatórios em PDF personalizados e atribuição de ações.

Estes exemplos serão fundamentais para a aplicação que a empresa pretende desenvol-

ver. Trata-se de uma aplicação simples, que permitirá realizar auditorias através da inserção de ocorrências com observações e imagens. Além disso, será necessária a geração de um relatório final em PDF, semelhante à aplicação descrita anteriormente.

2.5.3 Automatização de Agendamento de Reuniões Internas

Com a automatização de agendamento de reuniões internas a empresa pretende o agendamento de reuniões informais, todas as semanas, entre os seus colaboradores em grupos aleatórios por forma a promover a comunicação e colaboração entre os membros das diferentes equipas.

Com esta automatização em mente analisou-se a ferramenta para computador "Calendly" [13]. Esta é projetada para otimizar o processo de agendamento, eliminando tarefas manuais e repetitivas, como a troca de e-mails e a coordenação manual de horários, o que permite a concentração da equipa em prioridades mais importantes. A ferramenta oferece integrações com outras plataformas, fornecendo os detalhes de agendamento dos utilizadores, o que facilita o agendamento de reuniões de grupo com base na disponibilidade dos seus membros, garantindo a participação de todos os convidados.

Para além disso, o "Calendly" valoriza a experiência do utilizador, introduzindo desde questionários pré-chamada até à definição de tipos de eventos, assegurando a comunicação adequada e consistente entre os participantes e melhora a qualidade da experiência da reunião. As características desta aplicação serão consideradas na solução que a empresa pretende criar.

2.5.4 Sistema Dinâmico de Gestão de Notificações e Tarefas

Esta solução faz parte de uma aplicação para computador destinada à identificação e relato de problemas, bem como à atribuição e gestão de tarefas. O objetivo é implementar um sistema de notificações eficiente, permitindo alertar os utilizadores sobre problemas relatados na aplicação e automatizar a criação e atribuição de tarefas com base em condições específicas.

Durante a pesquisa por soluções semelhantes, foi analisada a aplicação móvel "Task Bird" [14]. Esta permite que os membros de uma equipa recebam alertas imediatos sobre alterações de agenda, cancelamentos de compromissos ou solicitações de emergências por meio de uma aplicação móvel.

Entre as funcionalidades principais do "Task Bird", destacam-se as atualizações em tempo real, que mantêm a equipa informada sobre alterações na sua agenda e novas ocorrências,

garantindo que todas as comunicações importantes sejam recebidas e tratadas rapidamente. Além disso, o “Task Bird” oferece recursos para monitorizar tarefas, permitindo rastrear o estado das tarefas desde o início até ao progresso e conclusão, facilitando o acompanhamento remoto da eficiência no trabalho.

Posteriormente foi identificada a funcionalidade “Custom Triggers for Notifications”, parte do “ManageEngine ServiceDesk Plus MSP” [15].

O “ManageEngine ServiceDesk Plus MSP” é uma plataforma para computador para gestão de serviços de tecnologias de informação. Uma das suas funcionalidades mais relevantes é a capacidade de definir critérios específicos para notificações e automação de ações. Através dessa funcionalidade, oferece a oportunidade ao utilizador de configurar *triggers* que disparam automaticamente ações pré-definidas, como o envio de notificações via e-mail ou SMS, a execução de funções personalizadas, ou até *scripts* programados.

De forma diferente de outros sistemas que apenas permitem um conjunto de regras para definir os *triggers* das notificações estáticas ou genéricas, a possibilidade de personalizar esses *triggers* e a ampla configuração da notificações aumenta as possibilidades de automação, tornando-a uma solução mais flexível.

As diferenças entre as duas plataformas destacam como cada solução atende a necessidades específicas em contextos distintos. A “Task Bird” foca-se em fornecer atualizações em tempo real para a coordenação e comunicação de equipas que precisam de atualizações constantes no terreno. Por outro lado, a “ManageEngine ServiceDesk Plus MSP” permite configurar notificações automáticas baseadas em condições específicas, podendo incluir notificações por e-mail, SMS ou ações customizadas.

No contexto deste projeto, essas funcionalidades serviram como referência para a criação de um sistema dinâmico para a gestão de notificações e tarefas que seja responsivo e flexível, permitindo que notificações sejam disparadas automaticamente com base em condições personalizadas e que tarefas sejam atribuídas e completadas sem intervenção manual.

Capítulo 3

Metodologia

Neste capítulo, será apresentada a **metodologia** adotada para o **desenvolvimento das aplicações** propostas durante o estágio, com o objetivo de fornecer uma visão clara do processo de trabalho. A metodologia escolhida é **fundamental** para garantir que as aplicações sejam desenvolvidas atendendo aos **requisitos estabelecidos** e sejam **entregues dentro dos prazos** e com a **qualidade** esperada.

Inicialmente, irá descrever-se conceitos fundamentais utilizados pela empresa para o desenvolvimento de soluções, incluindo as diferentes **fases e modelos de desenvolvimento de software** utilizados atualmente.

De seguida, com base nos conceitos descritos, serão apresentadas as considerações sobre a **modelo proposto** pela empresa para o desenvolvimento de software, destacando as suas contribuições para o sucesso dos projetos e refletindo sobre a eficácia da metodologia escolhida.

Este capítulo incluirá ainda uma **descrição detalhada das soluções desenvolvidas**, especificando o seu contexto, objetivos, requisitos funcionais e não funcionais e metodologia utilizada em cada uma, garantindo assim uma visão organizada das necessidades de cada projeto.

3.1 Fases e Modelos de Desenvolvimento de Software

O **desenvolvimento de software** pode ser um **processo complexo e demorado** o que leva a **adoção de práticas testadas e reconhecidas** na indústria, fundamentais para um processo de desenvolvimento baseado em **abordagens sistemáticas, disciplinadas e quantificáveis** que visa a redução de riscos, garantia de qualidade, eficiência e produtividade.

Habitualmente, as empresas que desenvolvem software baseiam-se nestes métodos pré-

concebidos de forma a criar o seu **próprio modelo de desenvolvimento**, utilizando-o continuamente nos seus projetos de forma a obter um processo de desenvolvimento consistentemente eficiente.

Diz-se por **processo de desenvolvimento** de software um **conjunto de atividades**, parcialmente **ordenadas**, com a finalidade de obter um **produto de software**. É estudado dentro da área de **Engenharia de Software**, e é considerado um dos principais mecanismos para se obter software de qualidade e cumprir corretamente projetos de desenvolvimento. O conceito de **Ciclo de Vida do Desenvolvimento de Software** está frequentemente associado a esse processo de desenvolvimento, organizando-o em fases estruturadas e sistemáticas.

3.1.1 Ciclo de Vida do Desenvolvimento de Software

Um conceito que se considerou durante todo o estágio na empresa é o do **Ciclo de Vida do Desenvolvimento de Software** (SDLC) [16]. Este conceito visa dividir o processo de desenvolvimento em fases bem definidas, permitindo uma melhor gestão desse processo sistemático, com objetivos claros que podem ser alcançados em diferentes etapas do desenvolvimento. Trata-se de uma estrutura que define fases iterativas para o desenvolvimento de software que incluem **análise**, **design**, **desenvolvimento** e **teste**. Estas fases podem ser visualizadas na Figura 3.1.

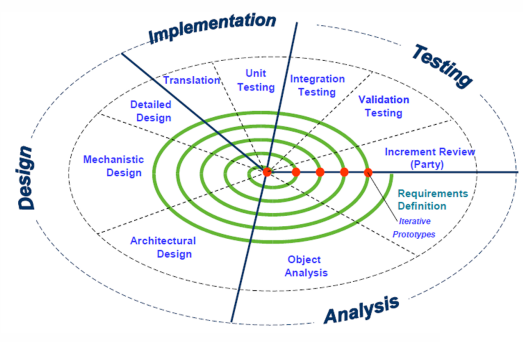


Figura 3.1: Processo iterativo do Ciclo de Vidade do Desenvolvimento de Software [Douglas, 2004]

Abaixo, apresenta-se em detalhe estas quatro fases do Ciclo de Vida do Desenvolvimento de Software:

Análise

Nesta fase, analistas, líderes do projeto e os líderes das equipas de desenvolvimento reúnem-se com os clientes de modo a perceber como as suas propostas se relacionam e como

podem ser realizadas, utilizando a tecnologias disponibilizadas pela empresa, respondendo a perguntas como e a quem se destina o software, como irá ser utilizado e que informação irá ser processada. Nesta fase são definido os objetivos e as necessidades do projeto, a caraterização dos utilizadores, os recursos a utilizar, o cronograma e o orçamento. Para isso recorre-se a algumas técnicas como a criação de *personas*, avaliação dos diversos casos de utilização e a estimação dos requisitos funcionais e não funcionais que pode e deve ser guardado sobre a forma de documento (Documento de Especificação de Requisitos). Este documento servirá de base para a próxima fase deste ciclo de desenvolvimento e deve ser consultado, sempre que necessário, pelos profissionais de desenvolvimento de software. É também nesta fase que os riscos e a qualidade do produto são avaliados.

Design

É a fase onde se define, através da análise realizada na fase anterior, o design detalhado do software, fundamental para garantir que produto final atende aos requisitos definidos. Isso inclui a escolha das linguagens de programação e a plataforma de desenvolvimento, a definição da interface do utilizador através de *Wireframes* e *Mockups*, a criação de diagramas de comunicação entre o utilizador e o software, a arquitetura do modelo de dados e também os diagramas de fluxos dos dados. É ainda importante especificar qualquer integração tecnologias providenciadas a partir do exterior. Na âmbito da empresa, esta fase, é realizada por colaboradores mais experientes como o líder do projeto ou o arquiteto.

Implementação

É quando a equipa de desenvolvimento converte os designs definidos na fase anterior em software, através das plataformas e linguagem de programação escolhidos. É importante que os profissionais de desenvolvimento de software trabalhem colaborativamente, sigam rigorosamente o calendário e dividam a fase de implementação em etapas com prazos rigorosos para manter expetativas realistas, para que se consiga acompanhar o progresso do projeto e para que este seja realizado corretamente. A complexidade desta fase está diretamente relacionada com o resultado das fases anteriores, ou seja quanto melhor a análise e o design do projeto mais fácil e eficiente é a seu implementação. Se houver um menor esforço e consequentemente um menor tempo de realização para os protótipos, diagramas de comunicação e fluxos de dados, a qualidade do software final irá ser afetado negativamente.

Teste

É a fase onde o software desenvolvido na fase anterior é testado pela equipa de controlo de qualidade. Esta fase está diretamente relacionada com as fases anteriores num processo cíclico, como se observa na Figura 3.1. Quando erros são detectados, dependendo do tipo e da gravidade do erro, o processo poderá ter a necessidade de retornar às fases anteriores de modo a solucionar os problemas identificados. Após a correção dos erros, o software é novamente testado até atingir a qualidade necessária. Este processo contínuo não só garante que o software atenda aos requisitos definidos, como também melhora a experiência do utilizador e a segurança do software. Diferentes testes podem ser feitos, como por exemplo o teste de desempenho que avalia a velocidade e escalabilidade do software sob diferentes condições, o teste funcional que verifica se o software atende aos requisitos definidos, o teste de regressão assegura que as alterações não afetaram negativamente funcionalidades existentes, o teste de segurança identifica possíveis vulnerabilidades e fraquezas, o teste unitário que examina componentes individuais do software, o teste de usabilidade que avalia a interface do utilizador e a experiência geral de utilização.

Após as tradicionais fases do Ciclo de Vida do Desenvolvimento de Software, análise, design, implementação e teste, o produto desenvolvido tem de ser lançado no ambiente do cliente. A fase responsável por este processo denomina-se fase de **produção**. Após esta fase, existe muitas vezes, uma última fase de **manutenção**, onde uma equipa designada de profissionais de desenvolvimento de software corrige eventuais problemas após o lançamento do produto, realiza ajustes de desempenho e atualiza o software para atender às necessidades dos utilizadores. Estas duas fases são muitas vezes adicionadas ao tradicional Ciclo de Vida do Desenvolvimento de Software, como complementares a este, isto pode verificar-se na Figura 3.2.

Produção

Esta fase, geralmente começa com uma réplica do ambiente de produção, permitindo a realização de testes tanto por um grupo designado do público-alvo como pelos profissionais de desenvolvimento de software, utilizando o conceito de User Acceptance Testing (UAT). Na empresa, o lançamento dos produtos é geralmente realizado por membros seniores da equipa de desenvolvimento de software, pois envolve um processo delicado que pode afetar diretamente o ambiente do cliente.

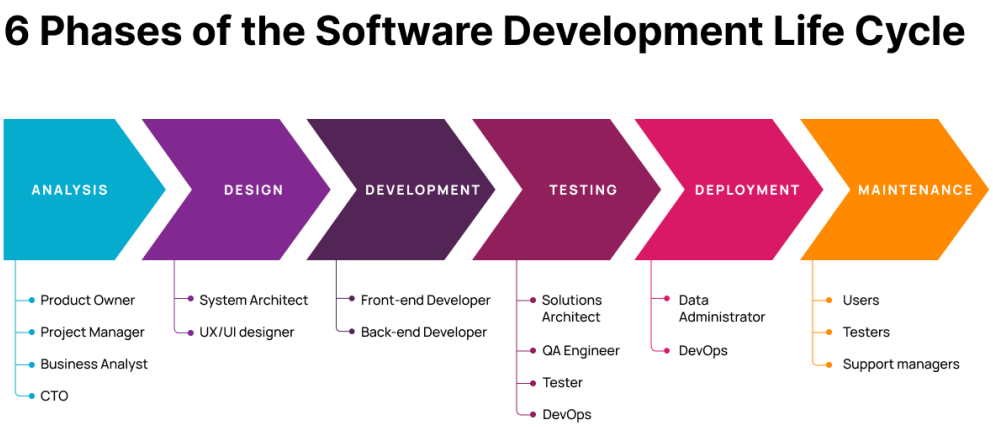


Figura 3.2: Ciclo de Vida do Desenvolvimento de Software com fases complementares de produção e manutenção [17]

O artigo "AgileUAT: A Framework for User Acceptance Testing based on User Stories and Acceptance Criteria" [18] menciona que a fase UAT se encaixa no final do ciclo de desenvolvimento, validando se o software está pronto para o utilizador final. Este artigo, adiciona ainda a ideia de que esta fase valida os requisitos funcionais e não-funcionais antes do sistema ser entregue ao cliente em produção, mostrando como a fase de análise de requisitos é mapeada diretamente para a fase de UAT.

De acordo com [19], o UAT valida se os utilizadores conseguem desempenhar as suas funções utilizando o novo sistema, garantindo que os requisitos foram corretamente identificados, documentados, interpretados e implementados durante todo o ciclo de desenvolvimento. Assim o autor reforça que fase de UAT é uma ferramenta de validação de ponta a ponta, sugerindo que valida todo o ciclo de desenvolvimento, desde a fase de análise até à entrega, essencial para assegurar que as necessidades dos utilizadores foram devidamente transformadas em soluções técnicas.

A fase de UAT pode levar a vários tipos de testes [20], incluindo Operational Acceptance Testing (OAT), que avalia se o software é compatível, confiável e estável o suficiente para ser implementado no ambiente de produção; Beta Testing, onde grupos de utilizadores fornecem os seus comentários ao utilizarem o software; Black Box Testing, que se foca nas funcionalidades do software sem visibilidade do código, validando se o software funciona conforme o esperado; Outros tipos incluem Contract Acceptance Testing, que verifica o cumprimento dos critérios contratuais, e Regulation Acceptance Testing, que valida a conformidade com

regulamentos legais.

Se o cliente estiver satisfeito com os resultados dos testes e confirmar que o produto funciona conforme o esperado, este é efetivamente lançado no ambiente de produção e torna-se acessível para todos os utilizadores finais.

Manutenção

Após o lançamento do produto no ambiente de produção, a manutenção do produto é realizada pelos profissionais de desenvolvimento de software. Isso inclui a resolução de qualquer problema que possa surgir e a implementação de melhorias conforme necessário [21].

A fase de manutenção garante que a solução permaneça funcional, segura e eficiente ao longo do tempo. Na situação do estágio, foi abordada por uma equipa especifica que acompanha os projetos de perto após o seu lançamento denominada Equipa de Suporte.

3.1.2 Modelos de Desenvolvimento de Software

No contexto da Engenharia de Software, diversos **modelos de desenvolvimento** são utilizados para **garantir a entrega de produtos de alta qualidade**. Cada modelo possui **características distintas** que podem ser mais ou menos adequadas dependendo das necessidades e circunstâncias de um projeto específico. Abaixo, são apresentados dois dos modelos mais amplamente utilizados: o **Modelo em Cascata** e o **Modelo Ágil**.

Modelo em Cascata

O Modelo em Cascata é a abordagem mais **tradicional e clássica** de desenvolvimento de software. Foi primeiramente introduzida por **Winston Royce** em 1970, referindo que o desenvolvimento complexo de sistemas de software poderia ser realizado de forma sequencial e de uma só vez, onde todos os requisitos podem ser obtidos no início do processo, depois todo o design é completo e por fim esse design é implementado e testado. Este modelo é caracterizado por ter uma **estrutura linear e sequencial**, onde cada fase do ciclo de vida do desenvolvimento de software deve ser **completada antes que a próxima fase comece**. Esta abordagem resulta num fluxo de trabalho bem **definido e documentado** [16], [22].

Este modelo tem as seguintes características principais:

- As fases do modelo são realizadas de forma sequencial, começando com a análise de requisitos e avançando para design, desenvolvimento, teste, produção e manutenção.

Cada fase deve ser concluída antes que a próxima comece, o que proporciona um controlo rígido e uma gestão mais simples do projeto, facilitando o acompanhamento do progresso e a coordenação das atividades.

- Gera uma documentação extensa em cada fase. Desde o documento de requisitos até aos manuais de utilização, cada fase é bem documentada. Esta documentação detalhada não só facilita a revisão e o acompanhamento do progresso, como também serve como referência para futuras manutenções e atualizações do sistema.
- Todo o planeamento e a definição dos requisitos são realizados no início do projeto. Isso permite um entendimento claro das necessidades e objetivos do sistema desde o princípio, mas com pouca flexibilidade para mudanças subsequentes.
- Adota uma abordagem hierárquica, começando com uma visão geral do sistema e refinando progressivamente para detalhes específicos das tecnologias utilizadas. A ideia é que o sistema seja inicialmente abordado de forma geral, e gradualmente, os detalhes são elaborados e refinados em fases posteriores, garantindo que todos os aspectos do sistema sejam abordados de maneira estruturada e ordenada.

No entanto, apresenta algumas limitações:

- A natureza sequencial do modelo impede a sobreposição de atividades, o que pode causar atrasos se houver necessidade de mudanças.
- O modelo não se adapta facilmente a mudanças nos requisitos ao longo do projeto, o que pode resultar num produto que não atende completamente às necessidades mais atualizadas do cliente.
- O produto final é entregue somente no final do ciclo de desenvolvimento, oferecendo pouca visibilidade ao cliente durante o processo.

Em resumo, o Modelo em Cascata é mais adequado para projetos com **requisitos bem definidos e estáveis**, onde a **previsibilidade e uma extensa documentação** são prioritárias. Exemplo: Desenvolvimento de software para sistemas bancários com requisitos e regulamentos rigorosos.

Modelo Ágil

O Modelo Ágil é uma abordagem mais **moderna e flexível** para o desenvolvimento de software, que se contrapõe ao caráter rígido do Modelo em Cascata. Este é caracterizada

pela sua **adaptabilidade e foco na colaboração contínua** com o cliente, permitindo **ajustes frequentes e incrementais** durante todo o ciclo de vida do projeto [22], [23].

As características principais do modelo ágil são:

- Evita a recolha extensa de todos os requisitos no início do projeto, permitindo que o trabalho comece antes que todos os requisitos estejam completamente definidos, ajustes contínuos são possíveis conforme o desenvolvimento avança.
- O desenvolvimento é dividido em ciclos curtos denominados *sprints*, onde cada ciclo aborda todas e cada uma das diferentes fases. Cada ciclo resulta numa versão funcional do software. Isto permite ajustes rápidos com base no *feedback* contínuo do cliente.
- Prioriza as funcionalidades com maior valor, permitindo que a equipa se concentre nas características mais importantes. A cada iteração, o cliente pode reavaliar e selecionar as funcionalidades com base no retorno sobre o investimento, garantindo que o desenvolvimento se alinhe aos objetivos de negócio.
- A metodologia enfatiza a comunicação constante entre a equipa de desenvolvimento e o cliente. Mudanças são esperadas e bem-vindas, e o projeto pode ser ajustado de acordo com novas informações e *feedback*, garantindo que o software atenda às necessidades reais e em evolução.
- Utiliza abordagens empíricas para acompanhar o progresso, medindo o esforço estimado que a equipa pode completar numa iteração. Com o esforço estabelecido é possível estimar a conclusão do trabalho restante, permitindo previsões sobre a conclusão do projeto e facilitando a gestão de orçamento.

A suas limitações passam pelos seguintes pontos:

- O ênfase é colocada mais na produção de software funcional do que na documentação, o que pode levar a uma menor rastreabilidade.
- A flexibilidade pode resultar num âmbito menos definido e, se não gerida adequadamente, pode levar a mudanças contínuas que afetam prazos e o orçamento.

O Modelo Ágil é ideal para projetos **dinâmicos e complexos** onde os **requisitos podem evoluir** e a **colaboração constante** com o cliente é crucial. Em ambientes de desenvolvimento mais fluidos, a metodologia Ágil permite maior **adaptabilidade** e uma **abordagem centrada no cliente**. Exemplo: Desenvolvimento de aplicações móveis com base no *feedback* contínuo dos utilizadores.

3.2 Modelo Proposto

O modelo de desenvolvimento de software proposto pela empresa em que se realizou o estágio **combina elementos dos métodos em Cascata e Ágil**, aproveitando as caraterísticas de ambos para obter um processo de desenvolvimento eficaz e adaptável.

No **início dos projetos**, adota uma **abordagem em Cascata para a análise de requisitos**. Um grupo destacado de analistas, líderes do projeto e os líderes das equipas de desenvolvimento da empresa fazem um planeamento detalhado, onde todos os **requisitos do projeto são recolhidos e documentados** de forma exaustiva. Este processo permite uma **compreensão clara das necessidades e expectativas** do cliente antes de iniciar o desenvolvimento do sistema. A documentação criada durante esta fase serve como uma base sólida para as fases seguintes, detalhando os objetivos do projeto, as necessidades dos utilizadores e as especificações técnicas necessárias.

Após a análise de requisitos, o modelo utilizado na empresa **transita para uma abordagem mais Ágil**. Nesta fase, as tarefas são organizadas em **sprints**, que são períodos curtos e iterativos, onde cada sprint **aborda uma parte específica do projeto** e resulta, no final de cada ciclo, numa **versão funcional do software**. Em cada sprint, o processo inclui o **design** detalhado das funcionalidades a serem desenvolvidas, este é realizado por profissionais de desenvolvimento de software seniores, como o líder da equipa de desenvolvimento ou o arquiteto. A seguir a **implementação** realizada pela equipa de desenvolvimento e posteriores **testes** pela equipa de controlo de qualidade. Após o testes do produto no ambiente da empresa, a versão funcional do software não é lançada no ambiente do cliente, invés disso é demonstrada no ambiente da empresa, ao cliente, em reuniões gravadas, para **recolher *feedback***, utilizado para refinar e ajustar o desenvolvimento ou mesmo o design do produto, garantindo que o software esteja alinhado com as expectativas.

Quando todos os sprints foram realizados, todas as funcionalidades desenvolvidas e demonstradas, a empresa **lança o produto numa réplica do ambiente de produção** do cliente num contexto de **UAT**, conforme explicado anteriormente na secção 3.1.1. Neste processo, uma equipa de desenvolvimento da empresa trabalha em conjunto com uma equipa de utilizadores destacados, para **testar o produto final** e refinar quaisquer problemas identificados. Quando o **cliente está satisfeito** com o resultado do produto final, este é **lançado no ambiente de produção** para assim ser utilizado pelo cliente.

Após o lançamento, a empresa continua ativamente atenta aos comentários e pedidos do cliente depois da utilização do produto final, numa fase chamada **manutenção**, realizada pela Equipa de Suporte.

Este **modelo híbrido combina a estrutura e o planeamento detalhado do modelo em Cascata com a flexibilidade e a adaptabilidade do modelo Ágil**. A fase inicial, à imagem do modelo em **Cascata**, assegura que todos os **requisitos sejam bem compreendidos e documentados**, enquanto a abordagem **Ágil** subsequente permite **ajustes rápidos e contínuos** durante o desenvolvimento. Isto resulta num processo que é tanto **bem planeado quanto adaptável** às mudanças, garantindo a entrega de um produto de qualidade que atende às necessidades reais dos clientes. A **fase de UAT** permite validar o produto final com utilizadores reais, garantindo que o software atenda às expectativas e requisitos reais antes do lançamento oficial. A **fase de manutenção** assegura que, após o lançamento do produto, qualquer problema ou necessidade de melhoria identificada pelos utilizadores seja resolvida de forma rápida e eficiente.

3.3 Aplicações Desenvolvidas/Abordadas

Nesta secção descreve-se, de uma forma geral, o contexto dos projetos, em que se inseriu o estágio, a metodologia utilizada, os objetivos das soluções e os seus requisitos.

A metodologia dos projetos seguiu o modelo iterativo de desenvolvimento de software proposto pela empresa, explicado na secção anterior 3.2. O processo começa com a angariação de requisitos, seguida por uma fase de design pelos arquitetos e líderes da equipa de desenvolvimento. As tarefas são depois atribuídas e executadas ao longo das semanas seguintes, em forma de sprints, para no final serem testadas e apresentadas ao cliente.

3.3.1 Aplicação de Gestão de Cartões de Negócio Digitais - Cartões Horizontais

A solução Aplicação de Gestão de Cartões de Negócio Digitais foi desenvolvido para alinhar o cliente com a tendência crescente da digitalização, substituindo os seus tradicionais cartões empresariais físicos por cartões digitais interativos. Este projeto visa a criação de uma aplicação móvel que permita aos funcionários da empresa cliente criar, administrar e personalizar os seus cartões de negócio através de uma aplicação móvel em Power Apps e partilhá-los de forma independente à plataforma.

Objetivos

Através da solução apresentada tem-se como objetivo:

- Reduzir o custo da empresa, pois os atuais cartões físicos têm um preço de fabrico superior à criação de cartões digitais;
- Facilitar atualização dos dados, sem a necessidade de imprimir de novo os cartões se houver algum erro na informação ou se a informação precisar de ser atualizada;
- Organização dos dados, já que o cliente atualmente utiliza os serviços de armazenamento de dados da Microsoft 365 e estes podem ser diretamente utilizados na solução;
- Segurança, tirando partido dos benefícios das políticas de segurança do Microsoft 365;

Requisitos Funcionais

De seguida, são enumerados os requisitos funcionais da aplicação, estes descrevem as funcionalidades específicas que a aplicação deve oferecer:

- Dados como o nome do empregado, o nome de preferência, cargo oficial, departamento, sub-grupo, número de telefone, número de telefone fixo, número de telemóvel, número de fax, endereço eletrónico, imagem do empregado e morada devem ser automaticamente populadas a partir da atual estrutura de dados do cliente.
- O utilizador deve poder visualizar os cartões digitais em Inglês e em Árabe.
- O utilizador deve poder adicionar o cartão digital na carteira virtual do seu smartphone (Google ou Apple).
- Os utilizadores devem poder partilhar os seus cartões digitais através do Outlook.
- Os destinatários da partilha do cartão digital devem ser providenciados com um *link* para visualização do mesmo, o qual deve ainda permitir o seu *download* e salvar os dados do contato diretamente no seu dispositivo.
- O *link* que contém o cartão digital deve ser disponibilizado em hiperligação HTTP ou código QR.

Requisitos Não Funcionais

Abaixo são apresentados os requisitos não funcionais da aplicação, estes não estão diretamente relacionados a funcionalidades, mas afetam a qualidade e o desempenho do sistema.

- O design e esquema de cores dos diferentes *templates* dos cartões devem estar alinhado com os fornecidos pelo cliente.

- A aplicação deve ser otimizada para telemóvel.

Metodologia

O desenvolvimento da aplicação móvel foi realizado ao longo de todos os sprints do projeto, mas as tarefas associadas a este desenvolvimento não estavam incluídas no âmbito principal deste estágio.

Dentro do âmbito do estágio, o primeiro sprint, com duração de duas semanas, teve como objetivo a criação e adaptação de todos os *templates* dos cartões digitais utilizando as tecnologias HTML e CSS. Estes *templates* foram preparados para as plataformas que disponibilizam os cartões, que incluem uma aplicação móvel desenvolvida em Power Apps e uma aplicação Web. No sprint seguinte, com duração de uma semana, foi concebido o *front-end* e o *back-end* de uma aplicação Web para facilitar a partilha dos cartões, sem a necessidade de instalar a aplicação móvel. A adaptação da disposição das informações dos cartões, nas aplicações móvel e Web para o dinamismo dos dados, foi realizada num sprint de mais duas semanas. A integração de um código QR, para facilitar a partilha dos cartões digitais, e a funcionalidade na aplicação Web de salvar a informação dos cartões no dispositivo do utilizador foram realizado em sprints separados de uma semana cada.

Após cada iteração, foram realizados testes e ajustes para garantir que todas as funcionalidades estivessem operacionais e alinhadas com os requisitos. Após todos os sprints, deve encaminhar-se a aplicação para produção, de modo a ser testada por clientes.

3.3.2 Aplicação de Gestão de Cartões de Negócio Digitais - Cartões Verticais

Este projeto pretende a cópia e edição da aplicação Power Apps de cartões de negócios digitais, realizada anteriormente, para uma demonstração destinada a um novo cliente. Para isto é necessário copiar a aplicação atual e adaptá-la com os detalhes apropriados.

Este pedido implica a adaptação dos detalhes visuais na aplicação móvel em Power Apps, como cores e logotipos, para corresponder à nova empresa. Envolve também a realização de uma nova plataforma Web para apresentar os cartões digitais e a sua conexão com a aplicação móvel.

Adicionalmente, o novo cliente expressou a necessidade de um cartão digital de negócios mais moderno no formato vertical, em contraste com o formato horizontal clássico previamente utilizado pelo cliente anterior.

Dito isto, como a aplicação a ser demonstrada é uma cópia de uma aplicação já desenvolvida, com a única distinção no design dos cartões virtuais, tanto os objetivos da solução,

como os seus requisitos passam pelos mesmo pontos da solução anteriormente descrita na sub-secção 3.3.1. A única distinção entre estas é nos requisitos não funcionais, já que na aplicação anterior, fazia parte o requisito “O design e esquema de cores dos diferentes *templates* dos cartões devem estar alinhado com os fornecidos pelo cliente”, nesta nova aplicação, o design dos cartões é um processo livre, atribuído inteiramente ao âmbito do estágio. Dito isto, o requisito análogo ao referido para a aplicação atual será: “O esquema de cores dos diferentes *templates* dos cartões deve estar alinhado com os fornecidos pelo cliente”

Após os requisitos e o design da aplicação móvel definidos, é necessário estruturar o design do cartão digital vertical. Por se tratar de uma adaptação de uma solução já existente, para uma demonstração a um novo cliente, a cópia, edição e testes da aplicação anterior, o design do cartão digital vertical e o seu desenvolvimento precisam de ser o mais breves possíveis. Para todo o processo atribui-se um sprint de duas semanas para todo o desenvolvimento da solução. Como a aplicação irá ser utilizada apenas para uma demonstração, não há necessidade de recorrer às fases de produção e manutenção.

3.3.3 Aplicação de Gestão de Auditorias

Todo o processo de verificação e gestão de auditorias no terreno exige uma grande coordenação, planeamento e a utilização de várias ferramentas. A solução, proposta visa atender a esses requisitos, criando um aplicação móvel que agilize este processo de forma eficiente e integrada.

Objetivos

A solução apresentada visa alcançar os seguintes objetivos:

- Digitalizar listas de verificação e inspeções o que simplifica e normaliza os procedimentos, reduzindo a burocracias e erros.
- Garantir a recolha os dados de forma precisa, permitindo que pessoal autorizado aceda às listas de verificação e às inspeções em tempo real.
- Facilitar a análise avançada, visualização e criação de relatórios de dados recolhidos, promovendo a tomada de decisões baseadas em dados e ações proativas.

Requisitos Funcionais

Segue-se os requisitos funcionais da aplicação, estes definem o comportamento e as funcionalidades principais do sistema:

- Um utilizador pode pertencer a apenas um departamento.
- Cada departamento terá 3 funções: Oficial, Supervisor e Gerente de Plantão.
- Inspeções podem ser atribuídas a apenas uma pessoa.
- Os utilizadores devem ver apenas inspeções que se relacionam com eles.
- Uma inspeção pode ser planeada com antecedência ou realizada de forma espontânea.
- Ao executar uma inspeção, o utilizador deve ser capaz de ver todas as questões abertas para aquele tipo de inspeção.
- Deve ser possível realizar inspeções usando um "sistema de luz verde", o que significa:
  - Por padrão, todos os itens são considerados em conformidade.
  - O inspetor deve, portanto, apenas alterar as declarações para as quais o padrão não é atendido.
- Haverá seções e subseções de perguntas para cada modelo de inspeção.
- Para cada declaração/pergunta, os botões "Observação", "Comentário"e "Foto"serão configuráveis.
- Pode haver várias observações por pergunta/declaracão.
- Ao aceder ao ecrã de "Observações", os botões exibidos sempre serão:
  - Ações
  - Fotos
  - Geo-localização
  - Perigos
- Pode haver várias ações por observação.
- O status das ações pode ser:
  - Realizada

  - Necessária

- Se uma observação é feita e tem uma ação necessária, um problema será criado quando a inspeção for para o status "Fechado".
- As ações terão um nível de risco associado, que será padronizado nas seguintes opções (escolha única):
  - Baixo
  - Médio
  - Alto
- Este nível de risco é usado para estabelecer o prazo de resolução para o problema criado.
- Pode haver vários perigos por observação.
- Os perigos terão um nível de risco associado, que será o mesmo que o das ações.
- As categorias de perigos serão padronizadas nas seguintes opções:
  - Biológico
  - Químico
  - Ergonômico
  - Saúde
  - Físico
  - Segurança
  - Vida selvagem
  - Organização do trabalho
  - Outro
- O fluxo de aprovação da inspeção será configurado por modelo.
- Os aprovadores escolhidos podem reatribuir a tarefa para outros utilizadores.
- Uma inspeção pode ser cancelada em qualquer estágio pelos seguintes utilizadores/-perfis:
  - O utilizador que a criou

  - Os Gerentes de Serviço do departamento de inspeção

- Após o fecho de cada inspeção, um relatório em PDF com as respetivas informações deve ser gerado e enviado ao oficial, Supervisores e Gerentes de Serviço.

Requisitos Não Funcionais

De seguida os requisitos não funcionais, ou seja os que definem as características de qualidade e restrições que a aplicação deve atender:

- A aplicação de inspeções utilizará o *layout* móvel.

Metodologia

O âmbito do estágio neste projeto, foi iniciada numa fase tardia, quando foi necessário a realização do relatório final a enumerar todos os problemas relatados na aplicação. Projetou-se realizar uma automatização em Power Automate para a criação dinâmica do relatório em HTML e posteriormente convertê-lo para PDF. Dado a boa exibição na construção dos *templates* dos cartões digitais em HTML, os gestores do projeto definiram que o desenvolvimento, testes e entrega desta tarefa deveria ser realizado num sprint com duração de duas semanas. Após este sprint a aplicação deve partir para produção, sendo testada em UAT.

3.3.4 Automatização de Agendamento de Reuniões Internas

A solução pretende automatizar o agendamento de reuniões internas semanais denominadas "Weekly Coffee Break". Os colaboradores devem ser agrupados aleatoriamente todas as semanas, por forma a criar reuniões para uma interação contínua e espontânea entre os diferentes membros da equipa, promovendo uma melhor comunicação e colaboração dentro da empresa.

A solução deve integrar-se diretamente com os sistemas de calendário e comunicação da empresa, garantindo que todos os participantes recebam convites e notificações apropriadas sem a necessidade de intervenção manual.

Objetivos

Através da automatização apresentada, pretende-se alcançar os seguintes objetivos:

- Facilitar conversas informais e discussões sobre temas atuais, problemas e soluções, contribuindo para uma melhor comunicação entre os colaboradores.

- Promover a interação entre os diferentes membros da equipa, ajudando os colaboradores a conhecerem-se melhor e a colaborarem de forma mais eficaz.
- Automatizar o processo de agendamento de reuniões para evitar a necessidade de coordenação manual e garantir a inclusão de todos os colaboradores.

Requisitos Funcionais

De seguida serão enumerados os requisitos funcionais da automatização, estes descrevem as funcionalidades específicas que a solução deve oferecer:

- A automatização deve agrupar aleatoriamente os colaboradores para reuniões semanais, garantindo que diferentes pessoas sejam incluídas em cada semana.
- O sistema deve enviar convites automáticos para todos os participantes, incluindo detalhes da reunião, data e hora.
- Os convites e notificações devem ser integrados com o calendário e o sistema de comunicação da empresa.
- A solução deve permitir a visualização e o ajuste dos grupos de participantes, caso haja necessidade de modificações.
- As reuniões devem ser agendadas num horário que maximize a participação e a disponibilidade dos colaboradores.

Requisitos Não Funcionais

A seguir, estão os requisitos não funcionais, que não estão diretamente ligados às funcionalidades da automatização, mas influenciam sua qualidade e desempenho:

- A automatização deve ser eficiente e executar o agrupamento e o envio de convites sem atrasos significativos.
- A interface de gestão da automatização deve ser intuitiva e de fácil utilização para os administradores responsáveis pela configuração e monitoramento dos agendamentos.

Metodologia

Como a solução irá ser desenvolvida para uma utilização interna, definiu-se um prazo de uma semana para o seu design, implementação, testes e envio para produção da automatiza-

ção em Power Automate. Estima-se que este tempo é suficiente para atender às necessidades do projeto e garantir que o sistema fique pronto para utilização imediata.

3.3.5 Sistema Dinâmico de Gestão Notificações e Tarefas

O cliente atualmente espera por uma aplicação para verificação e gestão de problemas por parte da empresa, Make The Shift. No contexto do estágio, é pretendido um sistema dinâmico, complementar à aplicação, que terá de ser capaz de capturar alterações realizadas na aplicação e, com base nessas ações, enviar notificações apropriadas aos destinatários, criar novas tarefas ou completar automaticamente tarefas existentes.

Este processo assegura uma gestão eficiente das tarefas e uma comunicação ágil entre os envolvidos, eliminando a necessidade de acompanhamento manual constante da aplicação e reduzindo o risco de falhas na execução das tarefas.

Objetivos

Os objetivos principais da solução são:

- Automatizar o envio de notificações para os destinatários relevantes com base em alterações específicas em SharePoint.
- Criar tarefas pendentes associadas às ações realizadas na aplicação, assegurando que nenhum passo importante seja ignorado.
- Completar automaticamente as tarefas quando as condições definidas são satisfeitas, reduzindo a necessidade de intervenção manual.

Requisitos Funcionais

Abaixo seguem os requisitos funcionais do sistema, estes descrevem as funcionalidades específicas que o sistema deve oferecer:

- Enviar notificações automáticas para os destinatários designados, com base nas ações realizadas na aplicação.
- Criar tarefas pendentes quando determinadas ações são realizadas na aplicação, com a possibilidade de definir prazos e responsáveis.
- Completar automaticamente as tarefas criadas quando as condições para a sua conclusão forem cumpridas.

- Fácil customização, permitindo que o cliente adicione ou modifique notificações e tarefas conforme necessário, sem exigir conhecimentos técnicos avançados.

Requisitos Não Funcionais

De seguida, os requisitos não funcionais, que descrevem as restrições ou características de qualidade do sistema:

- Altamente responsiva, garantindo que as notificações e tarefas sejam geradas e enviadas em tempo real.
- Integrar-se perfeitamente com o ambiente do SharePoint e outras ferramentas do Microsoft 365 utilizadas pelo cliente.
- Deve ser escalável, suportando um aumento no número de utilizadores e na quantidade de dados monitorados sem perda de desempenho.
- Garantir a segurança dos dados, respeitando as políticas de privacidade e controlor de acesso estabelecidas pelo cliente.

Metodologia

Uma primeira parte do projeto, focado na implementação do sistema de captura de eventos será pretendida num primeiro sprint de duas semanas. Seguido de um sprint de três semanas dedicado à conclusão do sistema e incorporação de condições mais complexas conforme necessário. Um último sprint de duas semanas levará a cargo a conclusão da introdução e configuração de todos os dados necessários ao sistema. Após cada sprint são necessário testes assegurando o completo e eficaz funcionamento do sistema. Após o último sprint a aplicação deve ser conduzida para produção, de modo a ser testada no ambiente do cliente na fase de UAT.

Capítulo 4

Implementação

Neste capítulo, apresenta-se detalhadamente o processo de implementação dos projetos durante o estágio na empresa Make The Shift. Cada projeto apresentou desafios únicos, abordagens e técnicas distintas para transformar os requisitos identificados em soluções funcionais. A fase de implementação foi caracterizada pela utilização das tecnologias de desenvolvimento da Microsoft, incluindo ferramentas da Microsoft 365, como Power Apps para o desenvolvimento de aplicações móveis, Power Automate para automatizações de fluxos de dados e SharePoint para armazenar dados e como interface de sistemas. Além disso, recorreu-se à plataforma Microsoft Azure App Service, utilizando a sua tecnologia Azure Web Apps para a realização tanto do *front-end* como do *back-end* de um aplicação Web, e a a tecnologia Azure Functions, para invocar "micro-serviços" através de *triggers*. Também se utilizaram as tecnologias de desenvolvimento Web HTML, CSS, Javascript e .NET.

A fase de desenvolvimento dos projetos foi realizado em conformidade com o modelo de desenvolvimento de software adotado pela empresa e explicado no capítulo secção 3.2. Este baseai-se em ciclos iterativos, com foco em sprints, que permitem uma abordagem ágil e controlada na execução das tarefas. A implementação não apenas envolveu o desenvolvimento das funcionalidades solicitadas, como também a resolução de problemas, integração de sistemas e otimização das soluções para garantir um desempenho robusto, alinhado com as expectativas dos clientes.

A seguir, descreve-se em detalhe o processo de abordagem aos requisitos, as soluções desenvolvidas, os desafios encontrados e as estratégias adotadas para superá-los.

4.1 Aplicação de Gestão de Cartões de Negócios Digitais - Cartões Horizontais

Esta aplicação móvel foi desenvolvida como uma solução que visa aos funcionários da empresa cliente criar, gerir, personalizar e partilhar cartões de negócios digitais de forma eficaz, aproveitando a infraestrutura da Microsoft 365.

4.1.1 Requisitos Atribuídos

Numa fase inicial, não foi âmbito do estágio o desenvolvimento de uma solução completa. Em vez disso, o trabalho foi focado em atender apenas alguns dos requisitos da solução, nomeadamente:

Requisitos Funcionais

Estes requisitos definem as funcionalidades que a aplicação deve oferecer:

- O utilizador deve poder visualizar os cartões digitais em Inglês e em Árabe.
- Os destinatários da partilha do cartão digital devem ser providenciados com um *link* para visualização do mesmo, o qual deve ainda permitir o seu *download* e salvar os dados do contato diretamente no seu dispositivo.
- O *link* que contém o cartão digital deve ser disponibilizado em hiperligação HTTP ou código QR.

Requisitos Não Funcionais

Estes requisitos especificam detalhes sobre o design e a performance da aplicação:

- O design e esquema de cores dos diferentes *templates* dos cartões devem estar alinhado com os fornecidos pelo cliente.
- A aplicação deve ser otimizada para telemóvel.

4.1.2 Abordagem para o Desenvolvimento

O projeto inseriu-se numa fase inicial do estágio, onde as habilidades em HTML e CSS, adquiridas durante a licenciatura e mestrado em Engenharia Informática e Multimédia, foram aproveitadas para desenvolver oito *templates* de cartões virtuais. Cada *template* foi desenvolvido para um departamento específico da empresa cliente e disponibilizado em duas versões: inglês e árabe. Cada versão incluí a direção de escrita apropriada e os logotipos definidos pela empresa.

Apesar da solução passar pelo desenvolvimento de uma aplicação móvel recorrendo à plataforma Power Apps, houve a necessidade do desenvolvimento de uma aplicação Web, suportada pela da framework .NET, com recurso à plataforma Azure Web Apps, de modo a facilitar e agilizar a partilha dos cartões virtuais, sem ter que instalar a aplicação móvel.

De modo a integrar dinamicamente os dados nos cartões HTML/CSS, houve necessidade de realizar alguma lógica no *backend* do projeto .NET, utilizando a linguagem C#, para manipular o conteúdo dos cartões. Esse processo, envolveu a recuperação de dados armazenados no SharePoint, via uma API Microsoft Graph, e o desenvolvimento de algoritmos específicos para resolver problemas de *layout*, como colapsar linhas de informação vazias e ajustar o conteúdo de acordo com o espaço disponível. Esse mesmo dinamismo foi aplicada na *app* desenvolvida em Power Apps, através da linguagem PowerFX.

Para permitir que os cartões digitais fossem salvos pelo utilizador, realizou-se a conversão das suas informações para ficheiros vCard e adicionou-se botões à aplicação Web, que através de Javascript, ativam o *download* das informações do utilizador.

Com a necessidade de integrar, na aplicação móvel, um código QR para facilitar a partilha dos cartões digitais, foi implementada uma função Azure. Esta, gera o código QR contendo o *link* para a aplicação Web, onde estão disponíveis os cartões digitais.

4.1.3 Soluções Implementadas e Desafios Encontrados

Durante o desenvolvimento do projeto, várias soluções técnicas foram implementadas para atender aos requisitos definidos e superar os desafios encontrados.

Recriação dos *templates* em HTML/CSS

O cliente deseja substituir os cartões de negócio físicos por versões digitais.

Assim, a primeira tarefa no contexto do estágio é replicar fielmente os oito *templates* existentes usando HTML e CSS. Cada *template* foi facultado, em formato de imagem, pelo cliente e tem de ser desenvolvido nas duas versões disponíveis: inglês e árabe. Cada versão tem de ser ajustada para a direção de escrita apropriada e incorporando os logotipos definidos pela empresa. Os cartões digitais têm de ter em conta o tamanho e a fonte dos títulos, subtítulos e texto, a palete de cores disponibilizada pelo cliente no documento de design da empresa e ainda os logótipos e imagens fornecidos.

A solução envolve a apresentação dos cartões digitais tanto na aplicação móvel, desenvolvida em Power Apps por outros colegas inseridos no mesmo projeto, como também num Website recorrendo à plataforma Azure Web Apps, parte integrante do estágio. Isto evita a necessidade da instalação da aplicação móvel ao partilhar os cartões digitais.

O desenvolvimento dos cartões em HTML e CSS para o Website foi relativamente direto, dado ao histórico do mestrado nessas tecnologias. No entanto, integrar esses cartões na Power Apps apresentou desafios adicionais. No ambiente Power Apps, os cartões tinham que ser renderizados utilizando um controlador de texto HTML (*HTML text controller*). Embora o controlador permita a utilização de HTML e CSS, possui limitações significativas, suportando apenas um conjunto básico de *tags* e propriedades CSS, o que restringe o uso de disposições flexíveis, tabelas, animações, e outras funcionalidades avançadas disponíveis nos navegadores Web. Estas restrições impostas pelo controlador HTML na aplicação Power Apps resultaram em dificuldades na recriação exata dos modelos disponibilizados pelo cliente. Após ajustes e otimizações, todos os 16 *templates* foram implementados com sucesso.

No passado, a empresa tinha sofrido com problemas de consistência, ao terem uma solução com a mesma funcionalidade desenvolvida em plataformas diferentes. Quando havia uma alteração na funcionalidade, a mesma alteração tinha de ser realizada em ambas as plataforma levando obviamente mais tempo de desenvolvimento. Dito isto, decidiu-se que para manter a consistência visual entre a página Web e a aplicação móvel, teria de se utilizar o mesmo código HTML/CSS, para a criação dos cartões digitais, nas duas plataformas. De acordo com este critério, foi utilizado o código mais básico (o criado para a aplicação Power Apps) nas duas plataformas. Uma das limitações do controlador é a impossibilidade de adicionar um ficheiro com estilo CSS, ou seja, todo o estilo dos cartões tem de ser adicionado dentro das *tags* HTML.

Testes foram constantemente realizados ao longo do desenvolvimento, e no final do sprint, a versão digital dos cartões foi apresentada ao cliente, que expressou grande satisfação com o resultado.

Abaixo encontra-se a explicação de como foi desenvolvido um dos *templates* dos cartões virtuais:

No cabeçalho do documento HTML, identificado pela marca como *<head>*, são adicionados os metadados que embora não sejam exibidos diretamente na página, são essenciais para o seu funcionamento correto. Neste caso, o atributo *name="viewport"* no elemento *<meta>* define propriedades da *viewport*, a área visível de uma página Web num dispositivo, como um smartphone ou monitor. O atributo *content="width=device-width, initial-scale=1"* especifica que a largura da *viewport* deve ser igual à largura do dispositivo, ajustando o *layout* para caber no ecrã, e define que o nível de *zoom* inicial deve ser 1, exibindo a página na escala padrão sem ampliação. Sem estas configurações, a página poderia aparecer ampliada ou cortada em dispositivos móveis.

No corpo do documento HTML, inicia-se com um elemento * * mais externo, de modo a criar, com a ajuda de CSS, um fundo branco, uma sombra e cantos arredondados, proporcionando ao elemento uma aparência de cartão.

Como uma das limitações do controlador de texto HTML da tecnologia Power Apps é a impossibilidade de utilizar *media queries*, a responsividade do cartão é conseguida através de dimensões percentuais. Por exemplo, as propriedades CSS *padding* e *margin*, que definem o espaço à volta dos elementos HTML, e *width*, que define a largura dos elementos, são dimensionadas em percentagem. Isso permite que o *layout* se adapte a diferentes tamanhos de ecrã.

Dentro desse primeiro * *, foi colocado o logotipo, no formato de imagem flutuante à direita, com uma largura máxima também em percentagem. Isto permite que a imagem se redimensione conforme o tamanho do cartão. Esta imagem é incluída no HTML com uma codificação em base64, uma vez que o logotipo é fixo para cada *template*, e pretende-se incorporar a imagem diretamente no HTML sem depender de ficheiros externos.

Por baixo do logotipo do departamento, foram introduzidos os detalhes do colaborador, como nome (exibido com uma fonte maior e em negrito), função e departamento (exibidos por baixo do nome em texto ligeiramente menor).

Por baixo destas três informações encontram-se dois grupos lado a lado com informações pessoais, como exemplificado na Figura 4.1. O primeiro grupo contém números de telefone, número de telemóvel e e-mail, todas em diferentes linhas, cada uma destas informações contém um prefixo indicativo do tipo de informação. Cada informação deste grupo está estruturados com a propriedade *flexbox* para alinhamento. E com a propriedade *clear: both*, para que apenas a devida informação e o seu prefixo fiquem lado a lado, removendo qualquer elemento

à esquerda ou à direita destes, movendo-o automaticamente para a linha abaixo.

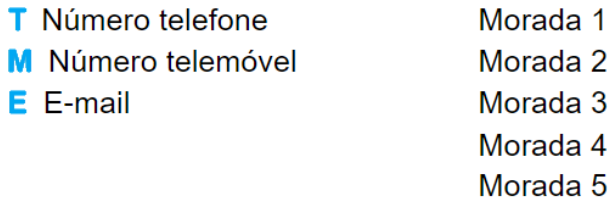


Figura 4.1: Exemplo da disposição das informações pessoais lado a lado, nos modelos de cartão de negócio digital, versão inglesa

O segundo grupo de informações pessoais, no lado oposto, contém cinco linhas de moradas. Estas estão centradas num elemento *div* com uma percentagem complementar da propriedade CSS *width*, complementar à do primeiro grupo, de modo a ocupar o espaço disponível a seguir a este, ficando ambos os grupos de informações lado a lado responsivamente.

O código, explicado anteriormente, utilizado para criar os dois grupos de informações pessoais, lado a lado, está presente na Listing 4.1.

Listing 4.1: HTML e CSS utilizados para a realização da disposição de informação lado a lado, num dos cartões de negócio digitais, versão inglesa

```
<div style="clear:both;margin-top: 5%;">
    <div style="float: left;width: 50%;">
        <div style="display: flex; align-items: start; margin-top: ...;
            margin-bottom: ...; clear:both;">
            <b style='color: ...; margin-right: ...;'>T</b> Employee Phone 1
        </div>
        <div style="display: flex; align-items: start; margin-top: 3px;
            margin-bottom: ...; clear:both;">
            <b style='color: ...; margin-right: 7px;'>M</b> Employee Phone 2
        </div>
        <div style="display: flex; align-items: start; margin-top: ...;
            margin-bottom: ...; clear:both;">
            <b style='color: ...; margin-right: ...;'>E</b> Employee Email
        </div>
    </div>
    <div style="float: left;width: 50%;">
        <p style="font-size: ...;padding: 0;margin-top: ...;margin-bottom:
            ...;">Employee Address 1</p>
```

```
        <p style="font-size: ...;padding: 0;margin-top: 3px;margin-bottom:
            ...;">Employee Address 2</p>
        <p style="font-size: ...;padding: 0;margin-top: 3px;margin-bottom:
            ...;">Employee Address 3</p>
        <p style="font-size: ...;padding: 0;margin-top: 3px;margin-bottom:
            ...;">Employee Address 4</p>
        <p style="font-size: ...;padding: 0;margin-top: 3px;margin-bottom:
            ...;">Employee Address 5</p>
    </div>
</div>
```

Finalmente o rodapé contém uma barra com cantos arredondados na parte inferior, consistente com o esquema de cores do cartão. Inclui também um *link* para o site da empresa, com estilo na cor do tema e a negrito para manter a consistência com os *templates* físicos.

Cada versão dos *templates* criados foram introduzidos numa lista Sharepoint "Card Templates". Esta lista tem 3 colunas, uma que é *unique*, ou seja que não pode ter conteúdo repetido, com o nome do *template*. Seguida de outras duas, "EnglishHTML" e "ArabicHTML", onde é armazenado o código HTML da versão em inglês do template e a versão em árabe, respetivamente.

Assim converteu-se, de forma bem sucedida, todos os *templates* fornecidos pelo cliente para uma plataforma digital.

Website com Azure App Service

Após o primeiro sprint, é necessária a criação de uma aplicação Web. Esta aplicação não só disponibiliza os cartões digitais de forma acessível e segura, sem a necessidade de instalação da aplicação móvel, como também oferece um *link* que facilita a partilha dos cartões.

Como a empresa em que se inseriu o estágio é uma Microsoft Gold Partner, está comprometida em adotar e implementar as melhores práticas e tecnologias oferecidas pela Microsoft. Dito isto, optou-se por uma solução utilizando a plataforma para a criação e implantação de aplicações Web da Microsoft, Azure Web Apps.

A plataforma oferece um ambiente de gestão de serviços na nuvem através de uma aplicação Web intuitiva e fácil de usar, disponível em `https://portal.azure.com`. Esta plataforma possibilita a criação e a gestão de aplicações Web possibilitando a utilização de várias frameworks e linguagens de desenvolvimento.

Assim, foi proposto um projeto utilizando a framework da Microsoft para desenvolvimento Web, ASP.NET. Esta framework utiliza a linguagem de programação *C#* para o desenvolvimento do *backend*. O desenvolvimento foi realizado no ambiente Visual Studio, já que esta tecnologia oferece uma integração eficiente com o ambiente. Entre as opções disponíveis, foi utilizado o GitHub para sincronizar o código com a Azure Web App.

O projeto segue o padrão *Model-View-Controller* (MVC) [24]. Este padrão de arquitetura promove uma separação clara entre três componentes principais: o controlador (em inglês, *controller*), que lida com a lógica da aplicação; a vista (em inglês, *view*), que é responsável pela interface do usuário; e o modelo (em inglês, *model*), que gere os dados. No padrão MVC, os pedidos Web são encaminhados para o controlador, que executa ações e manipula dados por meio do modelo. O controlador seleciona a vista apropriada para exibir os dados e fornece as informações do modelo necessárias a esta. A vista, por sua vez, é responsável por renderizar a página Web com base nos dados que o controlador obtém do modelo. Este fluxo de dados do padrão MVC pode ser visualizado em gráfico na Figura 4.2. Este padrão facilita a manutenção e escalabilidade do sistema, promovendo uma organização mais eficiente do código.

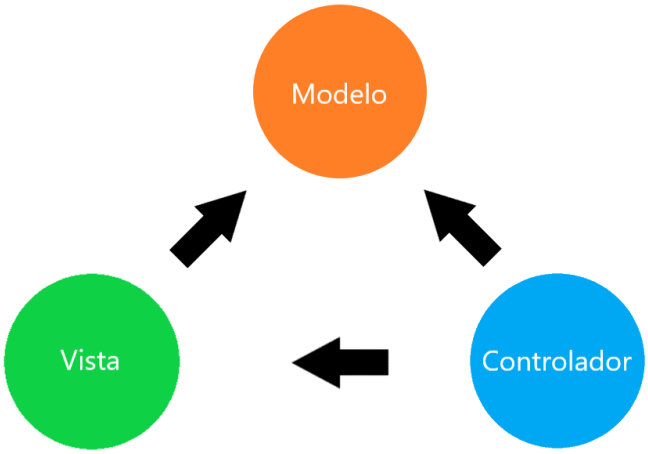


Figura 4.2: Fluxo de dados do padrão MVC

Para aceder aos dados do Microsoft 365 do cliente, utilizou-se o Microsoft Graph, uma API poderosa com caraterísticas RESTful. Esta API permite interagir com vários serviços e dados do Microsoft 365, incluindo SharePoint, onde os dados do cliente utilizados nos cartões digitais estão armazenados.

Além disso, para proteger as credenciais utilizadas para conectar a aplicação ao Micro-

soft Graph, utilizou-se o Azure Key Vault. Esta tecnologia permite armazenar e gerir com segurança credenciais de acesso, chaves e certificados, que são usados para autenticar a aplicação através do Azure Entra ID (solução de gestão de identidade e acesso baseada na nuvem da Microsoft). Com o Azure Key Vault, garante-se que as informações sensíveis permaneçam protegidas contra acessos não autorizados e que as operações de autenticação sejam realizadas de maneira segura.

A aplicação Web foi adaptada de um projeto já existente na empresa, ou sejam a estrutura de pastas e ficheiros de acordo com o padrão arquitetônico MVC já se encontrava criada. Duas funções essenciais para a comunicação segura com o Microsoft Graph, **GetVaultKey()** e **SetGraphServiceClient()**, já se encontravam desenvolvidas no controlador do projeto. Esse mesmo controlador continha ainda a função **GetUserGraph()** e **GetStructValue()**. A primeira utiliza a API do Microsft Graph para recuperar as informações do cliente e a segunda auxilia a primeira estruturando os dados de forma mais simples.

O projeto não poderia deixar de ter um ficheiro ***Web.config***, que contém as diversas definições e dependências da aplicação, como a configuração do Azure Key Vault e dados essenciais para a integração com o Microsoft Graph.

Todavia, foi fundamental **estudar e compreender completamente a estrutura e o código existente**, não apenas para garantir que as alterações fossem realizadas de maneira correta e eficiente, mas também para facilitar futuras resoluções de problemas ou manutenção.

Para entender completamente a parte do projeto já desenvolvida, foi utilizada principalmente a plataforma de aprendizagem Microsoft Learn, que oferece uma vasta gama de recursos educacionais e cursos sobre produtos e tecnologias da Microsoft. Em particular, foram consultados os seguintes tutoriais:

- “Visão geral do ASP.NET Core MVC” [25]
- “Deploy a Website to Azure with Azure App Service” [26];
- “Tutorial: Use a managed identity to connect Key Vault to an Azure Web app in .NET” [27]
- “What are managed identities for Azure resources?” [28]
- “Use managed identity for authentication among Azure App Service/Functions” [29]

Controlador

Alinhando o projeto num padrão **MVC** é necessário um **controlador**. O controlador têm o objetivo de tratar os pedidos do utilizador, manipulando os dados através do **modelo** de

modo a devolver a resposta apropriada por meio de uma **vista**. No projeto, o controlador está localizado no arquivo HomeController.cs.

Na função **GetVaultKey()** são obtidos os segredos essenciais à autenticação, armazenados no *Azure Key Vault*. Esta função já se apresenta desenvolvida na totalidade.

Inicialmente, a função cria uma instância da classe `DefaultAzureCredential()`, disponível na biblioteca *Azure SDK*, que fornece uma credencial para uma autenticação segura com os serviços Azure. Esta autenticação baseia-se na funcionalidade *Azure Managed Identity*, que pode assumir duas formas:

- *System-assigned Managed Identity*: Uma identidade atribuída automaticamente a um recurso (neste caso, à aplicação Web). Esta identidade é excluída automaticamente quando o recurso é removido, tornando-a uma opção económica e segura para autenticação.
- *User-assigned Managed Identity*: Um identidade vinculada a um utilizador específico e que pode ser atribuída a vários recursos. Esta, ao contrário da anterior, não é excluída, quando o recurso é removido.

Para este projeto, a *System-assigned Managed Identity* é mais vantajosa devido à sua simplicidade e custo reduzido. Para recuperar os segredos armazenados no *Azure Key Vault*, a função cria um cliente (com a classe `SecretClient()`), utilizando o URL do *Key Vault* (presente no ficheiro `Web.config`) e a credencial obtida. A função utiliza este cliente e um identificador específico presente no controlador, para recuperar o segredo desejado de forma assíncrona.

A função **SetGraphServiceClient()** é responsável por configurar e retornar uma instância da classe `GraphServiceClient()`, essencial para interagir com a API do Microsoft Graph.

Primeiro, utiliza a função `GetVaultKey()` (anteriormente explicada) para recuperar as credenciais necessárias armazenados no *Azure Key Vault*, que incluem o segredo e o ID do cliente. Estas são necessárias para criar as credenciais do cliente através da classe `ClientCredential()`. De seguida, o token de acesso é obtido através da autenticação no *Azure Entra ID*, utilizando a classe `AuthenticationResult()`. Finalmente, a classe `GraphServiceClient()` é configurada de modo a adicionar o token de acesso, num cabeçalho de autorização, em cada solicitação à API do Microsoft Graph.

A função **GetStructValue()** é utilizada para extrair e retornar o valor de um campo específico, a partir de uma coleção de dados. É uma função de apoio, que facilita o acesso e

manipulação direta da estruturas de dados complexas. O código desta função já se encontrava totalmente realizada.

As informações detalhadas sobre os utilizadores são obtidas a partir do Microsoft Graph na função **GetUserGraph()**. Esta função encontra-se parcialmente desenvolvida, partes como o acesso à API do Microsoft Graph já se encontravam realizado porém o código teve de ser adaptado ao caso do projeto atual, como a recuperação das informações pessoais dos utilizadores e o modelo do cartão digital escolhido.

A função começa por criar o modelo de dados, uma instância da classe `CardData()`, para armazenar as informações que serão recuperadas do SharePoint. Esta chama a função `SetGraphServiceClient()` anteriormente descrita, de modo a realizar solicitações à API do Microsoft Graph. Para realizar a chamada à API, é necessário configurar as opções de consulta, como expand e filter, que são passadas como parâmetros na lista `queryOptionsTemplate`. A primeira opção especifica que a API deve expandir os campos para incluir aqueles que estão aninhados. A segunda opção define o filtro dos resultados com base no identificador único do cartão do utilizador e no seu estado "Active".

Com as opções definidas, executa-se a chamada à API utilizando o método `GetAsync()`, de modo a obter os dados da lista "Cards" (criada por colegas) no SharePoint. Desta lista são recuperados os dados pessoais do utilizador como o nome, cargo, departamento, entre todos os outros necessários ao preenchimento devido do cartão digital. Estes são, posteriormente, armazenados no modelo de dados.

O código simplificado da configuração das opções e chamada à API do Microsoft Graph pode ser verificado na Listing 4.2.

Listing 4.2: Código simplificado da configuração das opções e chamada à API do Microsoft Graph

```
GraphServiceClient graphClientTemplate = SetGraphServiceClient();
var queryOptionsTemplate = new List<QueryOption>()
{
    new QueryOption("expand", "fields"),
    new QueryOption("filter", "fields/CardGUID eq '" + strGuid + "' and
        fields/Status/Value eq 'Active'")
};



var listTemplate = System.Threading.Tasks.Task.Run(() =>
    graphClientTemplate.Sites[strSiteGUID].Lists["Cards"].Items.Request(
queryOptionsTemplate).GetAsync()).GetAwaiter().GetResult();
```

```
if (listTemplate.CurrentPage.Count > 0)
{
    MyCardData.TemplateName = GetStructValue(listTemplate, "TemplateName");
    ...
}
...
```

Da mesma maneira que as informações do utilizador foram recuperadas, realiza-se uma nova chamada à API para recuperar o modelo do cartão em HTML. Esta chamada é configurada com novas opções de consulta, que filtram os dados da lista "Card Templates" no SharePoint, de modo a obter o corpo HTML do *template* específico associado ao utilizador. O HTML recuperado é armazenado no modelo de dados, completando as informações necessárias para a construção do cartão digital do utilizador na vista.

Por fim, de modo ao **controlador interagir com a vista** tem-se o método `Contact()`. Este recebe um parâmetro *id* com o identificador do cartão que se quer mostrar. Esse parâmetro faz parte do URL da aplicação e é definido no ficheiro `RouteConfig`. O URL é composto pelo domínio da aplicação, seguido da rota que direciona para o método `Contact()` no controlador "Home" (localizado no arquivo `HomeController.cs`), com o parâmetro *id* sendo adicionado no final.

O *id*, que representará o identificador do cartão a mostrar, é passado à função explicada anteriormente, `GetUserGraph(id)`, de modo a afetar o modelo de dados com as informações do mesmo. O HTML que constitui o cartão é extraído do modelo de dados e armazenado na propriedade `ViewBag`. Esta é uma propriedade da framework ASP.NET MVC que permite transferir dados de forma simples entre os controladores e as vistas.

Durante o processo, se ocorrer algum tipo de erro ou excepção nas funções acima descritas, são capturadas e registadas numa variável "LocalError". Esta armazena as mensagens de erro, permitindo que qualquer problema encontrado seja mostrado na view, facilitando a identificação e resolução de falhas na execução do código.

Modelo de Dados

Continuando o desenvolvimento de um projeto que visa um padrão MVC, é essencial a presença de um **modelo de dados**, este foi totalmente desenvolvido durante o estágio. Como o projeto visa apenas apresentar os cartões digitais num página Web básica, sem a perspetiva de escalar o projeto mais tarde, decidiu-se que modelo de dados fosse uma classe pertencente ao ficheiro *HomeController.cs*, tornando o seu acesso mais ágil a partir do mesmo ficheiro que contém o controlador e a fácil compreensão de todos na empresa.

Esta classe é projetada para encapsular todas as informações relevantes associadas ao cartão digital do utilizador e é fundamental para garantir que todas as informações necessárias sejam armazenadas e manipuladas de maneira eficiente e organizada dentro da aplicação.

O modelo de dados desenvolveu-se tendo em conta as informações pessoais dos utilizadores como: nome (*EmployeeName*), nome preferencial (*EmployeePreferredName*), o cargo (*EmployeeRole*), o departamento (*EmployeeDepartment*) e outros relacionados com os números de telefone, e-mail e moradas (*EmployeePhone1*, *EmployeePhone2*, *EmployeeEmail*, *EmployeeAddress1*, *EmployeeAddress2*, *EmployeeAddress3*, *EmployeeAddress4*, *EmployeeAddress5*).

Para atender às diferentes necessidades linguísticas, o modelo inclui campos adicionais com sufixo _ar, que armazenam as informações do utilizador em árabe (e.g *EmployeeName_ar*, *EmployeePreferredName_ar*, *EmployeeRole_ar*, etc). Isso permite a personalização do cartão digital para diferentes idiomas e regiões.

BodyHTML e *TemplateName* são campos que armazenam o conteúdo do cartão digital e o nome do template utilizado. O campo *BodyHTML* é particularmente importante, pois contém o HTML que define a aparência dos cartões.

Vista

As vistas são responsáveis por renderizar as página Web com as informações fornecidas pelo controlador.

Normalmente, uma vista que utiliza a framework ASP.NET MVC é uma página HTML que pode conter código Razor, uma sintaxe que permite a inclusão de lógica e dados diretamente no HTML. A vista necessária à disponibilização dos cartões digitais tem o nome de "Contact.cshtml". No cabeçalho desse documento HTML é onde são configurados os meta dados e o título da página.

Através da expressão Razor, `ViewBag`, recuperou-se do controlador as informações necessárias à renderização da página. O conteúdo HTML do cartão digital é inserido na página através da variável *BodyHTML* da propriedade `ViewBag`. De modo a renderizar o conteúdo da página, preservando todas as *tags* e formatações, adaptou-se o código HTML com a expressão Razor `@HTML.Raw`.

Se os cartões não conseguirem ser renderizados, a variável `ErrorMessage` dentro da propriedade `ViewBag` fica ativa. Através desta, o utilizador é informado do sucedido através de uma mensagem de erro.

Deste modo, realizou-se com sucesso o *front-end* e o *back-end* de uma plataforma Web, que disponibiliza os cartões digitais sem a necessidade de descarregar a plataforma móvel,

utilizando a tecnologia .NET e o padrão MVC.

Informações dinâmicas dos cartões digitais

Com a construção de todos os *templates* dos cartões de negócio digitais em HTML, do seu armazenamento em Sharepoint e a apresentação dos mesmos, tanto na aplicação Web como na aplicação móvel com base Power Apps, **é necessário preencher os cartões com as informações do utilizado**r. O preenchimento tem de ser realizado em ambas as plataformas, utilizando a linguagem de programação de cada uma.

Quando se abre a aplicação móvel, a própria plataforma Power Apps pede ao utilizador que faça a autenticação, utilizando credenciais do Microsoft 365. A partir desse momento, a aplicação conhece as credenciais do utilizador, e com elas, consegue recuperar todos os seus cartões. A recuperação dos dados necessários ao preenchimento de cada cartão é pelo seu identificador único. Na aplicação Web, esse identificador único é incluído, como parâmetro, no URL do pedido HTTP da página.

Para preencher os cartões com as informações do utilizador, **adaptou-se os campos que as detêm com *tags*, para serem substituídas de acordo com o seu proprietário**. Cada *tag* tem um nome alusivo à informação que a refere, com o símbolo "@" como prefixo. Por exemplo, o nome do detentor do cartão tomará o lugar da *tag @EmployeeName*, o seu cargo o da *tag @EmployeeRole*, o seu departamento o da *tag @EmployeeDepartment* e assim sucessivamente, para todas as informações dinâmicas das versões em inglês dos templates. Para as versões em árabe, utilizou-se o mesmo método, porém o prefixo invés de possuir apenas o símbolo "@", possuí "@Ar", um indicativo de que a informação a ser substituída deve estar em árabe. As *tags* das informações pessoais na versão inglesa, com disposição lado a lado, podem ser observadas na Figura 4.3.

Figura 4.3: Exemplo da inclusão de *tags* nas informações pessoais lado a lado, nos modelos de cartão de negócio digital, versão inglesa

Devido à simplicidade e limitações da linguagem *low-code* PowerFX, foi necessário adaptar a abordagem normal ao estruturar o algoritmo de substituição. Como solução, decidiu-se iniciar o desenvolvimento do algoritmo de substituição na aplicação Web, utilizando a linguagem C#, uma tecnologia mais familiar apesar da ausência de experiência prévia.

Aplicação Web

Na aplicação Web começou-se por recuperar o *template* em HTML, em inglês e em árabe, armazenados no modelo do projeto, colocando-os, cada um, numa variável do tipo string "bodyHTML".

Para realizar a substituição das *tags*, decidiu-se **dividir as diferentes informações em relação ao tipo de substituição** necessárias. Em todas as substituições foi utilizada o método `Replace()`, que permite substituir todas as ocorrências de uma `substring` dentro de uma `string`, por um dado do mesmo tipo.

A *tag* do **nome** tem de ser substituída pelo nome preferencial, porém se este não existir terá de ser substituída pelo nome completo. Para isto foi utilizado um estrutura de dados que armazena dois *arrays*, um para as informações e *tags* em inglês e o outro em árabe. Cada *array* possuí o nome preferencial do empregado, o nome completo e por fim a *tag* a ser substituída. A estrutura de dados é percorrida e verificado se, em cada *array*, o nome preferencial existe, se existir a *tag* é substituída por este, senão é substituída pelo nome completo. Se nenhum destes existir a *tag* é substítuida por um espaço em branco ` `.

De seguida realizou-se a mesma lógica para o **cargo**, mas sem a condição de preferência de informação. Desta vez utilizou-se uma coleção de pares chave-valor. Inseriu-se dois pares, um para cada linguagem, inglês e árabe. A chave contém a informação presente no modelo de dados enquanto o valor contém a *tag* a ser substituída. A estrutura de dados é percorrida e cada *tag* é substituída pela informação correspondente. Se o modelo de dados não contiver a informação, a *tag* é substituída por um espaço em branco.

No caso de **departamento**, um dos oito *templates* tem uma condição especial, não se encontra por baixo do cargo mas sim a seguir a este, separado por hífen. A lógica de substituição foi semelhante à anterior, com uma coleção de pares chave-valor, para todos os *templates* à excepção do que é diferente. Nesse caso, o símbolo do hífen tem de ser adicionado à informação como prefixo, antes de se substituir a *tag*.

As três informações referidas anteriormente devem sempre existir. Qualquer condição da sua omissão realizou-se de modo a prevenir eventuais erros.

De seguida agrupou-se as informações de **número de telemóvel, número de telefone e morada** num só ciclo de substituições, devido às suas caraterísticas semelhantes. Primeiro

aparecem nos *templates* com um prefixo. O número de telemóvel com o prefixo “M”, o número de telefone com o prefixo “T” e por fim a morada com o prefixo “E”. Segundo a disposição destas informações no cartão é de um alinhamento vertical em sequência, como se de uma lista se tratasse. Como qualquer uma destas informações pode estar omissa, o item a seguir a essa omissão tem de ocupar o seu lugar, resultando num deslocamento vertical para cima de todas as informações desta lista.

Dito isto, conclui-se que os prefixo não se podem introduzir no próprio HTML, têm de ser inseridos dinamicamente aquando da substituição das *tags*, para que sejam apresentados ou removidos consoante a omissão da sua informação correspondente.

Para solucionar esta substituição, criou-se uma coleção de pares chave-valor para armazenar cada informação e o prefixo correspondente. Percorre-se essa estrutura e verifica-se quais as informações que realmente estão populadas, as que verificarem esta condição são adicionadas a outra estrutura idêntica, mas apenas com as informações populadas, com o respetivo prefixo. De seguida, colocaram-se todas as *tags* num *array*. Percorreu-se o *array* de informações populadas e atribui-se a cada uma das *tags* existentes por ordem, juntamente com o respetivo prefixo e o seu devido estilo. No final substituiu-se as *tags* que faltam por espaços me branco. Deste modo, são substituídas o mesmo número de *tags* que informações disponíveis, e todas as informações são agrupadas no início da lista. Este algoritmo consta no Listing 4.3.

Listing 4.3: Algoritmo para substituição de *tags* para informações pessoais com prefixo

```
// Create a dictionary to store phone and email information in English.
Dictionary<string, string> phoneAndEmailInfoEng = new Dictionary<string, string>
{
    { "T", user.EmployeePhone2 }, // Maps the key "T" to the employee's second phone
        number.
    { "M", user.EmployeePhone1 }, // Maps the key "M" to the employee's first phone
        number.
    { "E", EmployeeEmailText } // Maps the key "E" to the employee's email address.
};


// Create a new dictionary to store non-empty phone and email information.
Dictionary<string, string> populatedPhoneAndEmaiInfoEng = new Dictionary<string,
    string>();
```

```
// Iterate over each entry in the phoneAndEmailInfoEng dictionary.
foreach (var entry in phoneAndEmailInfoEng)
{
    // Check if the value associated with the key is not null or just whitespace.
    if (!string.IsNullOrWhiteSpace(entry.Value))
    {
        // Add the entry to the dictionary if the value is not empty.
        populatedPhoneAndEmaiInfoEng.Add(entry.Key, entry.Value);
    }
}

// Define an array with tags to be replaced in the HTML.
string[] infoTagsEng = { "@EmployeePhone1", "@EmployeePhone2", "@EmployeeEmail" };
int countEng = 0;
// Iterate over each entry in the populatedPhoneAndEmaiInfoEng dictionary.
foreach (var entry in populatedPhoneAndEmaiInfoEng)
{
    // Replace the tags in the HTML with the corresponding value from the dictionary.
    // Append "</b>" after the key and concatenate the value.
    BodyHTML = BodyHTML.Replace(infoTagsEng[countEng], entry.Key + "</b>" +
        entry.Value);
    countEng++;
}

// Replace any remaining tags in the HTML with a non-breaking space
for (int i = populatedPhoneAndEmaiInfoEng.Count; i < infoTagsEng.Length; i++)
{
    // Replace tags that were not found with " ".
    BodyHTML = BodyHTML.Replace(infoTagsEng[i], "</b> ");
}
```

Para a versão em árabe o processo foi o mesmo, mudando apenas as informações do cartão, os prefixos e as *tags*.

Por fim, as **moradas** tem a mesma caraterística que as informações pessoais acima descritas, a sua disposição no cartão é de um alinhamento vertical em sequência, como se de uma lista se tratasse. Estas foram substituídas da mesma forma, apenas não se introduziu a lógica dos prefixos.

Um outro template tinha uma estrutura diferente para as moradas. Estas tinham uma

disposição no cartão de alinhamento horizontal em sequência, separadas por vírgulas. Para esta substituição, armazena-se num *array* todas as informações das moradas de modo a verificar quais as que estão populadas. À imagem do que se desenvolveu anteriormente, as informações populadas são adicionadas a uma nova estrutura de dados. De seguida percorre-se essa última estrutura e substitui-se todas as *tags* por ordem, se a informação não for a primeira a ser colocada, adiciona-se uma vírgula à informação, como se de um prefixo de tratasse. Para a versão em árabe o processo foi o mesmo, porém como se escreve da direita para a esquerda, a vírgula de separação entre moradas é associada como um sufixo.

Posteriormente, após realizar-se vários testes, notou-se que como o **e-mail** não possui espaços, ao contrário das outras informações, não quebrava naturalmente quando ultrapassa o seu contentor HTML. Primeiramente, idealizou-se realizar este dinamismo em Javascript, porém como tem de haver consistência entre a aplicação móvel e a aplicação Web, e como o controlador que apresenta HTML na aplicação Power Apps não suporta Javascript, não é possível fazer este dinamismo através desta linguagem. Resolveu-se realizar uma lógica de quebra, segundo um comprimento específico, ou seja, o e-mail deve quebrar sempre que chegar a um número máximo de caráteres, pode ser em duas, três ou mais linhas. Essa lógica foi convertida para um algortimo em C#, este está disponível na Listing 4.4.

Listing 4.4: Algoritmo para quebra de email consonate um número máximo de carateres, na aplicação Web

```
// Split the email into segments of intNumberCharBeforeEmailBreak
List<string> segments = new List<string>();



for (int i = 0; i < Math.Ceiling((double)user.EmployeeEmail.Length /
    intNumberCharBeforeEmailBreak); i++)
\{
    int startIndex = i * intNumberCharBeforeEmailBreak;
    int length = Math.Min(intNumberCharBeforeEmailBreak, user.EmployeeEmail.Length -
        startIndex);
    string segment = user.EmployeeEmail.Substring(startIndex, length);
    segments.Add(segment);
}
```

```
// Join segments with line breaks
StringBuilder formattedEmail = new StringBuilder();
foreach (string segment in segments)
{
    formattedEmail.Append(segment);
    if (segment != segments.Last()) // Add <br> except for the last segment
        formattedEmail.Append("<br>");
}
// Output the formatted email
string EmployeeEmailText = formattedEmail.ToString();
```

Primeiro criou-se uma estrutura de dados para armazenar os diferentes segmentos (linhas) do e-mail quebrado. De seguida percorreu-se um ciclo, tantas vezes quanto o número de linhas que o e-mail terá, através da divisão entre o seu comprimento e número máximo de carateres por linha. Dentro do ciclo, calcula-se o índice de começo e o comprimento de cada segmento, de modo a quebrar a string do e-mail sem nunca ultrapassar o limite máximo de carateres e adiciona-se à estrutra de dados de segmentos. No final, unem-se todos os segmentos utilizando a classe `StringBuilder()`, adicionando a cada segmento uma quebra que linha. Deste modo o e-mail quebra sempre no número máximo de carateres e continua na linha seguinte. Adicionou-se, por fim, o número máximo de caráteres até o e-mail ser cortado, no ficheiro `Web.config`, assim este número fica acessível e pode ser configurado pelo cliente.

Aplicação móvel

Na aplicação móvel, a lógica de substituição das *tags* realiza-se no evento "*OnVisible*", do ecrã que apresenta ao utilizador o seu cartão digitial.

Conforme mencionado anteriormente, o raciocínio em relação ao desenvolvimento de código está moldado pelas linguagens de programação tradicionais, o que inicialmente resultou numa certa dificuldade em conceber um algoritmo de substituição utilizando a linguagem PowerFX. Esta linguagem, tem muitas semelhantes com as funções utilizadas por exemplo no Microsoft Excel.

Neste caso a lógica que queremos implementar é a mesma que a lógica implementada na aplicação Web. Existem diferentes *tags* num cartão digital em HTML e pretende-se substituir as *tags* pelas informações reais dos utilizadores. Porém, com esta tecnologia, é impossível atualizar variáveis dentro de ciclos, o que torna também impossível recriar o mesmo algoritmo implementado na aplicação Web.

Utiliza-se a função `UpdateContext()` para criar a variável local *EnglishHTML*, onde será armazenado o código HTML da versão inglesa do cartão. Esta variável será afetada com três

funções Substitute(), uma a seguir à outra, de forma a substituir de forma seguida o **nome, cargo e departamento**. Esta lógica pode ser visualizada na Listing 4.5.

No primeiro argumento da função `Substitute()`, esta espera a `string` à qual se pretende fazer a substituição. Assim invoca-se a função `LookUp()`, de modo a recuperar o *template* a substituir, da variável global que contém todos os *templates* dos cartões. No segundo argumento da função `Substitute()`, esta espera a `substring` que se quer substituir. Aqui coloca-se a respetiva *tag*, por exemplo*@EmployeeName*. No terceiro e último parâmetro da função `Substitute()`, é necessário fornecer a `string` que substituirá a `string` original. No caso do nome, invoca-se a função `If()`, pois há a necessidade de se verificar se a informação do nome preferível está populada.

Utilizou-se o que retorna desta função `Substitute()`, como a string "mãe" para realizar a substituição seguinte. Assim colocou-se toda a expressão no primeiro argumento de uma segunda função `Substitute()`, desta vez para encontrar a *tag @EmployeeRole* e substituí-la pela informação correta.

De igual forma, o que retorna da segunda função `Substitute()`, utilizou-se numa terceira para realizar, desta vez, a substituição da *tag @EmployeeDepartment*. Esta última tem ainda de fazer a verificação de qual o *template* presente, já que num dos *templates* é necessário a adição de um prefixo com o símbolo hífen.

De forma análoga ao explicado anteriormente, efetuou-se a substituição do **nome, cargo e departamento, desta vez para a versão árabe** dos *templates*, com a variável *ArabicHTML*.

Listing 4.5: Algoritmo de substituição de tags na aplicação móvel

```
UpdateContext( //ENGLISH card version
    {
        EnglishHTML: Substitute( //Department substitution
            Substitute( //Role substitution
                Substitute( //Name substitution
                    LookUp(
                        dataSharePointListCardTemplate, ID = gvarCard.'Template
                            Id'.Id).EnglishHTML,
                    "@EmployeeName",
                    If(IsBlank(gvarCard.EmployeePreferredName), //Check if blank
                        If(IsBlank(gvarCard.EmployeeName), " ",
                            gvarCard.EmployeeName),
                    gvarCard.EmployeePreferredName)
                ),
```

```
            "@EmployeeRole",
            If(IsBlank(gvarCard.EmployeeRole), //Check if blank
                " ",
                gvarCard.EmployeeRole
            )
        ),
        "@EmployeeDepartment",
        If(
            IsBlank(gvarCard.EmployeeDepartment), //Check if blank
             " ",
            If(LookUp(dataSharePointListCardTemplate,ID = gvarCard.'Template
                Id'.Id).Title = "Securities", //Template exception
                Concatenate(" - ", gvarCard.EmployeeDepartment),
                gvarCard.EmployeeDepartment
            )
        )
    )
}
```

De seguida, desenvolveu-se a lógica de deslocamento vertical das informações pessoais populadas, **número de telemóvel, número de telemóvel e morada**. Para isso, utiliza-se a função `UpdateContext()` três vezes, para inicializar variáveis contendo um valor de verdade, se cada uma das respetivas informações se encontra populada no Sharepoint. O código utilizado para esta primeira parte do algoritmo de deslocamento vertical está disponível na Listing 4.6.

Listing 4.6: Variáveis, em PowerFx, com valor de verdade se a informação estiver populada na base de dados

```
UpdateContext({varEmployeeEmail: !IsBlank(gvarCard.EmployeeEmail)}); //True if
    EmployeeEmail exists
UpdateContext({varEmployeePhone: !IsBlank(gvarCard.EmployeePhone)}); //True if
    EmployeePhone exists
UpdateContext({varEmployeePhone2: !IsBlank(gvarCard.EmployeePhone2)});//True if
    EmployeePhone2 exists
```

Por forma a guardar as substituição destas informações, utiliza-se novamente a função `UpdateContext` porém desta vez afetando a variável já inicializada, *EnglishHTMl* e posteriormente, *ArabicHTML*. Afeta-se primeiro esta variável com o resultado da substituição da

primeira *tag* (*@EmployeePhone1*). A string que a irá substituir é escolhida dentro de uma função `Switch()`, esta permite no seu primeiro argumento o valor esperado (neste caso será ser verdade que a informação está populada), e nos pares de argumentos seguintes, o valor a verificar e o valor a retornar. No valor a verificar coloca-se as variáveis com o valor de verdade se cada informação está populada e no valor a retornar a própria informação. Assim a função `Switch()` irá escolher a primeira informação que estiver populada. Após essa informação ser escolhida, muda-se a variável com seu valor de verdade para falso.

Sempre que uma destas informação é escolhida na função `Switch()`, é adicionado o respetivo prefixo com a função `Concatenate()`. Esta permite a junção de duas ou mais *strings* numa só.

Para a segunda e terceira *tag*, procede-se à mesma lógica. Porém a informação escolhida para a substituição da *tag* anterior nunca pode ser escolhida nas seguinte, já que o valor da variável que anuncia a sua população torna-se falsa, sempre que essa informação é substituída. As *tags* que não chegam a ser substituídas, ficam com o uma espaço em branco. A substituição da primeira *tag* (*@EmployeePhone1*) em PowerFx para o deslocamento vertical, explicado anteriormente, consta na Listing 4.7.

Listing 4.7: Substituição das tags consoante valor de verdade

```
// Substitutes the first tag (@EmployeePhone1) with the first available information
// The tag is replaced with either EmployeePhone2, EmployeePhone, EmployeeEmail, or
    an empty string if none are available.
UpdateContext({
    EnglishHTML: Substitute(
        EnglishHTML,
        "@EmployeePhone1", // Tag to be replaced
        Switch(
            true, // Condition to evaluate
            varEmployeePhone2, Concatenate("T</b>", gvarCard.EmployeePhone2), //
                Replaces with EmployeePhone2 if available
            varEmployeePhone, UpdateContext({varEmployeePhone: Blank()});
                Concatenate("M</b>", gvarCard.EmployeePhone), // If not, replace with
                EmployeePhone and clear the phone variable
            varEmployeeEmail, UpdateContext({varEmployeeEmail: Blank()});
                Concatenate("E</b>", varEmployeeEmailText), // If not, replace with
                EmployeeEmail and clear the email variable
            true, "</b> " // If none of the conditions match, replace with a
                blank space))});
```

As **moradas**, tanto nas versões dos *templates* em inglês como em árabe, são substituídas de igual forma às informações pessoais, à excepção da concatenação do prefixo.

Também na aplicação móvel tem de ser realizado o algoritmo de **quebra de e-mail** segundo um número máximo de letras. Porém este foi desenvolvido de uma maneira mais eficiente, já que foi conversado com colegas mais experientes sobre as limitações da tecnologia, aquando da realização da substituição das informações em PowerFX. Assim, aprendeu-se que a alternativa a afetar variáveis dentro de ciclos, é afetar coleções.

Com isto em mente, criou-se uma coleção vazia para armazenar as partes segmentadas do e-mail.

De seguida, tal como foi realizado na aplicação Web, percorre-se um ciclo for tantas vezes quantos segmentos que vão ser analisadas. Para isso utilizou-se a função `Sequence()`, nesta gera-se uma sequência de números baseada no comprimento do e-mail dividido pelo comprimento máximo até à quebra de linha.

Com a função `Collect()`, afeta-se a coleção já inicializada com cada segmento de texto resultante. Cada segmento é obtido usando a função *Mid()*, que extrai uma *substring* do e-mail a partir da posição inicial calculada, e com um comprimento igual ao número máximo de caráteres.

Utilizou-se a função `Concat()` para juntar todos os segmentos presentes na coleção, adicionando um carater de mudança de linha (*
*) após cada um, de forma a quebrar o e-mail no HTML. Com a função `Left()` retirou-se os últimos quatro caráteres da *string* concatenada, já que a última linha do e-mail não necessita de mudança de linha. Por último, cria-se uma variável para armazenar o e-mail com as quebras corretas.

O código utilizado para o algoritmo de quebra de e-mail consoante um número máximo de caráteres, em PowerFx, pode ser verificado na Lisiting 4.8.

Listing 4.8: Algoritmo de quebra de email consoante um número máximo de carateres, em PowerFx

```
//ENGLISH VERSION
// Create a collection to hold segments of the email
ClearCollect(colEmailSegments, []);
// Loop through the email in chunks of varMaxLineLength and collect segments
ForAll(Sequence(RoundUp(Len(gvarCard.EmployeeEmail) / varMaxLineLength, 0)),
    Collect(colEmailSegments,
        {
            Segment: Mid(
                gvarCard.EmployeeEmail,
                (Value - 1) * varMaxLineLength + 1,
```

```
                varMaxLineLength
            )
        }));
// Remove last <br>
UpdateContext(varEmployeeEmailText,
    Left(Concat(colEmailSegments,Segment & "<br>"),
        Len(Concat(colEmailSegments,Segment & "<br>")) - 4)
);
```

Ao utilizar coleções para atualizar informações dentro de ciclos, o algoritmo para quebrar e-mails tornou-se muito mais conciso em comparação com o método anteriormente utilizado para a substituição de *tags*. Com base nisto, considerou-se em reformular este último, pois mesmo que o código executa-se rapidamente ao entrar no ecrã, este continha um número de linhas de código considerável. Devido a questões de eficiência de tempo, esta reforma não chegou a ser realizada.

Assim conclui-se com êxito a inserção dinâmica de informações nos cartões digitais, tanto na aplicação móvel utilizando a linguagem PowerFX, como na aplicação Web utilizando a linguagem C# da plataforma .NET.

***Download* dos vcards**

Outro dos requisitos da solução, era o utilizador conseguir **descarregar e salvar os dados do contato** diretamente no seu dispositivo. Para isso recorreu-se ao conceito de **vCard**.

Um vCard permite enviar informações de contactos num formato compatível com diferentes programas e serviços, incluindo Microsoft Outlook, Apple Contacts, Google Contacts, entre muitos outros. Este é guardado como um ficheiro *.vcf*, que é o padrão da Internet para partilhar informações de contactos. Quando descarregado e importado, as informações do contato presentes no vCard são adicionadas à lista de contatos.

O vCard recorre a uma string formatada com os definições e parâmetros necessários para serem lidos pelas aplicações [30], [31]. No controlador do projeto Web, através da API do Microsfot Graph e do id do cartão, passado no URL da aplicação, recuperam-se todas as informações do utilizador. De seguida, construiram-se duas `strings`, uma para a versão do Vcard em inglês e outra em árabe, com alguns dos parâmetros necessários associando os respetivos dados do utilizador a cada um.

Com a `string` do Vcard construída e formatada, é necessário a conversão desta num *array* de bytes. Utilizou-se a codificação UTF-8 recorrendo ao método `Encoding.UTF8.GetBytes` da

biblioteca de classes da framework .NET.

Por fim, através da `string` convertida em bytes, criou-se um ficheiro com o formato "text/v-card", recorrendo-se à classe `FileContentResult`, presente na biblioteca de classes da framework ASP.NET. Através desta é também possível definir o nome do ficheiro criado.

Criaram-se duas funções com a lógica apresentada acima, uma para cada versão da informação, inglês e árabe, de modo a serem chamadas através de dois botões distintos criados na vista. Estes botões podem ser visualizados na Figura 4.4. Quando o utilizador clica em algum dos botões, é redirecionado para o URL com o caminho para a respetiva função, o que inicia o *download* do arquivo vCard. A função que cria e descarrega a versão inglesa do vCard consta na Listing 4.9.

Figura 4.4: Botões incluídos na vista para descarregar a informação dos cartões digitais em forma de vCard

Listing 4.9: Função que cria e inicia o *download* do vCard, com as informações em inglês

```
public ActionResult GetVCARDENG(string id)
{
    try
    {
        // Get the vCard data for the user in English using the 'id' passed in the URL.
        string strVcard = GetCard("ENG", id);
        // Convert the vCard string to a byte array using ASCII encoding.
        Byte[] bytes = Encoding.ASCII.GetBytes(strVcard);
        // Create a response as a file result with the vCard content.
        var response = new FileContentResult(bytes, "text/vcard")
        {
            // Set the default filename for the downloaded vCard.
            FileDownloadName = "EngVCard.vcf"
        };
        // Return the file response.
        return response;
    }
    catch (Exception ex)
    {
```

```
        // If an error occurs, return an HTTP 405 error with the exception message.
        return new HttpStatusCodeResult(405, "Erro na criacao do Ficheiro:" +
            ex.Message);
    }
}
```

Assim foi concluído com sucesso a caraterística da solução que visa descarregar e salvar os dados do contato diretamente no dispositivo do utilizador.

Geração de Códigos QR:

No último sprint do projeto é necessário a geração de um código QR na aplicação móvel, que redirecionará o utilizador para a página Web com o seu cartão digital.

Como esta funcionalidade não é oferecida nativamente pela Microsfot Power Platform, decidiu-se utilizar uma função Azure. Esta, através de um pedido HTTP, terá de gerar um código QR com o *link* para a página Web do cartão digital do utilizador.

Dito isto, começou-se por investigar alguns blogs com tutoriais de funções Azure que gerassem códigos QR. Após alguma pesquisa decidiu-se utilizar um tutorial que se assemelhava ao pedido. Utilizou-se o tutorial "Building a QR code generator with azure functions", desenvolvido por Jeremy Morgan, disponível no seu blog de tecnologia "All hands on tech", [32].

Para desenvolver e testar a função Azure localmente foi instalado o Azure Functions Core Tools.

Começou-se por criar um projeto com as dependências necessárias à criação de funções Azure, através do seguinte comando na linha de comando da diretoria que queremos que fica alocado: *func init QRCodeGen –worker -runtime dotnet*. Este projeto será chamado "QRCodeGen" e utilizará a plataforma .NET.

Após o projeto ter sido criado, gerou-se a função Azure com o comando *func new –template "Http Trigger--name QRCodeGen –authlevel anonymous*. Com este comando, define-se o *trigger* da função como um pedido HTTP e o seu nome como "QRCodeGen". Já que esta função não lida com informações críticas decidiu-se o nível de autorização como "anonymous", o que quer dizer que se pode aceder à função sem necessidade de fornecer uma chave de autorização. Todavia, poderia-se colocar o acesso à função restrito com uma chave específica, se houvesse necessidade.

A seguir, utilizou-se a plataforma .NET para instalar uma biblioteca que permite transformar um texto ou dados binários numa imagem PNG contendo um código QR. Esta biblioteca tem como nome "QRCode Generator" e foi desenvolvida por Manuel BL [33]. Procedeu-se à

instalação desta através da framework .NET.

Esta biblioteca apenas gera os códigos QR no formato SVG. Já que queremos o resultado em PNG, adicionou-se uma extensão da biblioteca referida e instalou-se o pacote “SkiaS, harp”, que permite a geração destas imagens, através da plataforma .NET.

Através destas configurações, tem-se o projeto pronto para realizar a função Azure pedida.

Começou-se por dar um nome à função, neste caso “GenerateQRCode”. Este será o nome mencionado aquando do pedido HTTP. Dentro da função, foi criado um método chamado “Generate”, responsável por processar os parâmetros de configuração e lidar com o informação será recebida por HTTP.

A informação que irá ser transformada para o formato de código QR terá de vir do pedido HTTP, por isso recolhe-se o parâmetro presente no URL do pedido antes de ser transformado.

De seguida recorre-se à biblioteca de geração de códigos QR, anteriormente instalada, para transformar o parâmetro do URL num código QR, porém, como referido, esta biblioteca gera apenas imagens no formato SVG. Utilizou-se a extensão adicionada para converter o código QR para formato PNG, com alguns parâmetros é possível configurar o tamanho da imagem e da sua margem, a cor do fundo e a cor do código. Utilizou-se o tamanho 10 para o código o que gera um tamanho visível, uma borda miníma de 1 e as cores de fundo e do código branco e preto, como é tradicional.

Por fim, é necessário criar o objeto JSON que a função Azure irá retornar. Para isto utiliza-se uma segunda classe apenas com uma propriedade “Image” que irá conter o código QR. Este foi convertido para base64, de forma a conseguir ser renderizado mais facilmente pelos navegadores de Internet.

A função Azure desenvolvida está disponível na Listing 4.10.

Listing 4.10: Função Azure que gera um código QR, a partir de um parametro inserido no URL do seu pedido HTTP

```
public static class QRCodeGen
{
    [FunctionName("GenerateQRCode")]
    public static async Task<IActionResult> Generate(
        [HttpTrigger(AuthorizationLevel.Anonymous, "get", Route = null)] HttpRequest
            req, ILogger log)
    {
        // Retrieve the 'qrtext' query parameter from the HTTP request
        string qrtext = req.Query["qrtext"];
```

```
        // Encode the text into a QR code with medium error correction level
        var qr = QrCode.EncodeText(qrtext, QrCode.Ecc.Medium);
        // Convert the QR code into a PNG image
        var pngout = qr.ToPng(10, 1, SkiaSharp.SKColors.Black,
            SkiaSharp.SKColors.White);
        // Create an instance of ReturnObject to have the image data
        var ourResult = new ReturnObject { };
        // Convert the PNG image byte array to a Base64 string and assign it to the
            Image property
        ourResult.Image = Convert.ToBase64String(pngout);
        // Return the result as a JSON response
        return new JsonResult(ourResult);
    }
}

public class ReturnObject
{
    // Property to hold the Base64 encoded image string
    public string Image { get; set; }
}
```

Após o desenvolvimento e teste desta função localmente, a função foi colocada *online* através da plataforma Azure, por um colega sénior, inserido neste projeto.

Assim, quando se faz o pedido HTTP ao micro-serviço oferecido pela função Azure, inclui-se um parâmetro “qrtext”, onde se fornece o *link* para a página Web que contém os cartões digitais do utilizador. A função Azure, por sua vez, devolve um objeto JSON com o texto “Image” referenciando o código QR em base64. O resultado da chamada da função Azure através de um pedido HTTP aleatório, pode ser visualizado na Figura 4.5.

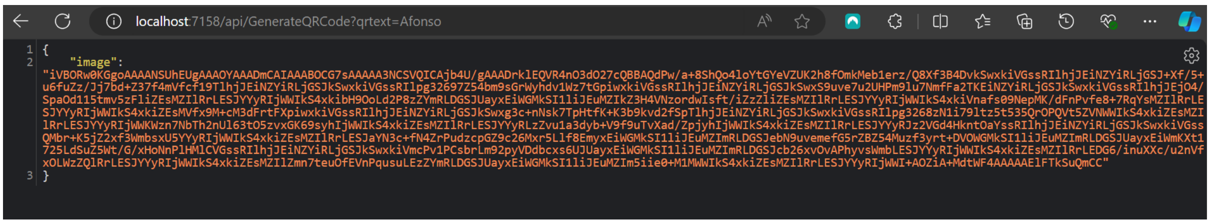


Figura 4.5: Resultado da chamada da função Azure através de um pedido HTTP

Esta funcionalidade foi mais tarde integrada, por colegas, na aplicação móvel utilizando a tecnologia Power Automate. Quando a aplicação é iniciada, uma automatização utilizando a tecnologia é despertada, de modo a realizar um pedido HTTP à função Azure, este pedido contém um parâmetro no URL contendo a página Web que apresenta os cartões digitais. Assim a função Azure devolve um objeto JSON com uma imagem do código QR codificada em base64. Este objeto é tratado pela automatização e é colocado numa página da aplicação móvel. Quando este código é capturado por qualquer smartphone, redireciona o utilizador para a página Web contendo os cartões digitais. A Figura 4.6 contém o código QR como é mostrado na aplicação móvel, em formato PNG.

Figura 4.6: Resultado da função Azure em imagem PNG, renderizada por um navegador de Internet

4.1.4 Avaliação e Resultados

Terminou-se com sucesso todos os sprints deste projeto dentro dos prazos definidos e com soluções de qualidade que atendem aos requisitos estipulados.

No primeiro sprint deste projeto, foi finalizada com êxito parte do primeiro requisito funcional atribuído ao âmbito do estágio: "O utilizador deve poder visualizar os cartões digitais em Inglês e em Árabe". Este sprint focou-se no desenvolvimento da aplicação móvel pelos colegas da empresa, enquanto o estágio objetivou a digitalização e recriação dos cartões de negócios digitais. Estes *templates* assumem a forma de cartões de negócios horizontal, como tradicionalmente a empresa utiliza para a divulgação dos seus serviços.

Durante o desenvolvimento dos cartões digitais, foi tido em atenção o requisito não funcional "O design e esquema de cores dos diferentes *templates* dos cartões devem estar alinhado com os fornecidos pelo cliente". Isto garantiu que os cartões digitais fossem o mais fiéis possível aos cartões físicos usados pelos colaboradores da empresa.

O principal desafio enfrentado no primeiro sprint foi superar as limitações do controlador de texto HTML da aplicação móvel, desenvolvida em Microsoft Power Apps. Apesar dessas dificuldades, todos os *templates* foram concluídos com sucesso dentro do prazo estipulado. Tanto os clientes quanto os colegas da empresa e os gestores de projeto expressaram grande satisfação com os resultados. Os resultados da recriação dos *templates* não podem ser de-

monstrados com imagens já que possuem direitos de autor.

O segundo sprint teve como objetivo conceber os requisitos funcionais “Os destinatários da partilha do cartão digital devem ser providenciados com um *link* para visualização do mesmo” e “O *link* que contém o cartão digital deve ser disponibilizado em hiperligação HTTP”. Foi desenvolvida tanto o *front-end* como o *back-end* de uma aplicação Web para apresentar e partilhar os cartões digitais tendo em conta o padrão MVC. Durante o desenvolvimento da aplicação considerou-se o requisito não funcional “A aplicação deve ser otimizada para telemóvel”, através da realização de uma aplicação responsiva e adaptada para diferentes smartphones. Todas as informações necessárias à apresentação dos cartões foram recuperadas do Sharepoint através do acesso seguro à API do Microsoft Graph, atendendo aos requisitos obrigatórios de segurança da empresa, utilizando o Azure Key Vault.

No terceiro sprint foram concebidos não só algoritmos de substituição dinâmica de *tags* consoante a informação disponível na base de dados, como também algoritmos de quebra de texto consoante um número máximo de caráteres. Estes algoritmos foram concebidos, com os mesmos resultados, para ambas as plataformas que apresentam os cartões: a aplicação móvel, com a lógica em Powerfx, e a aplicação Web, com a lógica em C#. Numa primeira instância a linguagem de programação PowerFx apresentou algumas limitações em relação à linguagem de desenvolvimento tradicional C#. Porém, após encontradas soluções alternativas às limitações encontradas, os algoritmos na aplicação Power Apps tornaram-se tão ou mais concisos e eficazes como aqueles utilizados na aplicação Web.

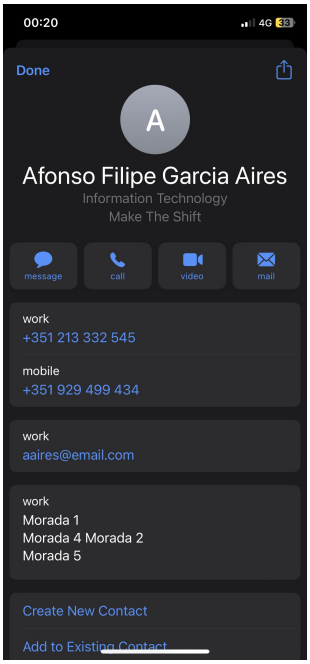
(a) Disposição da informação sem omissão de dados

(b) Disposição da informação com omissão do número de telemóvel, morada 2 e morada 4

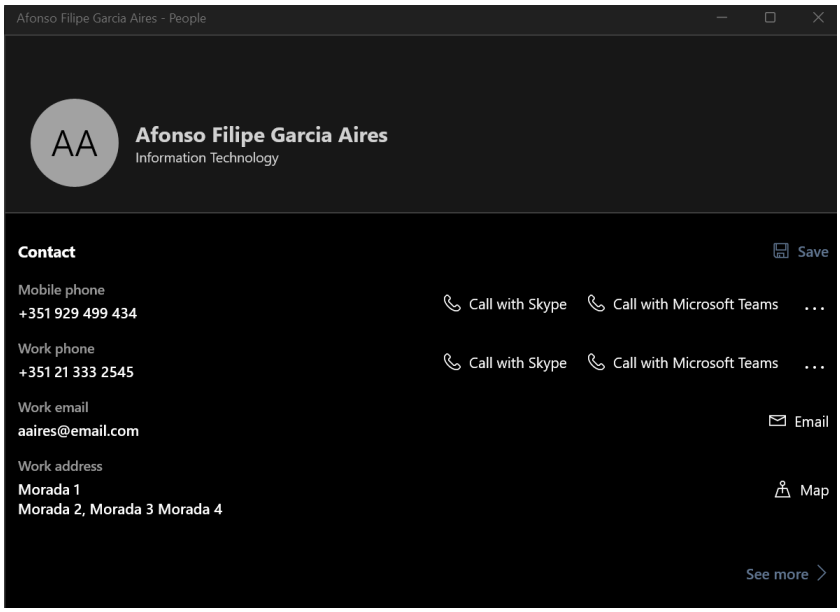
Figura 4.7: Resultado do deslocamento vertical da informação dinâmica do cartão de negócio digital

Na Figura 4.7 pode ser visto um exemplo do resultado da substituição dinâmica da informação pessoal presente no cartão de negócio digital. Na Figura 4.7 (a) consta a disposição da informação se nenhum dos dados for omisso. Na Figura 4.7 (a) consta a disposição da informação se o número de telemóvel, a morada 2 e 4 não estão presentes na base de dados, resultando num deslocamento vertical das informações disponíveis, que são agrupadas de forma contínua.

Com o quarto sprint deste projeto, conseguiu-se concluir um dos requisitos funcionais da solução: “Os destinatários da partilha do cartão digital devem ser providenciados com um *link* para visualização do mesmo, o qual deve ainda permitir o seu *download* e salvar os dados do contato diretamente no seu dispositivo”. Para este, foi utilizado o conceito de vCard, o padrão da Internet para guardar e partilhar informações de contacto. Através da conversão da informação do utilizador para este formato, utilizando a biblioteca de classes da framework utilizada .NET, é possível o descarregamento das informações do cartão digital, através da aplicação Web, de forma a serem salvos tanto no telemóvel (Figura 4.8 (a)) como no computador (Figura 4.8 (b)).



(a) Descarregamento do VCard, em smartphone



(b) Descarregamento do VCard, em computador

Figura 4.8: Resultado do descarregamento das informações do utilizador através do formato VCard

O quinto e último sprint objetivou a criação de um código QR através de uma função Azure. Esta função Azure foi definida para ser despertada através de um pedido HTTP e, com recurso a uma biblioteca externa, transforma qualquer texto num código QR, neste caso foi solicitado o URL da página Web que apresenta os cartões digitais de um utilizador. O código QR foi, posteriormente, colocado na aplicação móvel por um colega sénior. Este sprint concluiu assim a solução ao realizar a última parte do requisito funcional “O *link* que contém o cartão digital deve ser disponibilizado em hiperligação HTTP ou código QR”. A página, da aplicação móvel, que disponibiliza o código QR pode ser visualizada na Figura 4.9.

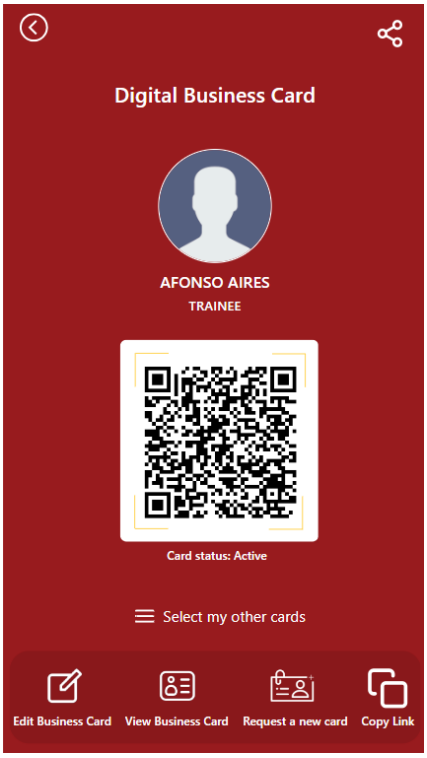


Figura 4.9: Resultado da página na aplicação móvel que disponibiliza o código QR

Após cada sprint as tarefas desenvolvidas foram testadas por um grupo de pessoas designadas de Equipa de Controlo. Esta equipa realizou testes de funcionalidades onde verificou se o software atende aos requisitos especificados e testes de regressão que foram executados para assegurar que alterações recentes não afetaram negativamente funcionalidades já existentes. Os testes são planeados com base nos requisitos angariados, e os casos de teste são documentados para garantir que todas as funcionalidades sejam verificadas. Durante a execução, se algum problema é identificado, é registado numa folha excel que todos os programadores têm acesso. Os problemas são caraterizados pela gravidade do erro (baixa, média, alta, crítica), a pessoa que deve proceder à sua resolução ou verificação (normalmente a resolução do problema é atribuída ao programador e a sua verificação ao líder do projeto) e o seu estado (pendente, verificar, corrigido). O único problema atribuído às tarefas descritas foi a impossibilidade de quebra do e-mail, que foi resolvida com a quebra dinâmica, considerando um número máximo de caráteres.

Com a conclusão do último sprint a solução passou para a fase de UAT, de modo a ser testada com auxílio do cliente, para a deteção e posterior resolução de problemas ainda não identificados.

Entregou-se assim, uma solução altamente escalável que facilita a partilha de cartões digitais de forma eficiente, mantendo a identidade visual dos cartões já utilizados pela empresa.

A solução atendeu às necessidades do cliente, garantindo uma integração perfeita com os sistemas existentes.

A implementação seguiu as boas práticas de desenvolvimento e segurança, o que contribuiu para a proteção confiável dos dados e assegurou uma plataforma sólida e eficiente.

4.2 Aplicação de Gestão de Cartões de Negócios Digitais - Cartões Verticais

Esta aplicação foi desenvolvida no âmbito da demonstração da aplicação anteriormente referida, “Aplicação de Gestão de Cartões de Negócios Digitais - Cartões Horizontais”, para um novo cliente.

Esta deve oferecer a mesma solução que a anterior, visando apresentar aos funcionários do cliente as funcionalidade de criar, gerir, personalizar e partilhar cartões de negócios digitais de forma eficaz, aproveitando a infraestrutura da Microsoft 365.

A única diferença desta aplicação para a anterior será o design do cartão digital, que neste projeto, deve ser um cartão digital vertical com um estilo moderno e sofisticado.

4.2.1 Requisitos Atribuídos

Como anteriormente referido, a aplicação a desenvolver é uma cópia de uma já realizada. Dito isto, os requisitos da aplicação que foram atribuídos ao contexto do estágio, são exatamente os mesmo que no projeto passado, referidos na sub-secção 4.1.1. Existe no entanto uma excepção nos requisitos não funcionais. O anterior requisito não funcional:

- O design e esquema de cores dos diferentes *templates* dos cartões devem estar alinhado com os fornecidos pelo cliente

Irá deixar de ter incluído o design, não é fornecido pelo cliente e está no total domínio do estágio. Assim, o referido requisito passa a ter a seguinte estrutura:

- O esquema de cores do *template* do cartão deve estar alinhado com o fornecidos pelo cliente
- O design do *template* do cartão deve ser desenvolvido seguindo as melhores práticas de usabilidade e acessibilidade.

4.2.2 Abordagem para o Desenvolvimento

Para a cópia da aplicação ser bem sucedida é necessário editar todas as cores e logotipos da aplicação Power Apps de acordo com o novo cliente. Para além disso é necessário a criação de uma nova Azure Web App, para disponibilizar e partilhar os cartões digitais, numa aplicação Web, sem a necessidade de instalar a aplicação móvel.

Para a criação dos cartões digitais, é concebida total liberdade no seu design e implementação.

No que toca ao código QR, a mesma função Azure que foi utilizada na aplicação original irá ser utilizada nesta cópia, por isso não é necessário nenhum desenvolvimento adicional.

4.2.3 Soluções Implementadas e Desafios Encontrados

Para realizar esta reestruturação da aplicação já existente, começa-se por criar uma cópia da aplicação original, na qual são implementadas as alterações necessárias. As cores e os logotipos são ajustados utilizando as variáveis globais definidas na função `OnStart()` da aplicação, despertada quando esta é iniciada. A abordagem de definir o esquema de cores e os logotipos através dessas variáveis globais, revelou-se uma boa prática altamente eficaz. A definição das variáveis pode ser constatada na Figura 4.10.

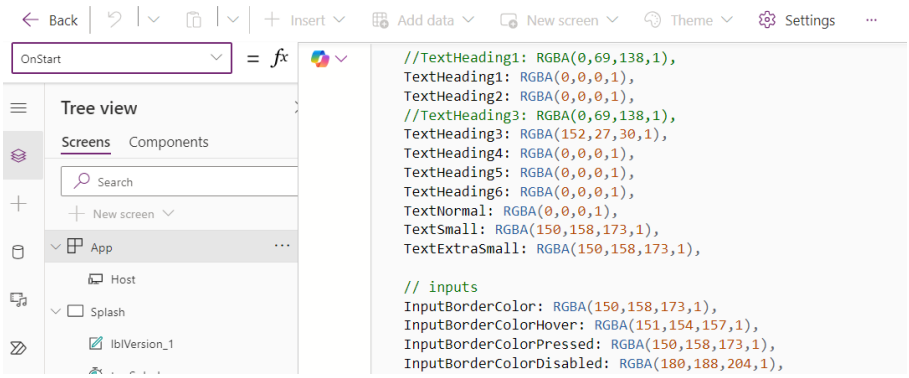
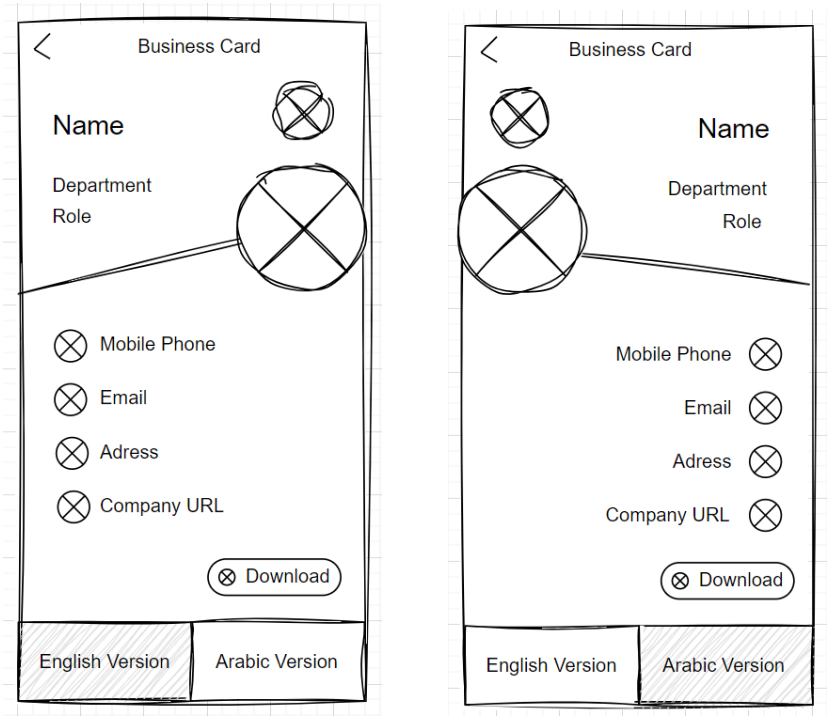


Figura 4.10: Função OnStart() de aplicação Power Apps, com a definição de variáveis globais a determinar a palete de cores.

Em relação ao cartão digital, como se tem toda a liberdade para a realização do design, efetuou-se inicialmente uma pesquisa preliminar para obter ideias e sugestões de *templates* existentes, que pudessem servir de referência, nomeadamente os da aplicação “HiHello” [10]. Após a identificação dos *templates* que se acha mais adequados, são desenvolvidos *wirefra-mes*, que constam na Figura 4.11.

Estes servem como representações estruturais para mostrar a organização e o *layout* dos elementos da interface de utilizador. Estes foram concebidos para as duas versões dos cartões, para língua inglesa, na Figura 4.11 (a) e para língua árabe, na Figura 4.11 (b). A seguir, foram adicionadas as cores e imagens pertinentes para conceber os *mockups*.



(a) Versão para língua inglesa (b) Versão para língua árabe

Figura 4.11: *Wireframes* desenvolvidos para o cartão digital vertical da nova aplicação

Assim como no projeto para o cliente anterior, foi garantida a consistência visual entre os cartões digitais apresentados na aplicação móvel e na aplicação Web. Para tal, os cartões digitais foram totalmente desenvolvidos em HTML e posteriormente integrados na aplicação móvel através do controlador de texto HTML.

Como se trata de um cartão vertical, de forma diferente à aplicação anterior, onde ambas as versões dos cartões (inglês e árabe) eram mostrados ao mesmo tempo na página Web, cada cartão digital da aplicação atual irá ocupar a página inteira. Para conseguir trocar entre cartões desenvolveu-se, na aplicação Web, dois botões em HTML, que recorrem à tecnologia de criação de conteúdo dinâmico em páginas Web Javascript, para esconder e mostrar cada versão do cartão digital. Já que o controlador de texto HTML na aplicação Power Apps não tem a capacidade de compilar Javascript, o mesmo processo teve de ser desenvolvido utilizando controladores nativos da Power Apps.

Toda a substituição dinâmica de informações consoante a sua omissão na base de dados foi realizada re-utilizando os algoritmos anteriormente desenvolvidos, tanto para a aplicação Web como para a aplicação móvel. A explicação destes algoritmos pode ser consultada na sub-secção 4.1.3.

Foi ainda realizado um guia auxiliar para a demonstração da aplicação, com todo o pro-

cesso de criação, edição e gestão dos cartões digitais. A demonstração ao novo cliente foi realizada com sucesso, atendendo aos seus requisitos e expectativas.

4.2.4 Avaliação e Resultados

A solução foi entregue dentro do prazo estipulado para o sprint e cumpriu todos os requisitos estabelecidos, contribuindo para uma completa apresentação da solução ao novo cliente.

Tanto a aplicação móvel, desenvolvida na plataforma Power Apps, quanto a aplicação Web, construída com a framework .NET, foram atualizadas para atender ao novo cliente. Todas as cores, imagens e detalhes da aplicação anterior foram ajustados para corresponder às novas especificações.

(a) Versão para língua inglesa (b) Versão para língua árabe

Figura 4.12: Cartões digitais verticais apresentados na aplicação Web

O design do cartão digital foi criado seguindo as normas de usabilidade e design. Inicialmente, as ideias principais foram esboçadas em *wireframes*, seguidas pelo desenvolvimento dos detalhes, como cores, imagens e fontes de texto, em *mockups*. Finalmente, o cartão foi implementado usando tecnologias Web tradicionais (HTML e CSS) para ambas as plataformas que apresentam os cartões na solução. Foi desenvolvido um design de cartão para cada idioma, inglês e árabe, com a troca entre os designs realizada por meio de botões na mesma

página de apresentação. A Figura 4.12 ilustra a página Web que exibe ambas as versões linguísticas dos cartões e os botões que permitem esta troca.

Realizaram-se testes de funcionalidade ao cartão de negócios digital, de modo a verificar se os requisitos funcionais como permitir a visualização das informações tanto em inglês como em árabe e descarregar o cartão para smartphone ou computador funcionam devidamente. Realizou-se ainda testes de usabilidade, de modo a verificar o compromisso com os requisitos não funcionais, fazendo entrevistas a colegas da empresa, para que avaliassem a interface do cartão digital em termos de funcionalidades e design, alterando alguns detalhes fornecidos pela equipa.

Assim, foi apresentada ao cliente uma aplicação totalmente personalizada para a criação, gestão e partilha de cartões digitais modernos.

4.3 Aplicação de Gestão de Auditorias

A solução, que visa criar a aplicação para gestão de auditorias no terreno, propõe utilizar os serviços da Microsoft Power Platform para responder à digitalização do atual processo de verificação e gestão de auditorias do cliente.

4.3.1 Requisitos Atribuídos

Este projeto teve início algum tempo antes da sua inclusão no contexto do estágio. A participação do estágio foi motivada pelo sucesso obtido na criação dos *templates* em HTML para a solução mencionada na secção 4.1 (Aplicação de Gestão de Cartões de Negócios Digitais - Cartões Horizontais. Os gestores do projeto atribuíram, ao âmbito do estágio, os seguintes requisitos desta solução.

Requisitos funcionais

- Após o fecho de cada inspeção, um relatório em PDF com as respetivas informações deve ser gerado e enviado ao oficial, Supervisores e Gerentes de Serviço.

4.3.2 Abordagem para o Desenvolvimento

O desenvolvimento da solução começou pela implementação da aplicação móvel, recorrendo-se à plataforma Power Apps, para a verificação e gestão de auditorias que ocorrem no local de trabalho do cliente, atendendo a todos os requisitos estabelecidos. Para suportar os dados desta aplicação irá utilizar-se o sistema de base de dados Dataverse, este consegue suportar grandes volumes de dados de maneira segura e eficiente. Tanto o design como a implementação da aplicação e da base de dados foram produzidos por colegas, inseridos desde o início do projeto.

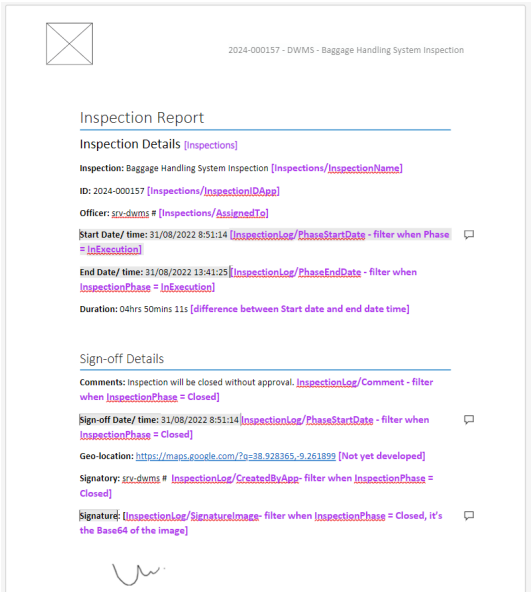
O contributo do estágio no projeto consiste numa automatização em Power Automate que pretende a criação dinâmica do relatório final das auditorias e o seu envio para as respetivas entidades. Este deve ser construído de forma dinâmica em HTML e estilos CSS, de modo a copiar o *template* criado. Posteriormente terá de ser convertido para PDF através dos conectores disponíveis na tecnologia.

O *template* de um relatório final exemplo, que descreve todos os problemas e recomendações inseridos na aplicação, foi criado através da tecnologia de produção de documentos Microsoft Word, também por colegas já inseridos no projeto.

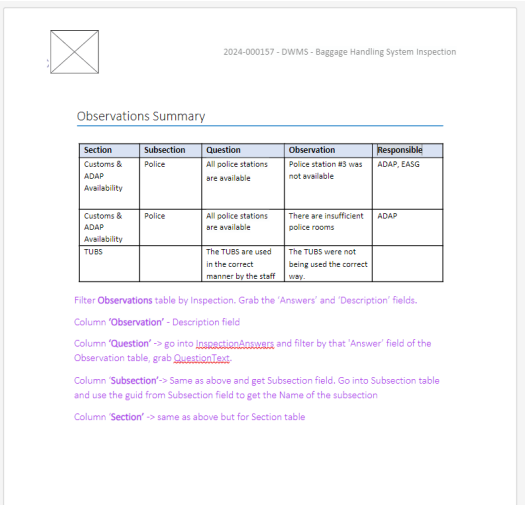
Durante todo o desenvolvimento da automatização do relatório final, em Power Automate, foi consultado a documentação da Microsoft em [34].

4.3.3 Soluções Implementadas e Desafios Encontrados

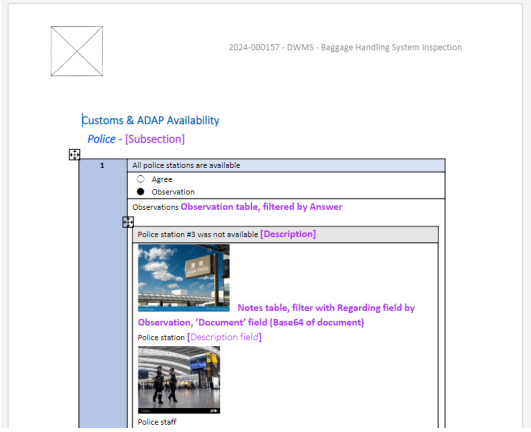
Começou-se por analisar o *template* do relatório para tentar perceber se era possível concebê-lo em HTML.



(a) *Template* do relatório em Word - Primeiro capítulo



(b) *Template* do relatório em Word - Segundo capítulo



(c) *Template* do relatório em Word - Terceiro capítulo

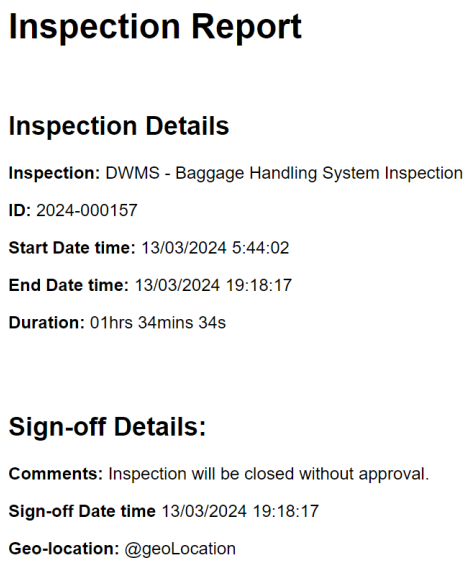
Figura 4.13: *Template* do relatório em Word

Verificou-se que o *template* continha três capítulos, cada um com um tipo de informação: um capítulo com uma lista estática, de detalhes da inspeção, com texto ou imagens, representado na Figura 4.13 (a); outro com um resumo da inspeção em formato de tabela dinâmica,

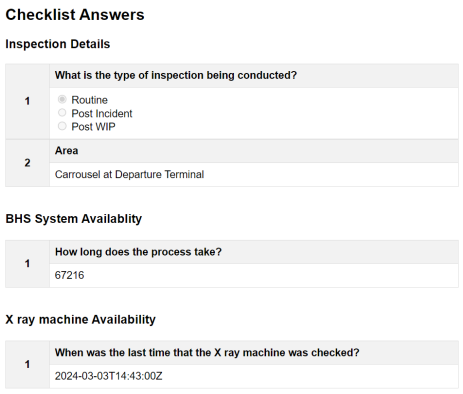
contendo apenas texto, representado na Figura 4.13 (b)); um último que detalha toda a inspeção, com secções e sub secções, cada uma com tabelas dinâmicas que podem conter ter texto, imagens ou e sub-tabelas, representado na Figura 4.13 (c)).

Conversão de HTML para PDF com conector OneDrive

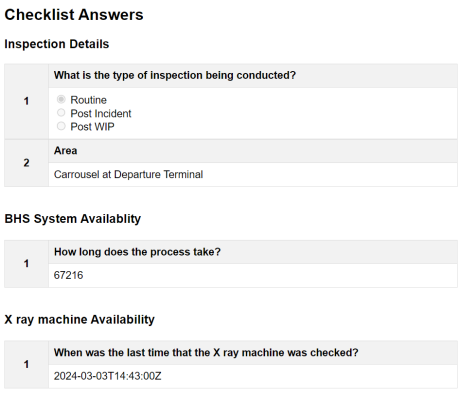
Após a verificação de todo o *template*, desenvolveu-se uma primeira do relatório estático em HTML, de modo a testar a concretização de cada capítulo com a tecnologia, resultando em algo bastante semelhante ao *template* original. Os resultados podem ser constatados na Figura 4.14 (a), (b) e (c).



(a) Recriação de teste do *template* - Primeiro capítulo



(b) Recriação de teste do *template* - Segundo capítulo



(c) Recriação de teste do *template* - Terceiro capítulo

Figura 4.14: Recriação de teste do *template* em HTML

De seguida, foi realizado o fluxo em Power Automate que constrói o relatório de forma dinâmica. Para isso é necessário recuperar as informações de cada inspeção. Como se pode observar nas Figuras 4.13 (a), (b) e (c) cada informação descrita no *template* contêm a explicação de como chegar ao seu valor através da base de dados, com esta é mais fácil a recuperação de informação. Através do conector Dataverse, invoca-se a operação “Get a row by id”, que através do nome da tabela da base de dados (presente na informação disponibilizada no *template*) e do ID da inspeção, devolve a informação desse item. Esta operação foi requisitada tantas vezes, quantas as necessárias, para recuperar toda a informação essencial ao preenchimento do relatório.

Começando pelo topo do relatório, para o primeiro capítulo, foi necessário preencher a lista de informações tanto para a secção “Inspection Details” como “Sign-off Details”, o seu *template* está presente na Figura 4.14 (a). Para a secção “Inspection Details” foi necessário: o nome da inspeção; o seu ID, o departamento; a data de inicio; a data de fim; e a duração, calculada subtraindo a data de fim à data de início. Para a secção “Sign-off Details” foi necessário: os comentário finais; a data e hora da assinatura; a geo localização; e a imagem da assinatura. Todos estes dados foram devidamente recuperados da base de dados, tratados (nos *scopes* “Date Difference” e “Get Inspection Log when ’Closed’” no topo da Figura 4.15) e incluídos nas *tags* HTML correspondentes ao *template* anteriormente concebido (observado na última operação da Figura 4.15).

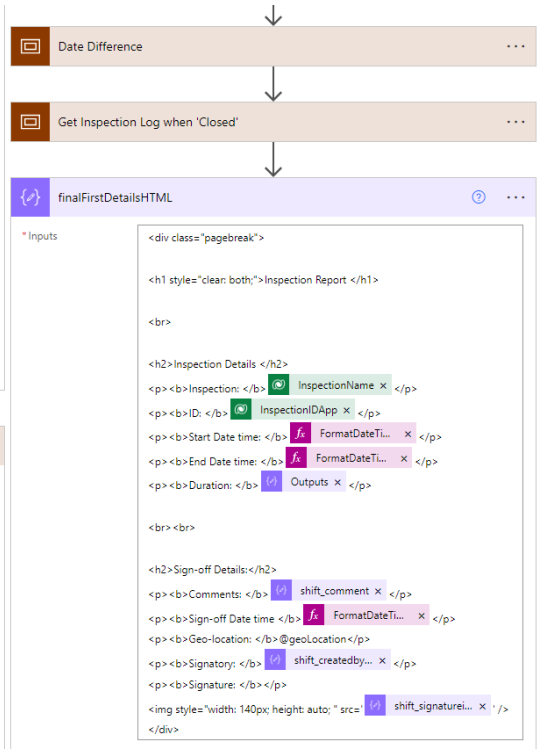


Figura 4.15: Inclusão das informações inicias para o relatório nas *tags* HTML

De seguida é necessário criar dinamicamente a única tabela que constitui segundo capítulo, “Observation Summary”, presente no *template* da Figura 4.14 (b), ou seja o preenchimento de cada linha dessa tabela com as respetivas informações. Para as recuperar, tem de se cruzar informações entre duas tabelas da base de dados de modo a filtrar os valores corretos e de seguida selecionar o valor pretendido desse filtro. Tradicionalmente, recorreria-se a uma operação “Filter Array” seguida de uma operação “Select” para cada informação. Porém, o número de informações que precisamos ainda é substancial e já que cada operação tem um custo no desempenho, utilizaram-se expressões XPath para realizar os filtros necessários. Assim, apenas é necessário uma operação “Select” (este retorna um *array* com cada informação) a selecionar as informações filtradas pelas expressão XPath, como se pode observar na primeira operação da Figura 4.16. Ainda nesta figura pode-se observar a ordenação das informações por ordem das questões, através da função Sort().

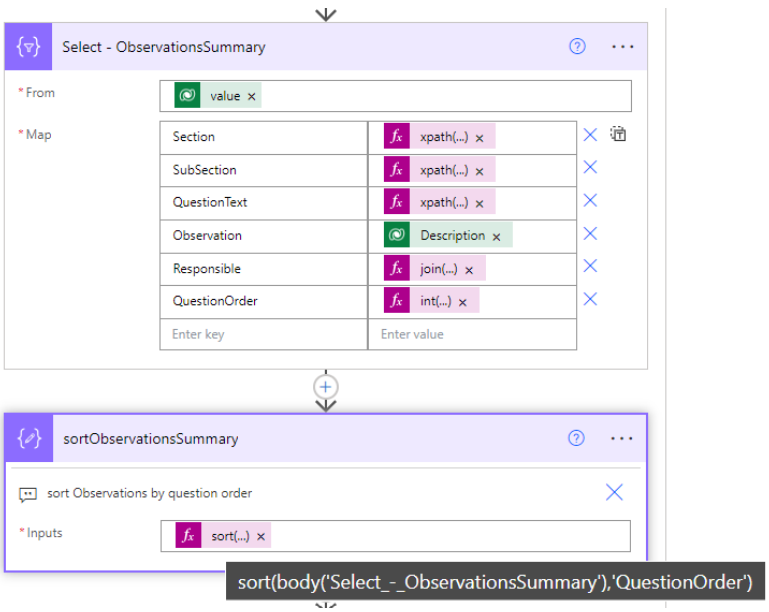


Figura 4.16: Operação ’Select’ com filtro recorrendo a expressão XPath, em Power Automate

As expressões XPath utilizadas têm todas o mesmo formato, mostrado no código presente na Listing 4.11. Estas faz um filtro na informação presente em “rootInspectionAnswers”, que é convertida para XML, e procede ao filtro baseado na comparação entre as informações “InspectionAnswerID” do XML a filtrar e a “_shift_answer_value” da operação “Select”. Depois de realizado o filtro seleciona-se a informação que pretendemos, neste caso será “Section” e acede-se ao primeiro valor retornado.

Listing 4.11: Expressão XPath para filtro de informação através de cruzamento entre tabela de base de dados e operação Select em Power Automate

```
xpath(
    xml(outputs('rootInspectionAnswers')),
    concat(
        '//Array[InspectionAnswerID/text()="',
        item()?['_shift_answer_value'],
        '"]/Section/text()'
        )
    )[0]
```

Após recuperar-mos todas as informações necessárias ao preenchimento da tabela de obervações, utiliza-se outra operação "Select" para criar cada linha da tabela em HTML em forma de *array*, este processo é mostrado na Figura 4.17.

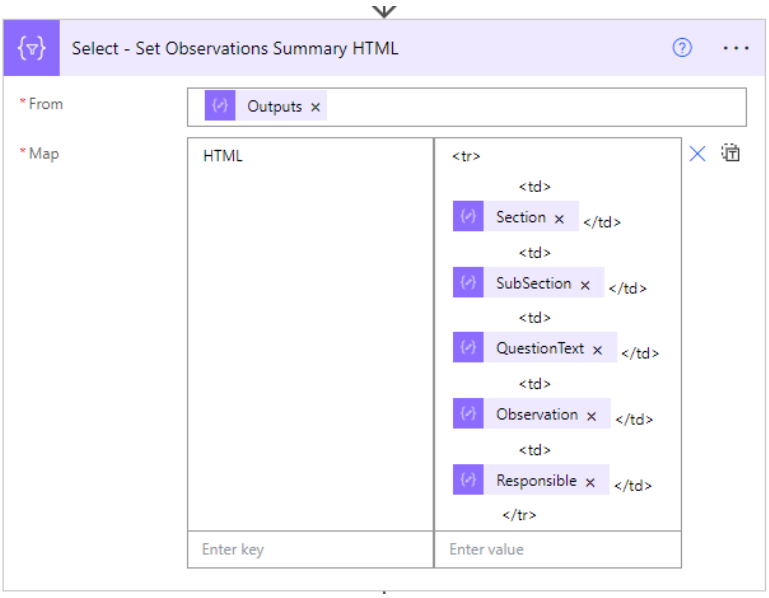


Figura 4.17: Composição, em Power Automate, de uma linha da tabela de observações recorrendo a HTML

Por forma a concluir a tabela de observações, é necessário tratar cada *array* que compõe as linhas da tabela HTML e introduzir o respetivo seu *header*, como observado na Figura 4.18.

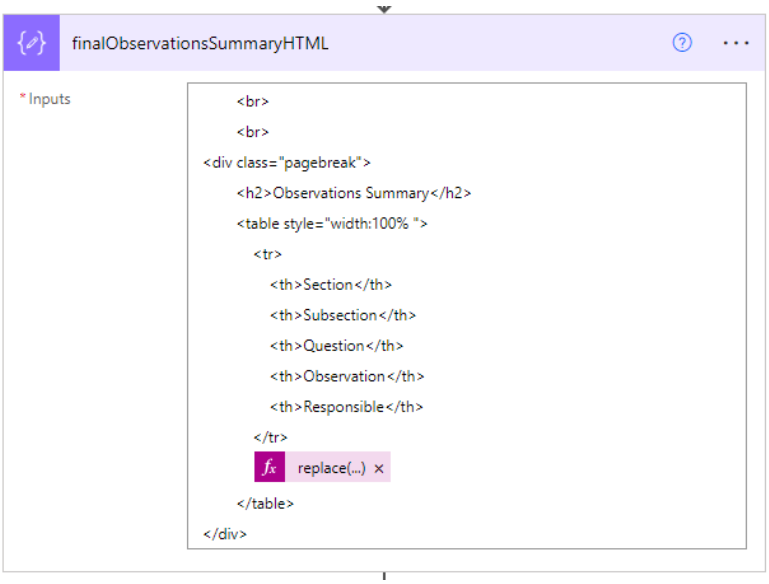


Figura 4.18: Composição, em Power Automate, da tabela de observações recorrendo a HTML

De forma semelhante à anteriormente referida, o terceiro e último capítulo do relatório, denominado “CheckList Answer”, é criado, por forma a preencher cada secção e subsecção. Para isto, primeiro estrutura-se, em árvore, toda a informação necessária e de seguida percorre-se cada um dos nós da árvore de modo a incluir as *tags* HTML necessárias. A parte inicial deste processo está presente na Figura 4.19, onde se pode observar os ciclos iniciais a percorrer cada secção e sub-secção enquanto se insere as *tags* HTML.

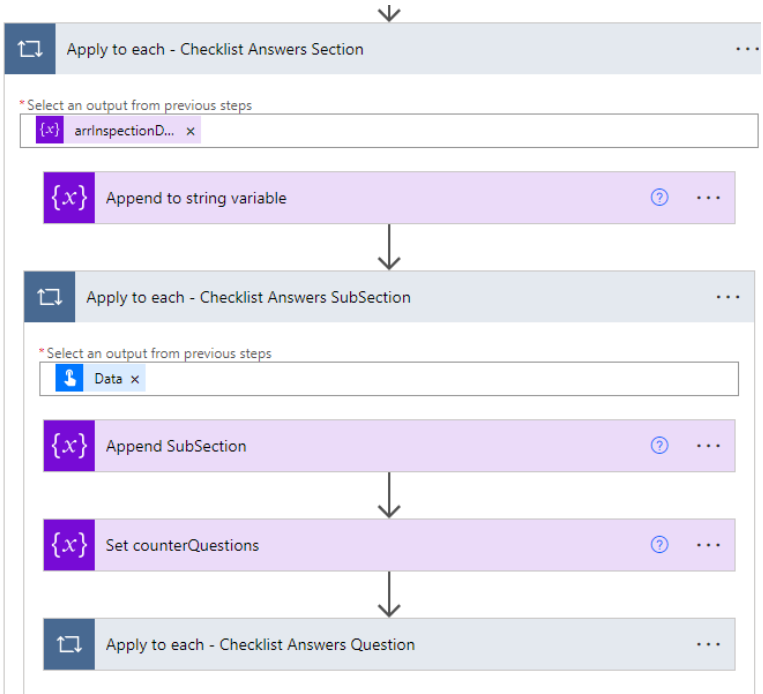


Figura 4.19: Composição, em Power Automate, da tabela de observações recorrendo a HTML

Finalmente juntam-se todos os capítulos e inclui-se o respetivo CSS, para fornecer ao documento o estilo pretendido.

De modo a converter o texto HTML em PDF utilizou-se algumas operações do conector OneDrive em sequência. Com a operação “Create file” começou-se por criar um documento *.html*, proveniente do texto HTML realizado no fluxo. De seguida converteu-se, com a operação “Convert file”, o documento *.html* para um documento PDF, de seguida criou-se esse ficheiro dando-lhe a terminação *.pdf*.

Depois deste processo, ao verificar-se o resultado, observa-se algumas inconsistências no resultado da conversão. O documento não contém nem cabeçalho, nem rodapé, nem o conector permite adicionar estas caraterísticas, o que torna o relatório inutilizável. Para além disso, a quebra de página gerada, pelo conversor, é aleatória podendo cortar qualquer elemento ao meio de forma brusca, por exemplo uma imagem ou tabela permanecer metade na página anterior e a outra metade na página seguinte. Ainda se recorreu a estilos CSS e até a um script Javascript para tentar uma quebra de página inteligente, porém o conversor do OneDrive não mantêm essas alterações.

Tentou-se, a conversão utilizando o conector Adobe, que requer um arquivo ZIP com um formato específico como entrada. No entanto, apesar das várias tentativas feitas, este formato não foi conseguido no seguimento de um fluxo Power Automate.

Tentou-se ainda a conversão de HTML para o formato Word de modo a usar alguma operação do conector Microsoft Word que permitisse a inclusão do cabeçalho e rodapé em falta, porém esta conversão era impossível através de Power Automate. Aquando desta pesquisa pelo conector Microsoft Word, encontrou-se uma operação denominada “Populate a Microsoft Word Template”. Após o preenchimento de um *template Word*, é possível converter esse documento para PDF, o que pode ser uma solução eficaz após os problemas encontrados na solução em HTML, testada anteriormente. Para melhor entender como se utiliza esta operação analisou-se o artigo da Microsoft *Generate Word Document Template Using Power Automate* [35].

Preenchimento de documento Word através de controladores

Esta operação utiliza a caraterística dos documentos Word, os controladores. Estes permitem atribuir *tags* a elementos de um documento Word que podem ser acedidas por ferramentas exteriores ao documento, como por exemplo o Power Automate. Esta caraterística é especialmente útil para criar formulários ou documentos que exigem entradas de dados estruturadas e repetitivas.

Recorreu-se ao controlador de conteúdo de texto simples (*plain text*), o controlador de conteúdo de imagens e o controlador de conteúdo repetitivo. Desenvolveu-se o *template* do relatório em Word, utilizando esses controlador com *tags*, que podem ser substituídas em Power Automate, através da operação "Populate a Microsoft Word Template".

Utilizou-se conteúdos de texto simples para o primeiro capítulo, já que este contém um número de informações fixo. Para o segundo capítulo, utilizou-se o controlador de conteúdo repetitivo com vários controlador de texto simples, por forma a realizar as várias linhas da sua tabela dinâmica. Para o terceiro capítulo, a estrutura é um pouco mais complexa pois este tem secções e sub-secções com tabelas e sub-tabelas que podem conter dinamicamente várias imagens. Isto envolve ter controladores de conteúdo repetitivo e controladores de conteúdo de imagens dentro de outros controladores de conteúdo repetitivo, todavia isso não é suportado. Apenas é possível adicionar o controlador de texto simples dentro dos controladores de conteúdo repetitivo.

De modo a solucionar a impossibilidade de recriar o terceiro capítulo como foi idealizado no *template* original, recorreu-se a uma versão em que este terceiro capítulo apenas contivesse texto simples e que a distinção entre sub-secções fossem alcançadas a partir de letras maiúsculas e prefixos.

A solução para as várias imagens presentes no relatório, é colocá-las num último capítulo, denominado anexos, porém este tem de ter tantos controladores de conteúdos de imagens quantos queremos popular, o que prejudica um pouco o dinamismo do documento. Chegou-se à conclusão que se poderia incluir um número superior de conteúdos de imagens, ao expectável, e que os que não fossem utilizados deixar vazios.

O terceiro capítulo do documento Word, com os diferentes controlos usados, encontra-se na Figura 4.20. Nesta encontram-se controlos como"CheckList Answers Section" que tem como objetivo conter o nome de cada secção e o controlo "CheckList Answers Content" que irá conter o conteúdo dessa secção e as suas sub-secções. Estes controladores são abrangidos por um controlador de conteúdo repetitivo por forma a repetir várias secções.

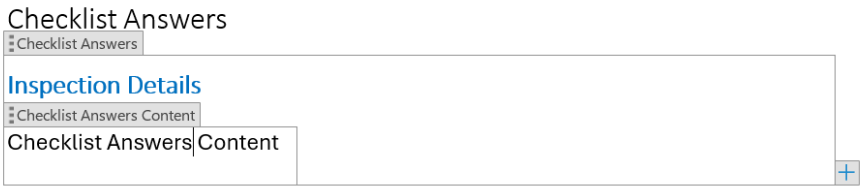
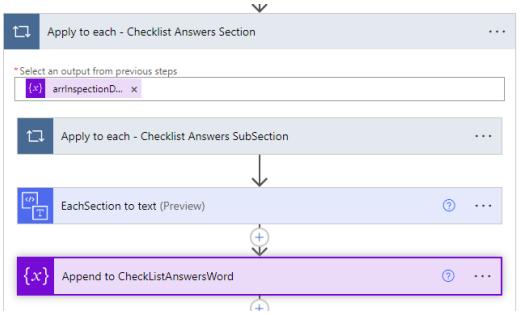
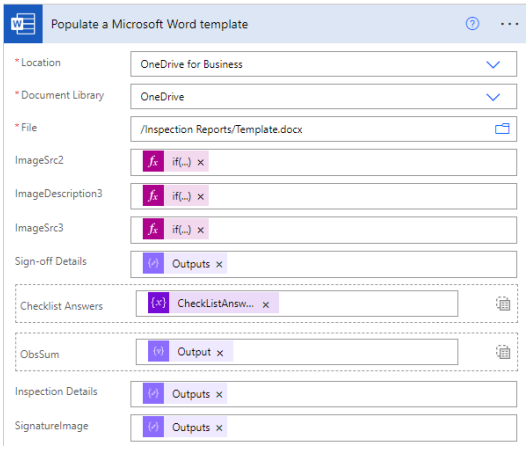


Figura 4.20: Controlos utilizados no terceiro capítulo do relatório em documento Word

Após a elaboração do *template* Word, com todos os controladores necessários, é preciso preenche-los através da devida operação “Populate a Microsoft Word Template”. Este processo pode ser constatado nas Figuras 4.21. Na Figura 4.21 (a), a estruturação prévia de cada informação para os controladores que compõe o terceiro capítulo (“CheckList Answers Section” e“CheckList Answers Content”). Na Figura 4.21 (b), a inserção das informações nos controladores Word, através da operação “Populate a Microsoft Word Template”, inclusive do controlador de conteúdo repetitivo, que compões o terceiro capítulo, “CheckListAnswers”.



(a) Estruturação prévia de cada informação para os controladores do terceiro capítulo



(b) Inserção das informações nos controladores Word através do Power Automate

Figura 4.21: Preenchimento de controladores de um *template* Word, através do Power Automate

Insatisfeitos com os resultados da solução implementada, a equipa estava determinada a encontrar uma melhor alternativa. Após alguma pesquisa encontrou-se o conector Encodian.

Conversão de HTML para PDF com conector Encodian

Encodian é uma plataforma de automação que oferece integração com o Microsoft Power Automate. Através do seu conector, oferece uma série operações que permitem a manipulação avançada de documentos, especialmente para tarefas relacionadas com documentos PDF, Word e imagens. Porém este conector é premium, o que significa que não faz parte do pacote da assinatura da Microsoft e tem um custo adicional. Resolveu-se realizar alguns testes com a versão de testes gratuita do conector, de modo a mostrar ao cliente os seus resultados.

Recorreu-se à abordagem inicial, obtendo a construção de todo o relatório em HTML. Utilizou-se a operação Encodian “Convert HTML to PDF”. Esta é bastante completa contendo enumeras definições como a orientação das páginas, a margem esquerda, direita, superior e

inferior. De seguida, utilizou-se a operação Encodian “Add HTML Header or Footer to PDF”. Através desta é possível adicionar o cabeçalho necessário de forma a igualar o *template* original do relatório. Recorreu-se ainda à operação “Add page Number” para adicionar a numeração da página, esta contém definições como o inicio da paginação, o número de página inicial, a formatação da paginação, o alinhamento horizontal e a fonte utilizada. Esta sequência de operações pode ser visualizada na Figura 4.22. Por fim, guardou-se o documento na base de dados Dataverse.

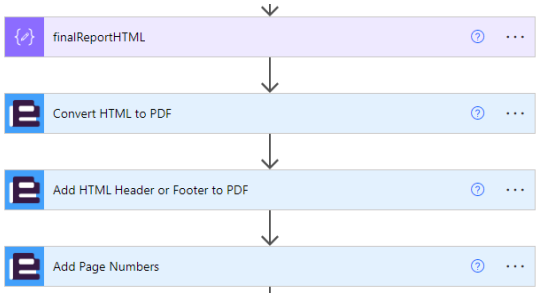


Figura 4.22: Sequência de operações de forma a construir um documento PDF através de HTML, utilizando o conector premium Encodian

As três alternativas elaboradas, foram entregues ao gestor de projeto para que fossem testadas e apresentadas ao cliente.

4.3.4 Avaliação e Resultados

O projeto foi concluído com sucesso. Embora fosse previsto simples de resolver, acabou por se revelar mais complexo do que o esperado, o que exigiu a elaborações de três soluções alternativas, para que o cliente pudesse escolher. Mesmo assim, a solução foi concluída com sucesso, atendendo a todos os requisitos estabelecidos e dentro dos prazos definidos.

Conversão de HTML para PDF com conector OneDrive

Primeiro, implementou-se a solução originalmente concebida pela equipa, com o objetivo de construir um relatório dinâmico em HTML, utilizando um fluxo no Power Automate. Este visa a utilização do conector OneDrive para fazer a conversão de um documento HTML para PDF.

Depois de todo o processo, da criação e preenchimento do relatório em HTML, a conversão deste formato para PDF, do conector OneDrive, não acompanhou as expectativas da equipa. Através a realização de testes de funcionalidade, verificou-se a conversão que é possível, através deste conector, não adiciona três elementos fundamentais ao documento PDF. O

documento não possuí nem cabeçalho nem rodapé, sem a possibilidade, na operação de conversão, de adicionar nenhum deles. Para além disso a quebra de página é aleatória, podendo cortar qualquer elemento ao meio de forma brusca.

Os resultados são os apresentados na Figura 4.23.

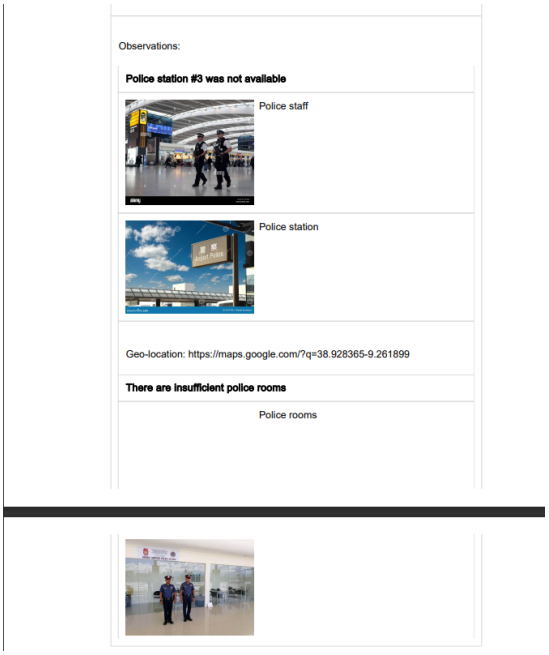


Figura 4.23: Resultados da conversão de HTML para PDF utilizando o conector OneDrive

A conversão de HTML para PDF através do conector OneDrive, revelou-se pouco prática já que não cria alguns dos elementos fundamentais à normal formatação do documento, como esperado. Dada a situação desta versão da solução, não pode ser utilizada, exigindo assim a consideração de uma alternativa. Esta situação podia ter sido evitado, se tivessem sido realizados testes unitários à operação de conversão do conector OneDrive, antes do início do desenvolvimento do relatório em HTML.

Preenchimento de documento Word através de controladores

Investigou-se outras maneiras de proceder à construção dinâmica do relatório. Um das alternativas é a utilização da operação “Populate a Microsoft Word Template” do conector Microsoft Word. Pensou-se que este seria uma boa alternativa, já que permitia realizar dinamicamente o relatório com a informação das inspeções.

Ao contrário da abordagem anterior, começou-se por fazer testes unitários a esta operação, de modo a verificar as suas capacidades. Através destes testes, percebeu-se que este também contêm algumas limitações. Notou-se que o controlador de conteúdo repetitivo apenas consegue repetir controladores de texto simples. Isto faz com que o último capítulo

do relatório, que contém tabelas, sub-tabelas, secções e sub-secções podendo ainda conter várias imagens é impossível de recriar.

A alternativa encontrada foi a construção desse último capítulo, em texto simples, como se de uma lista se tratasse com o conteúdo das tabelas e sub-tabelas a ser mostrado através de letras maiúsculas ou prefixos. Adicionalmente foi adicionado um capítulo de anexos, neste colocou-se um número de controladores de conteúdos multimédia um pouco acima do número médio de imagens que o cliente utiliza, já que é impossível associar-lhe um controlador de repetição de conteúdo. Os controladores que não forem utilizados deixam-se vazios.

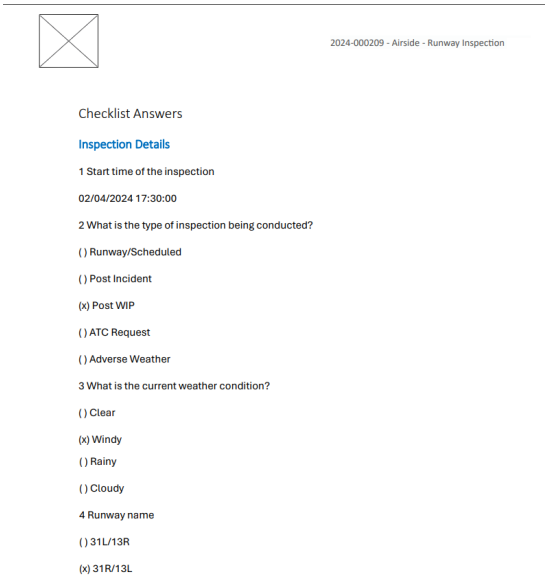


Figura 4.24: Resultados da criação dinâmica de um relatório populando um *template* Word em Power Automate

Os resultados obtidos para a criação dinâmica do relatório, populando um *template* Word em Power Automate são apresentados na Figura 4.24. Nesta Figura, observa-se parte do terceiro capítulo de um relatório, gerado automaticamente pelo fluxo realizado, utilizando o conversor do conector OneDrive. Pode verificar-se a utilização de listas para a substituição das tabelas e sub-tabelas.

Foram realizados testes de funcionalidade que demonstraram que os documentos finais gerados, por esta alternativa, não atendiam aos requisitos do cliente.

Conversão de HTML para PDF com conector Encodian

Após testes unitários ao conector Encodian, este revelou-se o mais bem preparado para lidar com situações de criação e conversão de documentos. Este oferece um grande número

de operações, relacionadas com documentos, todas elas com imensas opções e definições possíveis. Através deste conseguiu-se replicar em detalhe o *template* originalmente criado, através da excelente conversão do documento HTML para PDF.

Ao contrário do que aconteceu com o conector OneDrive, a conversão do conector Encodian resulta em quebras de páginas inteligentes, sem cortar nenhuma elemento do documento ao meio, de forma brusca, resultando num documento sem desformatações. Permite ainda a adição de cabeçalho e rodapés com imensas opções disponíveis. Os resultados de um relatório utilizando o conector Encodian podem ser constatados na Figura 4.25. Através de testes funcionais, conclui-se que esta solução é a que produz os resultados mais fieis ao *template* originalmente pensado, com a correta utilização de tabelas dentro de tabelas, cabeçalho, rodapé e com quebras de página dinâmicas e que mantêm formatação, cumprindo os requisitos do cliente.

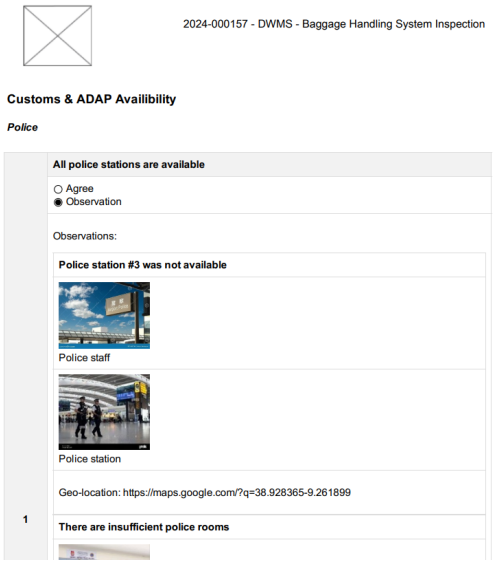


Figura 4.25: Resultados da conversão de HTML para PDF utilizando o conector Encodian

No final do sprint, foram realizados testes de desempenho para avaliar o tempo de execução do fluxo, especialmente em cenários de maior carga, onde relatórios com um número elevado de inspeções e dados extensos foram processados. Através destes testes conclui-se que o processo de geração e envio dos relatórios mantém tempos de resposta aceitáveis, mesmo em situações de maior complexidade. Após estes testes o projeto avançou para a fase de UAT com o objetivo de validar a solução num ambiente que simula condições reais de utilização por utilizadores finais.

4.4 Automatização de Agendamento de Reuniões Internas

Esta solução visa automatizar o agendamento de reuniões internas semanais entre os colaboradores da Make The Shift, promovendo a comunicação e colaboração dentro da empresa.

4.4.1 Requisitos Atribuídos

O projeto foi iniciado em uma fase mais avançada do estágio, quando a equipa já tinha desenvolvido confiança na capacidade de contribuição para as soluções. Dito isto, a responsabilidade pelo projeto foi atribuída totalmente ao contexto do estágio, o que implica o cumprimento de todos os requisitos estabelecidos.

Requisitos Funcionais

- A automatização deve agrupar aleatoriamente os colaboradores para reuniões semanais.
- O sistema deve enviar convites automáticos para todos os participantes, com os detalhes da reunião, data e hora.
- Os convites e notificações devem ser integrados com o calendário e o sistema de comunicação da empresa.
- As reuniões devem ser agendadas num horário que maximize a participação e a disponibilidade dos colaboradores.

Requisitos Não Funcionais

- A automatização deve ser eficiente e executar o agrupamento e o envio de convites sem atrasos significativos.
- A interface de gestão da automatização deve ser intuitiva e de fácil utilização para os administradores responsáveis pela configuração e monitoramento dos agendamentos.

4.4.2 Abordagem para o Desenvolvimento

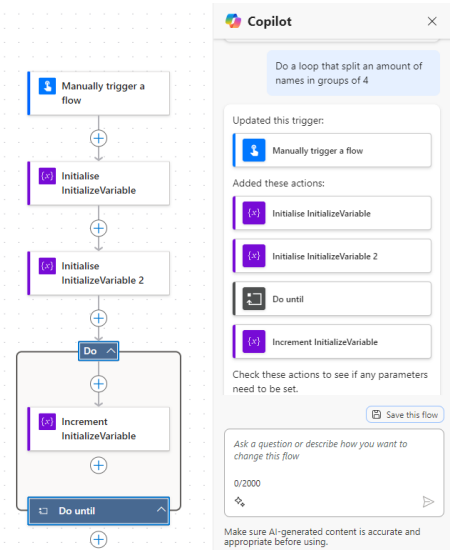
Com esta solução pretende-se que em cada semana sejam agendadas várias reuniões, cada uma com diferentes colegas da empresa. Na semana seguinte o sistema deve tentar

distribuir os colegas em grupos diferentes das semanas anteriores, de modo a rodar os colegas pelos diferentes grupos. Para esta automatização utilizou-se a ferramenta da Microsoft Power Platform, Power Automate e os seu conector Microsoft Teams.

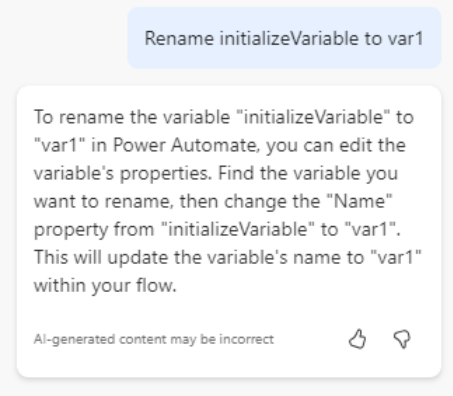
4.4.3 Soluções Implementadas e Desafios Encontrados

Por se tratar de uma solução interna e tendo mais liberdade no seu desenvolvimento, explorou-se a integração de inteligência artificial. Este tema é bastante atual e relevante, além de se alinhar com a proposta do estágio: "Desenvolvimento Front-End e Back-End de aplicações que solucionam problemas reais com a integração de Inteligência Artificial".

Para isso, começou-se por utilizar a integração da inteligência artificial generativa em tempo real da Microsoft, denominado "Copilot". Esta integração na tecnologia Power Automate, consegue proceder à criação e configuração de operações automaticamente, através de comandos livres.



(a) Pedido 1 de automatização à integração de inteligência artificial generativa em Power Automate



(b) Pedido 2 de automatização à integração de inteligência artificial generativa em Power Automate

Figura 4.26: Exemplo de pedido e resposta de inteligência artificial generativa para automatização automática usando a tecnologia Power Automate

Como consta na Figura 4.26, foi introduzido na área de comandos do "Copilot" o seguinte pedido: "Do a loop that splits an amount of names in groups of 4". Inicialmente, a inteligência artificial generativa, parece conseguir captar bem o pedido, adicionando uma variável com uma lista de vários nomes, uma variável vazia e uma operação de ciclo na automatização. Porém a operação de ciclo não estava bem definida, nem dividia a lista de nomes em grupos.

O pedido e a automatização criada pelo “Copilot” podem ser verificado na Figura 4.26 (a). Quando foi introduzido o seguinte pedido: “Rename InitializeVariable to var1”, para renomear uma variável presente na automatização, a resposta foi a explicação de como se procedia à renomeação da variável manualmente e não a sua realização automática pela tecnologia. Esta resposta consta na Figura 4.26 (b).

Como estes pedido, muitos outros foram feitos à integração do “Copilot”, porém esta integração parece muito limitante. Não consegue renomear nem afetar variáveis, ciclos ou operações já criadas. Quando cria algumas operações raramente é exatamente aquilo que é requisitado. Muitas vezes responde que não consegue entender os pedidos ou com explicações de como se há-de realizar o pedido manualmente pelo utilizador. Devido a restrições de tempo decidiu-se proceder à continuação da automatização manualmente.

A solução implementou-se recuperando todos os e-mails dos colegas da empresa e colocando-os numa lista, com a operação “List group members”. Esta operação devolve uma lista de todos os membros do grupo Teams geral da empresa e os seus dados, tais como nome, título, correio eletrónico, etc, onde se escolheu apenas o e-mail de cada um.

De seguida, distribuiu-se aleatoriamente as pessoas da empresa por grupos, estes com um determinado número de participantes, até não haver mais pessoas para atribuir. No final deste processo obtém-se vários grupos de pessoas no formato de listas.

Percorreu-se cada grupo, e para cada um cria-se uma reunião via Teams com a operação “Create a Teams meeting”. Esta operação permite criar uma reunião, acompanhada por uma ligação na parte inferior do convite para participar na mesma, o agendamento é associado automaticamente com o calendário Microsoft Teams. Através desta operação configurou-se o tema e corpo que acompanha o convite da reunião, o dia a agendar, a hora de início e de término e ainda com quanto tempo de antecedência o aviso deve ser despertado. O processo de agendar uma reunião para cada grupo pode ser verificado na Figura 4.27.

No final, foi implementado um ciclo que permite que o agendamento ocorra para tantas semanas seguintes quantas o administrador desejar.

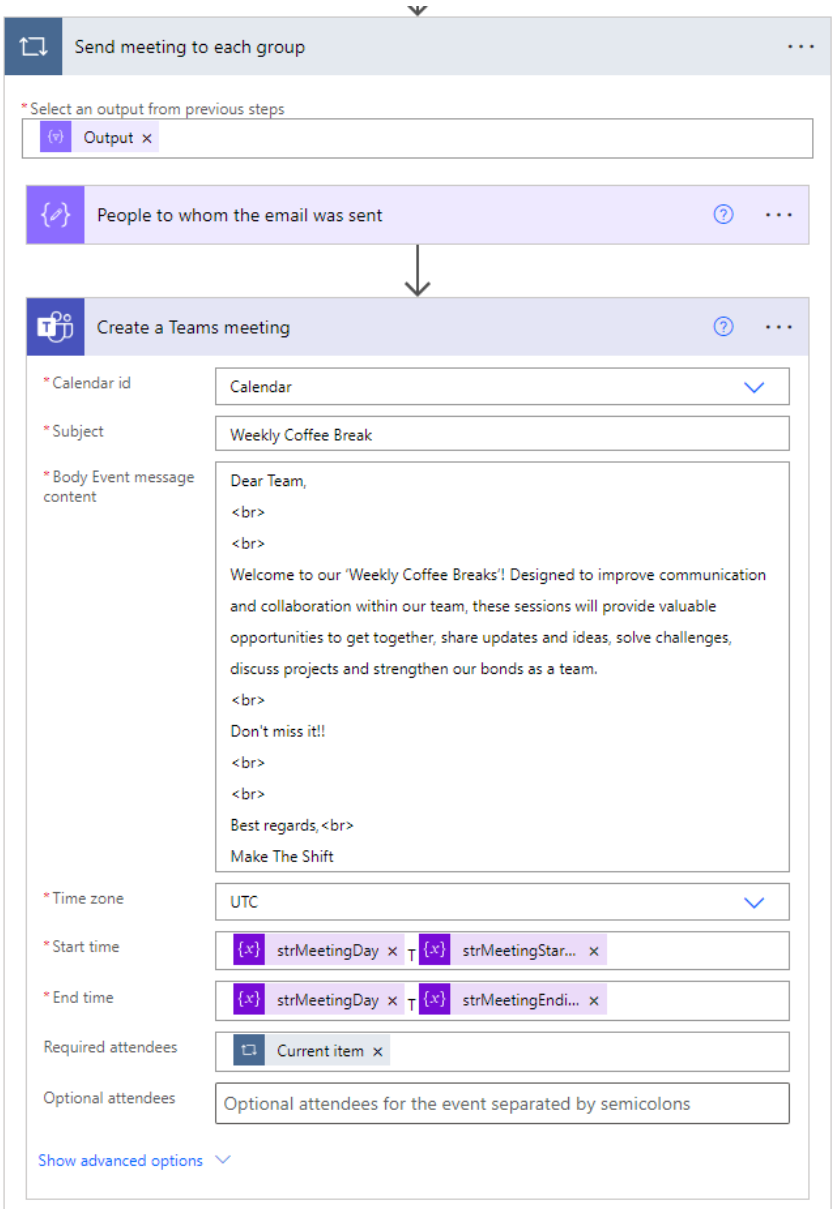


Figura 4.27: Agendar uma reunião para um dado grupo através do Power Automate

De forma à solução ser o mais flexível na sua configuração, alguns dados foram adicionados por forma a serem dinâmicos. Dados como:

- Número de semanas a agendar
- Número de participantes (membros) por grupo
- Hora de início da reunião
- Hora de termino da reunião

Estas informações foram colocadas como variáveis de ambiente de modo a que o administrador possa manipulá-las sem ter que editar a própria automatização.

O fluxo da automatização é descrito detalhadamente nos itens seguintes:

- **O dia da reunião** é 2 dias após a execução do fluxo
- Recuperar a **lista geral** de todos os membros de um grupo de equipas
- **Repetir** até que todas as semanas tenham grupos criados:
  - **Repetir** até que não haja mais membros na lista geral:
    - **Repetir** para o número de membros de cada grupo:
      - **Escolher aleatoriamente** um membro da lista geral de todos os membros
      - Adicionar o membro a um **grupo**
      - Adicionar o membro à **lista geral baralhada** (de forma a reiniciar o ciclo com os dados baralhados)
      - Remover o membro escolhido da **lista geral**
    - Adicionar o grupo de membros a uma lista de grupos contendo **todos os grupos criados** (para uma semana específica)
    - **Reiniciar** o grupo de membros
  - Quando **não houver mais membros na lista de todos** e **todos os grupos de membros estiverem definidos**, é necessário verificar **se o último grupo tem um tamanho definido**. Se não tiver:
    - **Remover o último grupo** da lista de grupos
    - **Repetir** até que o último grupo não tenha membros:
      - Colocar o **primeiro membro do último grupo** no **primeiro grupo da lista de grupos**
      - Colocar o **primeiro grupo da lista de grupos** como **o último grupo da lista**
      - **Remover** o primeiro membro do último grupo
  - **Tratar a lista de grupos** removendo caracteres indesejados
  - **Criar agendamento** para cada grupo da lista de grupos
  - **Substituir** a lista geral vazia pela lista geral baralhada
  - **Adicionar 7 dias ao dia da reunião** e substituí-lo
  - **Subtrair 1 ao número de semanas** a agendar

Para evitar que os mesmos membros fiquem juntos em semanas consecutivas, baralhou-se sempre a lista geral de membros antes de os agrupar de novo e proceder a um novo sorteio.

Para garantir que cada grupo tenha um número semelhante de membros e evitar que algum grupo tenha significativamente menos membros do que o número escolhido, distribuíram-se os membros do último grupo criado (com menos membros) entre todos os outros grupos. Desta forma, em vez de o último grupo criado ter um tamanho visivelmente menor, todos os grupos terão tamanhos ligeiramente maiores com os membros distribuídos.

Por exemplo, em vez de ter grupos com membros como 5, 5 e 2 membros, a distribuição resultaria em dois grupos de seis (6 e 6), assegurando uma alocação mais equilibrada.

Para alcançar isto, adicionou-se o primeiro membro do grupo mais pequeno ao primeiro grupo dos maiores e removeu-se do grupo mais pequeno. Em seguida, o primeiro grupo dos maiores será movido para o final, e este processo continuará até não haver mais membros no grupo mais pequeno.

Conclui-se assim com sucesso o primeiro projeto totalmente atribuído ao contexto do estágio.

Com a má prestação da integração da inteligência artificial generativa na concepção automática da automatização, pretendeu-se continuar a investigar como integrar inteligência artificial na solução. Para isto investigou-se o conector de inteligência artificial da Microsoft, AI Builder. Este conector permite diversas operações utilizando inteligência artificial para detetar voz, traduzir texto, descrever imagens entre outros. Algumas ideias para integrar na solução atual foram:

Ideia 1

- Utilizar questionários do Microsoft Forms para recolher informações sobre os participantes;
- Verificar cada reunião e quem está a participar;
- Utilizar a inteligência artifical generativa do conector AI Builder para gerar tópicos de discussão e/ou mini-jogos com base nos interesses dos participantes (operação “Create text with GPT using prompt”)

Ideia 2

- Transcrever gravações de áudio das reuniões para texto (operação “Extract information from invoices”)

- Processar a transcrição para gerar resumos concisos, notas de reunião, destacar pontos-chave ou decisões (operação “Extract key phrases from text”)
- Criar uma base de dados para armazenar os registos das reuniões
- Criar mais tópicos de discussão com base nestas informações recolhidas (operação “Create text with GPT using prompt”)

Devido às restrições de tempo e às prioridades estabelecidas na empresa, estas iniciativas não foram desenvolvidas. Embora a exploração de soluções baseadas em aprendizagem automática e inteligência artificial tenha sido considerada promissora, optou-se por concentrar os esforços em atender aos requisitos funcionais e a entrega imediata, assegurando a conclusão dentro dos prazos definidos. Além disso, com vários outros projetos à espera de equipas de desenvolvimento, o foco foi direcionado para atender às necessidades mais urgentes.

Todavia, a implementação dessas ideias poderá ser revista em fases futuras, caso haja disponibilidade de tempo e recursos. A integração de inteligência artificial nas soluções continua alinhada com os objetivos estratégicos da empresa, que procura constantemente inovação de forma para melhorar as suas soluções.

4.4.4 Avaliação e Resultados

Entregou-se a solução com sucesso, dentro do prazo previsto no sprint, uma ferramenta capaz de gerir todos os colaboradores da empresa, criando grupos aleatórios para participarem na reunião informal semanal. De forma a evitar repetir, muitas semanas seguidas, os mesmos colaboradores no mesmo grupo baralhou-se sempre a lista geral de membros antes de os agrupar de novo e proceder a um novo sorteio para a semana seguinte. Embora não elimine completamente a possibilidade de os mesmos colaboradores ficarem juntos em semanas consecutivas, o processo é aleatório o suficiente para minimizar esta repetição, garantindo o requisito funcional: “A automatização deve agrupar aleatoriamente os colaboradores para reuniões semanais, garantindo que diferentes pessoas sejam incluídas em cada semana”.

Após diversos testes com a integração do “Copilot” no Microsoft Power Automate, concluiu-se que, embora o “Copilot” seja promissor e possa, num futuro próximo, desempenhar um papel importante na otimização de processos, a sua eficácia, no momento, é bastante limitada. Simples tarefas, como atribuir valores a variáveis ou ajustar ciclos já existentes, não foram executadas corretamente, revelando que a ferramenta ainda não tem capacidade para construir ou modificar automatizações de forma autônoma e eficaz. Isso significa que não é possível realizar um projeto completo apenas com a sua assistência.

Contudo, quando a ferramenta não consegue realizar um comando, oferece orientações e dicas sobre como o utilizador pode realizar o seu pedido manualmente. Assim, apesar das suas limitações, o “Copilot” apresenta potencial como suporte educativo e facilitador para novos utilizadores da tecnologia, sendo uma ajuda importante, mesmo que ainda longe de ser uma solução totalmente eficaz.

A pedido do gestor do projeto, as reuniões têm de ser agendadas num horário previsível e fixo todas as semanas, para que os colaboradores consigam marcar reuniões com os clientes agilmente. Dito isto todas as reuniões foram agendada para o mesmo período de tempo, em todas as semanas consecutivas. Assim atendeu-se ao requisito que diz “As reuniões devem ser agendadas num horário que maximize a participação e a disponibilidade dos colaboradores”, de forma menos automática mas eficaz.

De modo a cumprir o requisito não funcional, “A interface de gestão da automatização deve ser intuitiva e de fácil utilização para os administradores responsáveis pela configuração e monitoramento dos agendamentos”, informações como: Número de semanas a agendar; Número de participantes (membros) por grupo; Hora de início da reunião; Hora de termino da reunião; foram colocadas como variáveis de ambiente de modo a que o administrador possa manipulá-las sem ter que editar a própria automatização.

Utilizando as tecnologias do Power Automate, que possui uma integração robusta com o Microsoft Teams, foi possível cumprir o requisito não funcional de que "A automatização deve ser eficiente e executar o agrupamento e o envio de convites sem atrasos significativos". Através desta integração, os requisitos funcionais foram também cumpridos ao garantir que o sistema envie convites automáticos para todos os participantes, contendo os detalhes da reunião, data e hora, com integração perfeita ao calendário e sistema de comunicação da empresa.

Assim, foi criado com sucesso um fluxo de operações que possibilita o agendamento semanal de reuniões para todos os participantes de uma determinada entidade, garantindo a formação aleatória de grupos diferentes a cada semana.

Devido às limitações encontradas, na construção automática de automatizações, pela integração da inteligência artificial generativa com a tecnologia Microsoft Power Automate, procurou-se novas formas de explorar esta área na solução, através do conector AI Builder. Este conector oferece uma vasta gama de funcionalidades, como detecção de voz, tradução e resumo de textos e descrição de imagens, sendo um recurso promissor. Embora tenham surgido ideias tirando partido deste conector, como sugestão de temas e jogos durante as reuniões com base nos gostos e carateristicas dos participantes, estas não foram implemen-

tadas devido a restrições de tempo e à prioridade de outros projetos da empresa. No entanto, a exploração da aprendizagem automática e inteligência artificial continua alinhada com a estratégia de inovação da empresa, podendo ser retomada futuramente quando houver tempo e recursos disponíveis.

Foram realizados testes funcionais de modo a assegurar o cumprimento dos requisitos funcionais e não funcionais. Foram ainda realizados, testes unitários que verificaram o correto comportamento das funções individuais dentro do fluxo do Power Automate. A parte do fluxo que baralha os grupos de colaboradores e a parte do fluxo que adiciona, a cada iteração, uma semana à data de agendamento foram testada repetidamente e individualmente, garantindo que grupos diferentes fossem formados a cada semana. A automatização foi ainda avaliada com testes de desempenho recorrendo a diferentes números de participantes (20, 30, 50 e 100 participantes) e com grandes limites de calendário (3, 6 e 12 meses) para perceber se a solução tinha a capacidade de proceder ao correto agendamento, sendo o resultado destes positivo.

Os testes confirmaram que o sistema atendia aos requisitos, tanto funcionais quanto não funcionais. A interface foi avaliada como intuitiva, e a automatização dos agendamentos semanais ocorreu sem atrasos significativos, cumprindo os objetivos estabelecidos.

4.5 Sistema Dinâmico de Gestão Notificações e Tarefas

Atualmente está a ser desenvolvida, pela empresa, uma aplicação para introdução, investigação e resolução de problemas. Consequentemente, é requisitado um sistema, complementar à aplicação, capaz de automatizar a gestão de notificações e tarefas para essa aplicação. O sistema deve monitorizar as diferentes alterações que acontecem na aplicação e, com base nessas, enviar notificações aos destinatários, criar novas tarefas ou concluir automaticamente tarefas já existentes. Este deve ainda ser dinâmico, de modo ao cliente poder adicionar ou editar as suas notificações e tarefas.

4.5.1 Requisitos Atribuídos

Devido ao desempenho positivo na solução anterior, que envolveu a automatização do agendamento aleatório de reuniões internas, foi confiada ao estágio, a responsabilidade de desenvolver o sistema dinâmico de notificações e tarefas. Contou-se com o suporte no design por um profissional de desenvolvimento de software experiente e pelo arquiteto da empresa. Assim, os requisitos atribuídos foram:

Requisitos Funcionais

- Enviar notificações automáticas para os destinatários designados, com base nas ações realizadas na aplicação.
- Criar tarefas pendentes quando determinadas ações são realizadas na aplicação, com a possibilidade de definir prazos e responsáveis.
- Completar automaticamente as tarefas criadas quando as condições para a sua conclusão forem cumpridas.
- Fácil customização, permitindo que o cliente adicione ou modifique notificações e tarefas conforme necessário, sem exigir conhecimentos técnicos avançados.

Requisitos Não Funcionais

- Altamente responsiva, garantindo que as notificações e tarefas sejam geradas e enviadas em tempo real.
- Integrar-se perfeitamente com o ambiente do SharePoint e outras ferramentas do Microsoft 365 utilizadas pelo cliente.

- Deve ser escalável, suportando um aumento no número de utilizadores e na quantidade de dados monitorados sem perda de desempenho.
- Garantir a segurança dos dados, respeitando as políticas de privacidade e controlor de acesso estabelecidas pelo cliente.

4.5.2 Abordagem para o Desenvolvimento

A solução proposta visa automatizar a gestão de tarefas e notificações para uma nova aplicação em desenvolvimento pela empresa. Esta aplicação será focada na introdução de investigações, problemas e as suas respectivas resoluções. A contribuição do estágio incide-se exclusivamente no sistema de notificações e gestão de tarefas, que complementa a aplicação principal.

O sistema visa ser desenvolvida utilizando a tecnologia de fluxos de automatização, Microsoft Power Automate. Através da qual irão ser capturadas as ações realizadas na aplicação de gestão de problemas, monitorizando as alterações na base de dados elaborada através de listas SharePoint. Esta monitorização deverá iniciar-se em intervalos de tempo contínuos, para que as alterações na base de dados sejam capturadas e processadas sucessivamente. Os diferentes eventos capturados e as suas definições foram fornecidos pelo cliente.

O sistema deve ter a sua interface utilizando também listas Sharepoint e tem de ser intuitivo o suficiente para que o cliente, já experiente nas tecnologias Microsoft 365, consiga adicionar, remover ou editar notificações, tarefas ou como ficam concluídas.

Esta solução garante uma gestão eficaz das tarefas, diminuindo o risco de erros na execução das mesmas, ao suprimir a necessidade de uma contínua monitorização manual do sistema.

4.5.3 Soluções Implementadas e Desafios Encontrados

De modo a desenvolver esta solução começou-se por elaborar o *trigger* da automatização em Power Automate. Colocou-se um *trigger* recorrente em intervalos de um minuto, para a contínua captura de eventos.

Consideram-se um evento toda e qualquer ação na aplicação que necessite da criação de uma notificação, uma tarefa ou a conclusão desta. As ações realizadas na aplicação traduzem-se em mudanças de valores na base de dados em Sharepoint, as quais o nosso sistema terá de verificar. Dito isto, elaborou-se alguns testes para o desenvolvimento da parte do sistema que captura eventos.

Captura de eventos

Primeiro tem de se calcular a partir de que momento tem de se verificar a base de dados, já que não queremos verificar mudanças antigas na base de dados que já capturámos anteriormente. Para isso necessita-se do horário atual e do intervalo de tempo que devemos verificar, este deve ser igual ao intervalo de tempo entre automatizações que verificam os eventos.

De seguida recuperou-se todos os itens de uma dada lista Sharepoint, com modificações posteriores ao momento calculado e armazenado anteriormente. Isto realizou-se através da operação "Get items" do conetor Sharepoint, que consta na Figura 4.28. Nesta colocou-se o site Sharepoint e a lista que queremos recuperar os itens. De modo a filtrar apenas os itens que fossem modificado, apenas dentro daquele intervalo de tempo utilizou-se a sua opção de "Filter Query". Esta permite um filtro na consulta à lista Sharepoint, neste caso colocou-se a expressão "Modified ge 'outputs('SinceToCheck')' and Modified lt 'outputs('UntilToCheck')'". Assim filtra-se apenas os itens cujas datas de modificação estão dentro do intervalo definido. A expressão indica que apenas se pretende os itens cujo o campo "Modified" for maior, com o operador "ge" (*greater* or *equal*, em português maior ou igual), que o momento calculado ("SinceToCheck") e menor, com o operador "lt" (less than, em português menor que) o momento do *trigger*, já que as operações levam algum tempo a serem executadas não se pode utilizar o tempo atual. Todos os itens criados nas listas Sharepoint têm campos obrigatórios e preenchidos automaticamente pelo sistema, como os campos: "Title", "ID", "Created", e os utilizados neste caso "Modified", "CreatedBy" e "ModifiedBy". Quando um item novo é criado no SharePoint, o campo "Modified" é automaticamente preenchido com a data e hora da criação do item e, nesta instância irá ter o mesmo valor que o campo "Created". O mesmo acontece para os campos "ModifiedBy" e "CreatedBy", que irão adotar ambos o utilizador que cria o item. Assim apenas é necessário examinar os campos de modificação mesmo que o item tenha sido criado.

Porém, no intervalo de tempo determinado, podem acontecer mais do que uma modificação nos campos devolvidos. Dito isto, para cada item que anteriormente se verificou que foi modificado, é necessário verificar todos os valores atribuídos aos campos durante esse período. Para isso recorreu-se à propriedade de histórico de versões do Sharepoint. Esta propriedade é um recurso que permite rastrear e gerir alterações feitas em documentos ou itens de listas Sharepoint, ao longo do tempo. Cada vez que uma alteração na lista é salva, uma nova versão do documento ou item modificado é criada, mantendo o registo das edições anteriores. Como não existe uma operação, do conector Sharepoint, que recupera diretamente as diferentes versões de um item, utilizou-se a operação "Send an HTTP request to Sharepoint", que

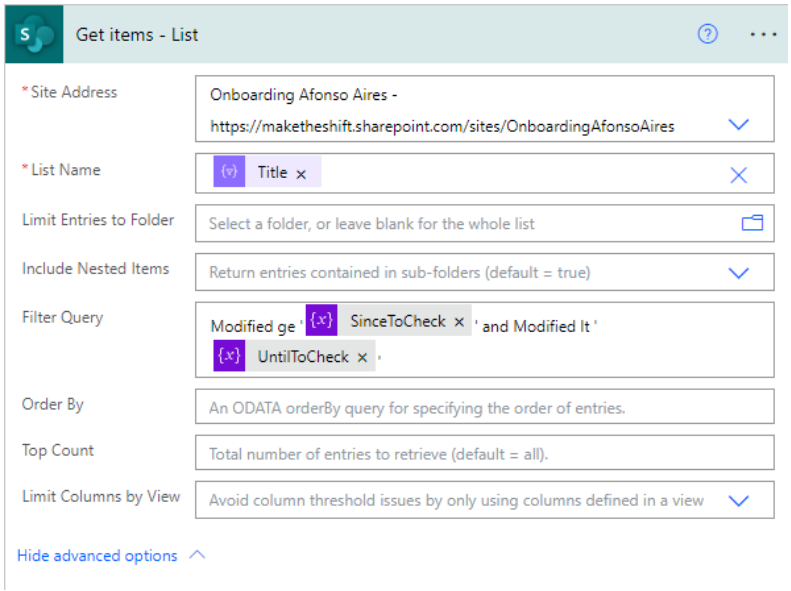


Figura 4.28: Operação 'Get items' com filtro temporal

permite enviar solicitações HTTP diretamente à API RESTful do SharePoint (esta operação está presente na Figura 4.29). Adicionou-se a esta operação, o método HTTP pela qual queremos fazer o pedido à API, neste caso GET, e o URL do endpoint que queremos aceder, neste caso "_api/Web/lists/getByTitle('outputs('ListTitle')')/items(outputs('ItemID'))/versions". A primeira parte obtém os itens da lista SharePoint com o título fornecido pela variável dinâmica "ListTitle", processo análogo ao que acontece na operação anteriormente explicada "Get items". A segunda parte especifica o item dentro da lista através do seu ID. Finalmente, o sufixo "/versions" acede o histórico de versões do item, permitindo ver todas as versões do item. Adicionou-se ainda um filtro ao sufixo "'/versions'", o seguinte parâmetro: "/versions?$sort=Modified desc". Esta consulta diz à API para ordenar os resultados pela coluna Modified em ordem decrescente. De seguida adiciona-se o mesmo filtro anteriormente referido nas diversas versões do item de modo a apenas recolher as versões dentro do intervalo de tempo pretendido. Para isso utilizou-se a operação "Filter array" que se observa na Figura 4.29.

Após recuperar todas as versões dos itens modificados dentro do intervalo de tempo pretendido, ainda não são conhecidos os campos que foram concretamente alterados. Para isso utilizou-se a operação "Get changes for an item or a file" do conector Sharepoint. Neste, é introduzido o website Sharepoint que queremos pesquisar, a lista, o id do item e o momento a partir do qual queremos fazer a pesquisa. Assim é devolvido uma lista com os campos modificados, para cada versão do item pedido, dentro de um intervalo de tempo específico.

Fica-se assim com uma estrutura de dados com todos as versões dos itens que foram

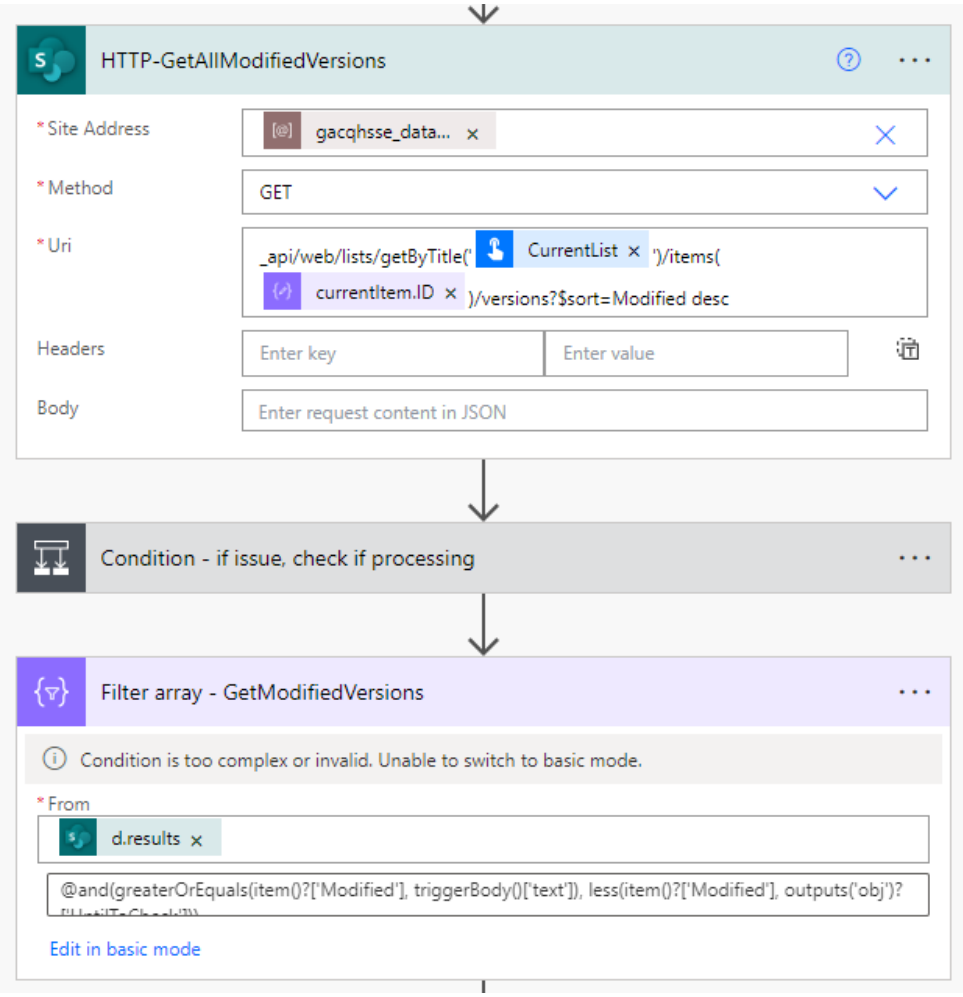


Figura 4.29: Operação 'Send an HTTP request to Sharepoint' com o objetivo de recuperar todas as versões de um item por ordem temporal decrescente seguida de operação 'Filter array' de modo a filtrar temporalmente o que é recuperado do pedido HTTP

modificados num intervalo de tempo específico, e quais os campos alterados em cada versão.

Trigger dos eventos

De seguida tem de se configurar o sistema para saber quais as listas e os campos que devem ter eventos associados. Para além disso, também tem de se perceber se determinadas modificações nesse campo despertam ou não algum deste tipo de notificação, tarefa ou conclusão desta. Para isto criou-se uma interface para personalização dos *triggers*, notificações e tarefas através de listas Sharepoint.

De modo a conceber esta interface, criou-se uma lista Sharepoint, como se de uma tabela se tratasse, denominada "Events", em que os seus itens serão todos os eventos necessários serem verificados e os campos (colunas) dessa lista, as definições necessárias à sua captura. Estes eventos podem ser do tipo notificação ou tarefa.

Esta lista irá conter, para cada evento, as seguintes cinco definições: A lista (coluna “ListToCheckChanges”, do tipo *Choice*) e o campo (coluna “FieldToCheckChanges”, também do tipo *Choice*) pela qual a pesquisa de eventos tem de ser realizada. A definição seguinte (coluna “CheckBeforeVsAfter”) é um valor binário que indica se a alteração do valor do campo interessa (por exemplo, se o campo X passa de A para B). Logicamente as definições seguintes são os valores dessa alteração, o valor anterior (coluna “OldValue”) e o valor seguinte (coluna “NewValue”), em texto.

De seguida, adaptou-se o fluxo para atender a estas definições. Ou seja, agrupa-se todas as diferentes listas presentes na coluna “ListToCheckChanges” de todos os eventos da lista “Events”. Obtêm-se todas as versões dos itens que sofreram alterações, no período permitido. Verifica-se se as modificações em cada item, foram realizadas nos respetivos campos (“FieldToCheckChanges”), contidos na lista “Events”. De seguida, verifica-se se o valor binário “CheckBeforeVsAfter” está ativo. Se não estiver ativo todas as versões dos itens que verificarem as condições referidas despertam um evento. Se estiver ativo, verifica-se em cada versão consecutivas do mesmo item, se houve a mudança do valor “OldValue” para o valor “NewValue”. Se o valor “OldValue” estiver vazio, a condição para despertar o *trigger* apenas considera o valor atual do item. Este algoritmo pode ser verificado em detalhe na Figura 4.30.

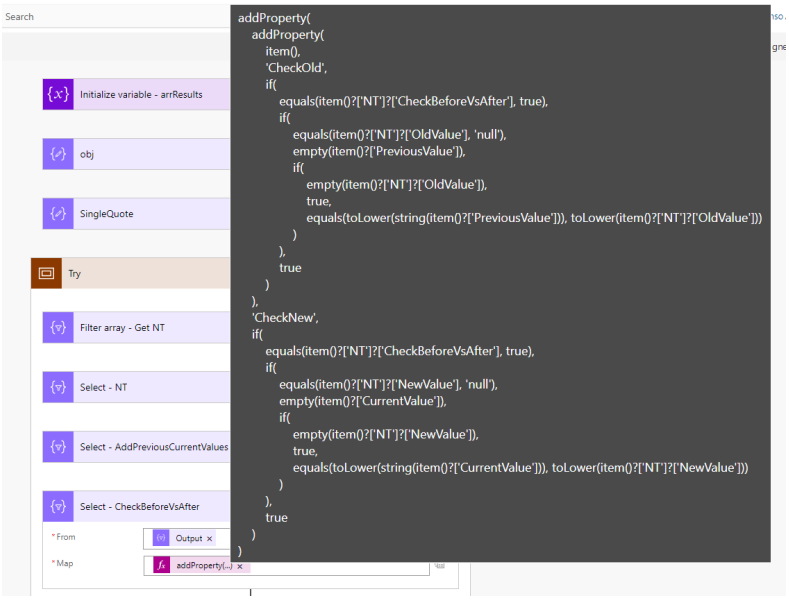


Figura 4.30: Algoritmo de verificação de versões consecutivas em Power Fx

Alguns eventos têm condições mais complexas do que apenas a mudança de um valor num campo de uma lista. Há eventos que tem de se ter em consideração a mudança de um valor em vários campos de diferentes listas. Para isso foi criado mais uma coluna denominada “TriggerCondition”. Esta coluna é do tipo texto e definirá as condições adicionais para os

eventos, estas estarão sempre relacionada com o *trigger* originalmente concebido e explicado anteriormente. Diversas condições podem ser inseridas nesta coluna, separando-se por ponto e vírgula. Cada condição tem seis elementos separados por hífen. O primeiro elemento é a lista que queremos verificar. O segundo e o terceiro são os campos pela qual essa lista a analisar e a lista do *trigger* original se relacionam, sendo o segundo, a chave primária da lista a analisar e o terceiro, o campo da lista do *trigger* original que se relaciona com a lista a analisar. O quarto elemento é o operador de comparação. O quinto e último elemento é o valor que o campo da lista da atual a ser comparado.

A lista "Events" com as colunas descritas acima encontra-se na Figura 4.31.

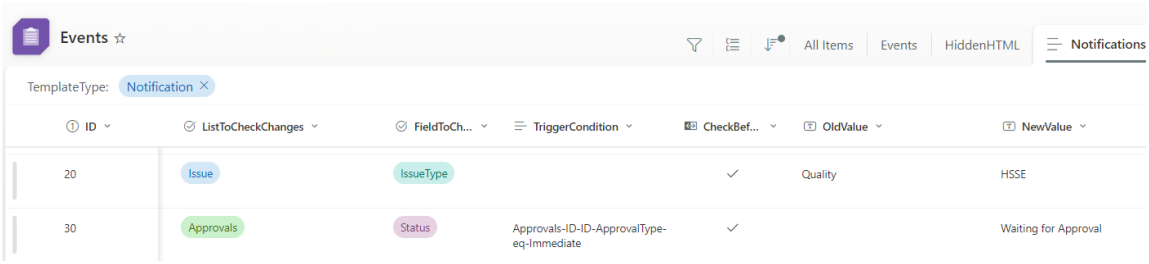


Figura 4.31: Lista 'Events' do sistema de notificações e tarefas, parcialmente completa

De seguida, um exemplo prático para se perceber melhor o fluxo até ao momento. Ou seja, um evento apresenta as seguintes definições de *trigger*:

- "ListToCheckChanges" - *Actions*
- "FieldToCheckChanges" - *Phase*
- "CheckBeforeVsAfter" - *True*
- "OldValue" - *Waiting for Execution*
- "NewValue" - *Waiting to be sent for approval*
- "TriggerCondition" - *Issue-Title-Reference/LookUpValue-IssueType/LookUpValue-eq-Quality*

O sistema irá obter todas as versões de todos os itens modificados, dentro do intervalo de tempo definido, presentes na lista Actions (ou todas as listas que estiverem referenciadas na coluna "ListToCheckChanges" da lista "Events"). Para cada um, irá obter quais os campos modificados e verificar se algum coincide com o campo presente na coluna "FieldToCheckChanges" do evento correspondente, neste caso *Phase*. Para cada versão do item que verificar essa condição, o sistema irá confirmar qual o valor binário da coluna "CheckBeforeVsAfter".

Se for verdadeiro, como é o caso, o sistema certifica-se que o valor atual dessa versão do item é igual ao valor que se apresenta na coluna “NewValue” (no exemplo *Waiting to be sent for approval*) e se o valor da versão anterior é igual ao valor da coluna “OldValue” (no exemplo Waiting for Execution). Se a coluna “OldValue” estiver vazia, significa que o valor anterior não é importante para o evento e o sistema apenas verifica o valor da coluna “NewValue”. Se a coluna “CheckBeforeVsAfter”, do evento associado ao item que está a ser verificado, tiver o valor de false, o sistema desperta o evento sem a necessidade das verificações do valor atual e anterior.

Se todas as condições anteriores tiverem validadas, o sistema começa a examinar as condições seguintes na coluna “TriggerCondition”, senão termina. De modo a verificar as condições seguintes o sistema divide o texto pelo caráter hífen e itera por cada elemento da condição. No caso do exemplo, o sistema procura na lista *Issue*, e obtém o item cujo o *Title* é igual ao *Reference/LookUpValue* do item que deu origem ao evento, através da operação “Get Items” e do filtro temporal explicado anteriormente. Depois de obter esse item da lista *Issue*, verifica se o seu campo *IssueType/LookUpValue* é igual ao texto “Quality”. Isto acontece para todas as condições, separadas por ponto e vírgula, na coluna “TriggerCondition” da lista “Events”.

Se todos as condições foram completadas, o sistema deve perceber se o evento despertado é uma notificação ou uma tarefa. Para isso criou-se uma nova coluna na lista “Events” denominada “Type”, que pode tomar dois valores: “Notification”, no caso de despertar uma notificação ou “Event” no caso de ser um evento destinado a uma tarefa.

Enviar notificações

Se a coluna “Type” da lista “Events” contiver o valor “Notification”, criaram-se as seguintes colunas, que devem ser preenchidas:

- “Subject” - O texto do assunto da notificação a enviar via e-mail
- “Body” - O texto do corpo da mensagem da notificação a enviar via e-mail
- “TagNotified” - Uma *tag* representativa das pessoas ao qual a notificação, deve ser enviada via e-mail
- “CC” - Uma *tag* representativa das pessoas ao qual a notificação, deve ser enviada com conhecimento via e-mail

O assunto e corpo do e-mail são idênticos na sua composição, devem conter ambos o texto que se pretende enviar com todas as informações dinâmicas como *tags*. Essas *tags* devem ser representadas entre parentese retos de modo ao sistema identificar e substituir.

Para o mecanismo de substituição de *tags*, foi criada uma outra lista, denominada "NotificationTagsMaping". Esta contém as colunas "TagName" que identifca a *tag* com uma palavra ou texto. "TargetListName" que representa a lista que detém o conteúdo da *tag*. "FilterQuery" que representa a consulta que o sistema deve fazer para encontrar a *tag* na lista, esta deve tomar o mesmo formato de uma consulta através da operação "Get Items", explicada previamente. "TargetFiledName" que representa a coluna da lista onde está a informação a ser substituída pela *tag*. "FieldType" que identifica o tipo de coluna de onde se retira a informação, pode ter o valor de "string" se for uma *tag* normal, "date" se for uma data ou *array* se for uma tabela. "ConvertFromUTCToTimezone", uma coluna que apenas é necessário preencher se a coluna "FieldType" for do tipo "date". "FormatDate", novamente apenas é preciso preencher se a coluna "FieldType" for do tipo "date". Esta lista, com algumas das suas colunas, consta na Figura 4.32.

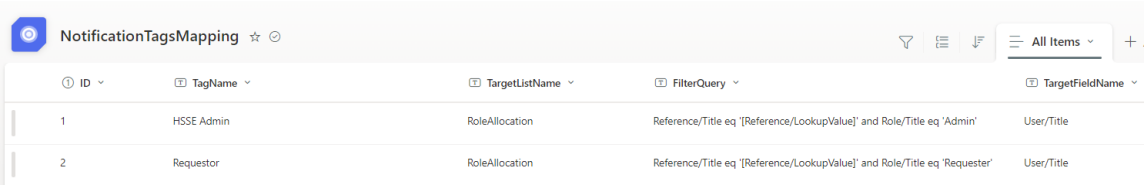


Figura 4.32: Lista 'NotificationTagsMaping' parcialmente completa, do sistema de notificações e tarefas

Assim, para o sistema de substituição de *tags*, verificou-se qual o nome da *tag* a substituir. A seguir, tenta-se obter o item ou itens, segundo a consulta da lista que lhe está associada com o filtro correspondente, da mesma forma que foi realizado para os *triggers* e adquire-se o valor que consta na coluna indicada. Se a *tag* em questão tiver associada, na coluna "FieldType" o valor de "string", verifica-se se apenas um item foi retornado, se for o caso termina-se ao substituir a *tag* no respetivo local, se foram vários devolvidos são separados os valores com ponto e vírgula. Se a *tag* tiver associada ao valor de "date", significa que se trata de uma data e converte-se, do fuso horário por omissão (UTC), para o fuso horário que consta na coluna "ConvertFromUTCToTimezone", com o formato que consta na coluna "FormatDate", isto utilizando a função "convertTimeZone('<data>', '<fuso-horário-original>', '<fuso-horário-destino>', '<formato>')". Se a *tag* tiver associada ao valor de *array*, é necessário formatar essa informação em tabela HTML, de modo a ser renderizada no corpo da mensagem do e-mail a enviar.

Após o sistema substituir todas as *tags* necessárias (no assunto e corpo do e-mail, nas pessoas a enviar o e-mail diretamente ou com conhecimento), invoca-se a operação “Send an e-mail” do conector Office 365 Outlook para cada destinatário.

Criar e concluir tarefas

De modo a criar e concluir tarefas, foi elaborada uma nova lista “NotificationTemplates-Tasks”, nome alusivo ao ficheiro de *templates* de notificações fornecido pelo cliente. É através desta lista, que se irá verificar os eventos para a criação de cada tarefa e os eventos para a sua conclusão. Esta lista pode ser verificada na Figura 4.33.

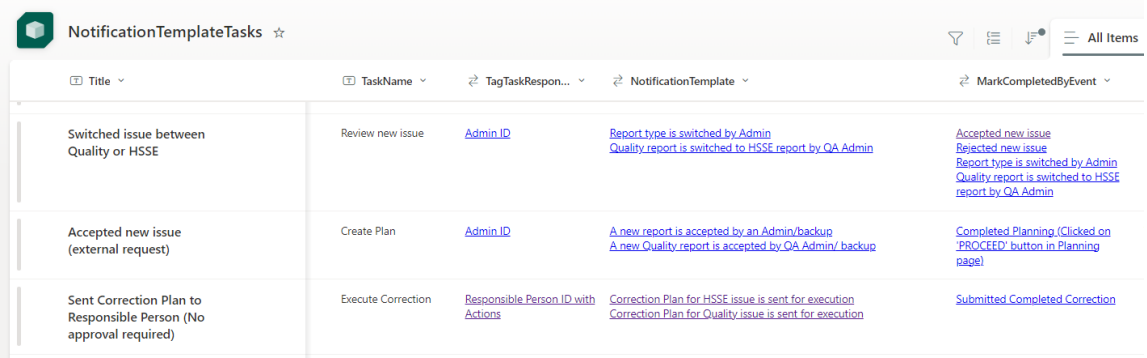


Figura 4.33: Lista ’NotificationTemplatesTasks’ do sistema de notificações e tarefas, parcialmente completa

Nesta lista, foram necessárias colunas como “TaskName”, com um texto representante do nome da tarefa. “TagTaskResponsible”, uma coluna *LookUp* para a lista de *tags*, que deve ser representativa da pessoa responsável pela tarefa. “Event”, uma coluna *LookUp* para a lista “Events”, com o evento responsável pela criação da tarefa. E “MarkCompletedByEvent”, outra coluna *LookUp* para a lista “Events”, desta vez para o evento atribuído à conclusão da tarefa. Todas estas colunas admitem várias referências.

De forma ao sistema criar tarefas de acordo com os eventos introduzidos, sempre que um evento da lista “Events” é despertado, confirma-se na lista “NotificationTemplatesTasks” se o mesmo é responsável por criar alguma tarefa. Se for o caso, é criado um novo item na tabela “Taks”, com o nome da tarefa e com e-mail da pessoa responsável pela sua conclusão, dessa maneira, os colegas responsáveis pela aplicação podem introduzi-la nas tarefas pendentes do utilizador.

A conclusão de uma tarefa é realizada de forma semelhante. Sempre que um evento é despertado é examinado a lista “NotificationTemplatesTasks”, por forma a conferir se este completa alguma tarefa. Se for o caso, encontra-se o utilizador que despertou o evento, atra-

vés do campo “ModifiedBy”, de modo a conferir que utilizador realizou a criação ou modificação do item. Percebe-se qual o evento que tem de ser completo pelo seu ID e, com a operação “Upate item” do conector Sharepoint, modifica-se o valor da coluna “Status” na tabela “Taks”, para “Completed”. Esta operação pode ser verificada na Figura 4.34.

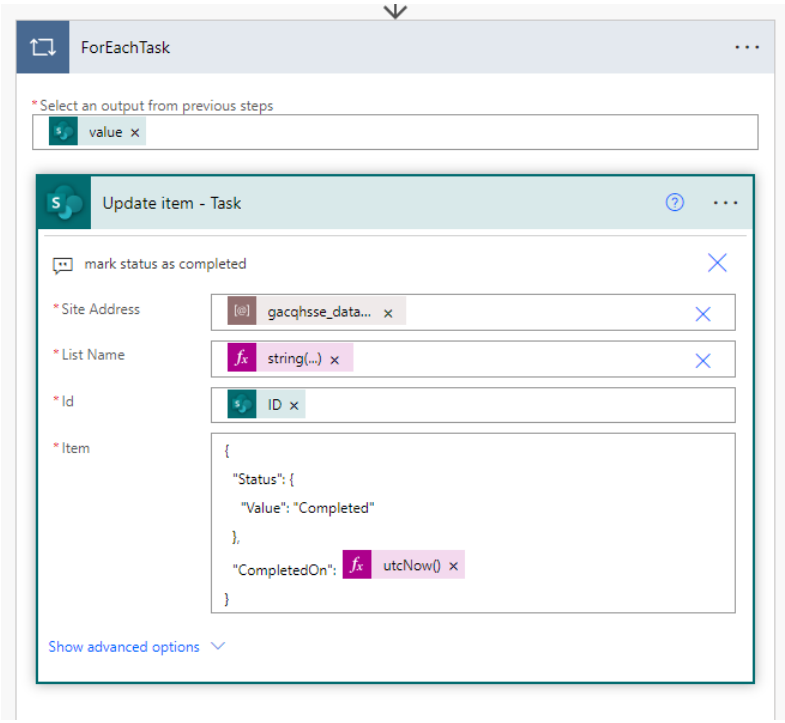


Figura 4.34: Algoritmo de verificação de versões consecutivas em Power Fx

No final do projeto, decidiu-se dividir a automatização em várias partes distintas, resultando numa solução fragmentada em múltiplas automatizações. Assim, a captura de eventos, o envio de notificações e a criação e conclusão de tarefas foram separados em fluxos diferentes. Esta abordagem não só melhora a organização do sistema, como também aumenta a sua velocidade de processamento, uma vez que a captura de eventos não tem de esperar que o sistema conclua o processamento para capturar os eventos seguintes. Isto faz com que o sistema seja concorrente e por sua vez mais eficiente, na captura de eventos, envio de notificações, criação de tarefas e conclusão das mesmas.

4.5.4 Avaliação e Resultados

A solução foi constantemente entregue dentro dos prazos estipulados nos sprints. Para o seu desenvolvimento foi utilizada a tecnologia Power Automate, que oferece uma robusta integração com o SharePoint, garantindo a segurança dos dados e respeitando as políticas de privacidade e controlor de acesso estabelecidas pelo cliente.

Realizou-se assim um sistema que cumpre os três requisitos principais do projeto. O envio de notificações e todas as informações presentes nos *templates* fornecidos pelo cliente. A criação de tarefas atribuídas a um ou vários responsáveis com as datas e prazos necessários. E a conclusão automática de tarefas na aplicação.

Todos esses requisitos foram atendidos com base no conceito de evento, definido como qualquer alteração nos dados da aplicação, em listas SharePoint. Cada evento é caracterizado por um *trigger* principal, ao qual podem ser adicionadas várias condições. Os eventos são configuráveis de forma intuitiva no sistema, permitindo uma flexível personalização. Como o cliente já está familiarizado com as tecnologias Microsoft 365, especialmente SharePoint, este pode adicionar ou modificar eventos conforme necessário, visando assim um sistema dinâmico. O sistema de tarefas e a conclusão destas foram implementados utilizando a mesma base de eventos. Foi ainda desenvolvido um sistema de substituição de *tags*, que pode ser aplicado tanto nos destinatários quanto no corpo das notificações.

De forma a garantir uma solução escalável e responsiva, capaz de suportar um aumento no número de utilizadores e na quantidade de dados monitorizados, o fluxo original foi fragmentado em vários fluxos mais pequenos. Esta abordagem permite uma clara separação entre os fluxos que capturam e processam eventos para determinar se devem gerar notificações ou tarefas, e aqueles responsáveis pela criação, envio e registo dessas notificações, tarefas ou conclusão. Através desta divisão assegura-se a geração e envio das notificações e tarefas em tempo real.

Para garantir que a solução funcionasse corretamente e atendesse aos requisitos do cliente, foram realizados testes unitários e funcionais durante o desenvolvimento.

Os testes unitários focaram-se em validar individualmente as funções de cada fluxo criado no Power Automate. Cada um dos fluxos que capturam eventos no SharePoint, geram notificações, criam tarefas e as concluem-nas automaticamente, foi testado de forma isolada para assegurar que os dados eram processados corretamente e os *triggers* de evento eram acionados de acordo com as regras configuradas. Adicionalmente, o sistema de substituição de *tags* foi testado para garantir que as notificações eram enviadas com as informações corretas e personalizadas, evitando erros no conteúdo e na atribuição dos destinatários.

Os testes funcionais e de desempenho foram aplicados ao sistema como um todo, simulando o uso real por parte do cliente. Através destes testes, simularam-se cenários com uma grande quantidade de eventos e dados monitorizados. Os resultados foram de qualidade com as notificações e tarefas geradas em tempo real sem atrasos significativos, com os responsáveis adequados, e com a garantia que o sistema continua a operar de forma estável.

Capítulo 5

Conclusões e trabalho futuro

Concluíram-se com sucesso todos os projetos realizados no âmbito do estágio. Estes tiveram influência direta para o meu desenvolvimento profissional e contribuíram ativamente para a evolução da empresa.

No projeto "Aplicação de Gestão de Cartões de Negócio Digitais - Cartões Horizontais", recriaram-se de forma precisa, os vários *templates* de cartões de negócios do cliente, através das tecnologias Web HTML e CSS. Para a apresentação dos cartões digitais foi desenvolvido, tanto o back-end como do front end, de uma página Web, suportada pela plataforma Azure Web Apps e seguindo o padrão de desenvolvimento MVC. No back-end foi garantida a segurança de dados sensíveis, através dos serviços de gestão segura de chaves e segredos da Microsoft, Azure Key Vault. Adicionalmente, foi implementada um mecanismo para ocultar informações consoante a sua disponibilidade na base de dados e espaço disponível no ecrã, através de um sistema de *tags*. Estes cartões tiveram ainda de ser adaptados às limitações dos controladores da tecnologia Power Apps, de modo a serem mostrados, consistentemente, tanto na aplicação móvel como na página Web. Para que o cliente pudesse descarregar os cartões digitais, foi aplicado o conceito de vCards, o formato padrão para cartões de visita. Ainda neste projeto, foi utilizado uma função Azure que, consoante um parâmetro concebido no URL do seu pedido HTTP, gera uma imagem com um código QR, oferecendo um modo de partilha ágil e prática dos cartões digitais. A integração na solução *low-code*, das tecnologias HTML e CSS para digitalizar os cartões de negócio, simultaneamente com as tecnologias da Microsoft Azure, tanto para o desenvolvimento de uma página Web utilizando a plataforma .NET, como das funções Azure, tornou-se uma abordagem eficiente e fundamental para entregar uma solução que cumpre todos os objetivos iniciais.

A cópia da aplicação anterior para a demonstração desta para um novo cliente, levou ao design de um novo cartão de negócios digital, seguindo as melhores práticas e normas do

ramo da interação pessoa-máquina. Na fase de implementação, foram aplicada as tecnologias Web HTML e CSS de forma a oferecer à empresa um *template* de cartão digital com uma interface moderna e funcional.

Durante o desenvolvimento do projeto "Aplicação de Gestão de Auditorias" integrou-se as tecnologias Web tradicionais, HTML e CSS, com a tecnologia Power Automate, permitindo a criação de um relatório dinâmico que complementa a aplicação. A implementação deste projeto foi a menos direta resultando em alguns problemas com o resultado final da solução. Dito isto, várias abordagens foram testadas até se encontrar a que melhor atendia aos objetivos definidos. Com este projeto conclui-se que é essencial testar de forma rigorosa todas as ferramentes que visam fazer parte da solução, antes de proceder ao seu desenvolvimento, para que não haja esforços desnecessários ou inesperados. Em relação à tecnologia Power Automate, conclui-se que existem vários conectores de origem mas que por vezes podem ter limitações em termos de opções e configurações. No entanto, a utilização de conectores premium pode proporcionar uma experiência mais rica e completa, embora essas funcionalidades adicionais sejam pagas.

Ao desenvolver o projeto "Automatização de Agendamento de Reuniões Internas" foi implementada uma solução que proporcionou à empresa uma forma eficiente e intuitiva de agendar reuniões periódicas, utilizando a tecnologia Power Automate. Esta automatização reduziu significativamente o tempo gasto com o antigo agendamento manual e garantiu maior consistência na gestão de horários, maximizando a organização e a comunicação interna. Durante implementação desta solução houve a oportunidade de investigar sobre a integração de inteligência artificial generativa para a composição do fluxo do agendamento periódico. Conclui-se que, embora promissora, essa integração ainda está em uma fase inicial e apresenta várias limitações. Para utilizadores que têm experiência com a tecnologia, é mais eficaz realizar o fluxo com o seu próprio raciocínio e lógica, porém para aqueles que estão a explorar a tecnologia, esta integração oferece dicas úteis que podem ajudar na introdução da tecnologia. Para além disso, a tecnologia Power Automate oferece conectores de inteligência artificial com funcionalidades bastante abrangentes, que vão desde o resumo e tradução de textos até ao treino de modelos através de redes neuronais, deixando a oportunidade, para os mais criativos, de explorar as diferentes possibilidades.

Na solução que implementa o "Sistema Dinâmico de Notificações e Tarefas" foi utilizada a API da plataforma Sharepoint para verificar versões antigas de itens e as alterações nos seus diversos campos, através do histórico disponível. Para além disto foi realizada uma interface, utilizando listas Sharepoint, de forma a tornar o sistema dinâmico com a possibilidade de

adicionar, editar ou remover notificações, tarefas e os seus *triggers*. Esta abordagem permitiu à empresa oferecer um sistema de notificações e gestão de tarefas totalmente novo, dinâmico e de fácil personalização, ampliando seu portfólio de soluções e aumentando seu potencial de vendas.

Conclui-se, assim que, embora as plataformas *low-code* sejam práticas e eficientes, apresentam algumas limitações. Estas limitações podem ser superadas pela integração de tecnologias tradicionais, o que permite expandir as suas funcionalidades e a flexibilidade das soluções desenvolvidas.

A conclusão deste trabalho não marca o fim, mas sim o início de novas oportunidades, que incluem a exploração da ferramente para análise de dados da Power Platform, Power BI. Integrá-la com as soluções já desenvolvidas, permite criar relatórios e painéis de controlo interativos, otimizando a visualização de dados e métricas empresariais. Para além disso irá incidir-se na exploração de várias formas de integração de Inteligência Artificial nas soluções da empresa por forma a desenvolver soluções de maneira mais eficiente, oferecendo aos clientes uma melhor experiência e personalização, o que permitirá à empresa um bom posicionando no mercado das soluções digitais.

Bibliografia

[1] J. Bratincevic, R. Taylor e Z. Stone. *The Low-Code Market Could Approach $50 Billion By 2028*. Último acesso em 5 de setembro de 2024. 2024. URL: https://www.forrester.com/blogs/the-low-code-market-could-approach-50-billion-by-2028/.

[2] M. Forsyth. *The latest low-code trends are heating up the market in 2024*. Último acesso em 5 de setembro de 2024. 2024. URL: https://www.outsystems.com/blog/posts/low-code-market/.

[3] Microsoft. *Microsoft Learn*. Último acesso em 20 de setembro de 2024. 2024. URL: https://learn.microsoft.com/en-us/.

[4] Jennifer Niederst Robbins. *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics*. 4ª ed. O'Reilly Media, 2012. URL: https://books.google.pt/books?hl=pt-PT&lr=&id=FJkVxtXr7n0C&oi=fnd&pg=PR11&dq=HTML+CSS+and+javascript&ots=lQCQRYfD2A&sig=KtekskDRKS26y6yAgJ6JsAv8UfI&redir_esc=y#v=onepage&q=HTML%20CSS%20and%20javascript&f=false.

[5] G. A. Boy. *The Handbook of Human-Machine Interaction: A Human-Centered Design Approach*. Ashgate Publishing, 2011. URL: https://books.google.pt/books?hl=pt-PT&lr=&id=t2kQEAAAQBAJ&oi=fnd&pg=PP1&dq=human-machine+interaction&ots=xrcaS5WBHZ&sig=85Z8DxaCNeWvOng83wMJHRUiiMQ&redir_esc=y#v=twopage&q&f=false.

[6] Bill Scott e Theresa Neil. "Designing Web Interfaces". Em: *Designing Web Interfaces: Principles and Patterns for Rich Interactions*. Último acesso em 22 de dezembro de 2024. 2008.

[7] *HiHello*. - Aplicação móvel. Último acesso em 17 de setembro de 2024. URL: https://play.google.com/store/apps/details?id=play.me.hihello.app&hl=en_US.

[8] *KADO*. - Aplicação móvel. Último acesso em 17 de setembro de 2024. URL: https://play.google.com/store/apps/details?id=com.kadonetworks.app.

[9] *Popl*. - Aplicação móvel. Último acesso em 17 de setembro de 2024. URL: https://play.google.com/store/apps/details?id=com.nfc.popl.

[10] *HiHello - Business Card Design: Ideas, templates to Make a Digital Business Card*. - Website. Último acesso em 17 de setembro de 2024. URL: https://www.hihello.com/blog/digital-business-card-design-ideas-templates.

[11] *Popl - Incident Report*. - Aplicação móvel. Último acesso em 17 de setembro de 2024. URL: https://play.google.com/store/apps/details?id=com.workmetrics.newir.

[12] *Popl - Report Generator*. - Aplicação móvel. Último acesso em 17 de setembro de 2024. URL: https://play.google.com/store/apps/details?id=com.Jacobs.TrackRecord.ReportGenerator.

[13] *Calendly*. - Aplicação móvel. Último acesso em 17 de setembro de 2024. URL: https://calendly.com/?gad_source=1&gclid=EAIaIQobChMIqdCGqIfBiAMVRK-DBx2bGSNLEAAYASAAEgIbkfD_BwE.

[14] *Task Bird*. - Website. Último acesso em 17 de setembro de 2024. URL: https://www.manageengine.com/products/service-desk-msp/help/adminguide/configurations/helpdesk/custom-triggers-for-notifications.html.

[15] *ManageEngine ServiceDesk Plus MSP - Custom Triggers for Notifications*. - Website. Último acesso em 17 de setembro de 2024. URL: https://www.manageengine.com/products/service-desk-msp/help/adminguide/configurations/helpdesk/custom-triggers-for-notifications.html.

[16] Roger Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2003. URL: https://books.google.pt/books?hl=pt-PT&lr=&id=bL7QZHtWvaUC&oi=fnd&pg=PA1&dq=Roger+Pressman.+Software+Engineering:+A+Practitioner%E2%80%99s+Approach.+McGraw-Hill,+2003.&ots=O9veaPuN2g&sig=LHer2KSu-W_4Oavb9NwztrlxwGM&redir_esc=y#v=onepage&q&f=false.

[17] S. Panda. *SDLC (Software Development Life Cycle) Phases, Process. What is SDLC*. - Website. Último acesso em 17 de setembro de 2024. 2023. URL: https://www.numpyninja.com/post/sdlc-software-development-life-cycle-phases-process-what-is-sdlc.

[18] P. Pandit e S. Tahiliani. *AgileUAT: A Framework for User Acceptance Testing based on User Stories and Acceptance Criteria*. 2015. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=880d436d70fc42f8972d0f8d34a055ae59a074ee.

[19] R. Cimperman. *UAT Defined: A Guide to Practical User Acceptance Testing*. Addison-Wesley, 2007. URL: https://books.google.pt/books?hl=pt-PT&lr=&id=spr3965oVlkC&oi=fnd&pg=PP8&dq=What+is+User+Acceptance+Testing+(UAT)&ots=UWkgN8BL9f&sig=WqCdcwmJq-TQelNWFqSnjzx8FJ4&redir_esc=y#v=onepage&q&f=false.

[20] Alexander S. Gillis. *What is User Acceptance Testing (UAT)?* Último acesso em 17 de setembro de 2024. 2022. URL: https://www.techtarget.com/searchsoftwarequality/definition/user-acceptance-testing-UAT.

[21] Shirley Radack. *The System Development Life Cycle (SDLC)*. Último acesso em 17 de setembro de 2024. 2009. URL: https://csrc.nist.gov/csrc/media/publications/shared/documents/itl-bulletin/itlbul2009-04.pdf.

[22] Luís Morgado. *Engenharia de Software - Processos de Desenvolvimento*. Moodle 2022/2023. Instituto Superior de Engenharia de Lisboa, 2010.

[23] V. Szalvay. *An Introduction to Agile Software Development*. Danube Technologies, Inc., 2004. URL: https://profs.info.uaic.ro/adrian.iftene/Licenta/Documentatie/Intro_to_Agile.pdf.

[24] A. Leff e J. T. Rayfield. "Web-application development using the Model/View/Controller design pattern". Em: *Proceedings of the Fifth IEEE International Enterprise Distributed Object Computing Conference*. Último acesso em 17 de setembro de 2024. 2001. URL: https://ieeexplore.ieee.org/abstract/document/950428.

[25] Microsoft Learn. *Visão geral do ASP.NET Core MVC*. - Website. Último acesso em 17 de setembro de 2024. 2023. URL: https://learn.microsoft.com/pt-pt/aspnet/core/mvc/overview?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-Website.

[26] Microsoft Learn. *Deploy a Website to Azure with Azure App Service*. - Website. Último acesso em 17 de setembro de 2024. 2024. URL: https://learn.microsoft.com/en-us/azure/key-vault/general/tutorial-net-create-vault-azure-Web-app?tabs=azure-cli.

[27] Microsoft Learn. *What are managed identities for Azure resources?* - Website. Último acesso em 17 de setembro de 2024. 2023. URL: https://learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/overview.

[28] Microsoft Learn. *Tutorial: Access Microsoft Graph from a secured .NET app as the app*. - Website. Último acesso em 17 de setembro de 2024. 2023. URL: https://learn.microsoft.com/en-us/azure/app-service/scenario-secure-app-access-microsoft-graph-as-app?tabs=azure-powershell.

[29] Microsoft Learn. *Use managed identity for authentication among Azure App Service/-Functions*. - Website. Último acesso em 17 de setembro de 2024. 2022. URL: https://learn.microsoft.com/en-us/shows/jdconf-2022/use-managed-identity-for-authentication-among-azure-app-servicefunctions.

[30] Internet Assigned Numbers Authority (IANA). *vCard Elements*. - Website. Último acesso em 17 de setembro de 2024. 2024. URL: https://www.iana.org/assignments/vcard-elements/vcard-elements.xhtml.

[31] Wikipedia. *vCard*. - Website. Último acesso em 17 de setembro de 2024. 2024. URL: https://en.wikipedia.org/wiki/VCard.

[32] J. Morgan. *How to Build a QR Code Generator Using Azure Functions*. - Website. Último acesso em 17 de setembro de 2024. 2023. URL: https://www.allhandsontech.com/programming/dotnet/how-to-qr-code-generator-azure-functions/.

[33] Manuel Blechschmidt. *QrCodeGenerator*. - Website. Último acesso em 17 de setembro de 2024. 2023. URL: https://github.com/manuelbl/QrCodeGenerator.

[34] Microsoft. *Microsoft Power Automate documentation*. - Website. Último acesso em 17 de setembro de 2024. 2024. URL: https://learn.microsoft.com/en-us/power-automate/.

[35] Power Apps Community Blog. *Generate Word Document Template Using Power Automate*. - Website. Último acesso em 17 de setembro de 2024. 2022. URL: https://community.powerplatform.com/blogs/post/?postid=ba99f35a-49ce-4b48-bd9a-263449f49e2e.