UNIVERSIDADE
BEIRA INTERIOR

Development of a Salesforce Solution
From Discovery to Implementation

Rita Ribeiro Martins

Relatório de Estágio Para Obtenção de Grau de Mestre
Engenharia Informática
(2º ciclo de estudos)

Orientador: Professor Doutor Frutuoso Gomes Mendes da Silva
Co-orientador: Gabriela Levy Dinkhuysen

Junho de 2024

Declaração de Integridade

Eu, Rita Ribeiro Martins, que abaixo assino, estudante com o número de inscrição M12337 de Mestrado em Engenharia Informática da Faculdade das Engenharias, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o Código de Integridades da Universidade da Beira Interior.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 07/06/2024

Rita Martins

Resumo

Este relatório detalha a experiência de estágio focada na aprendizagem e implementação do Salesforce, uma plataforma proeminente no domínio das Customer Relationship Management (CRM). A fase inicial destaca uma imersão aprofundada no Salesforce, abrangendo a assimilação de conhecimentos essenciais, compreensão arquitetural e familiarização com as ferramentas. Simultaneamente, são desenvolvidas habilidades práticas na condução de descobertas de projetos, enfatizando a importância da fase de Descoberta no ciclo de vida de um projeto.

À medida que o estágio avança, os objetivos voltam-se para a aplicação prática do conhecimento do Salesforce e das metodologias de *Discovery*. O estagiário é desafiado a desenvolver diversas funcionalidades para atender aos objetivos identificados durante a fase de *Discovery*. Essa fase posterior não exige apenas a aplicação de conhecimentos teóricos, mas também demanda a eficaz tradução dos requisitos identificados em soluções funcionais e eficientes.

Os objetivos gerais do estágio entrelaçam-se harmoniosamente entre a proficiência no Salesforce, uma compreensão profunda do processo de *Discovery* e a aplicação habilidosa desse conhecimento na criação de funcionalidades. O relatório fornece uma visão abrangente da jornada de estágio, capturando a curva de aprendizagem e aplicações práticas no ambiente dinâmico do Salesforce e na descoberta de projetos.

Palavras-chave

Salesforce, Discovery, CRM

Abstract

This report details the internship experience focused on learning and implementing Salesforce, a prominent platform in the realm of CRM. The initial phase emphasizes an in-depth immersion into Salesforce, covering essential knowledge assimilation, architectural understanding, and tool familiarization. Concurrently, practical skills in conducting project discoveries are honed, emphasizing the significance of the Discovery phase in a project's lifecycle.

As the internship progresses, the objectives shift towards the hands-on application of Salesforce knowledge and discovery methodologies. The intern is challenged to develop diverse functionalities to fulfill the goals identified during the Discovery phase. This later stage not only requires the application of theoretical knowledge but also demands effective translation of identified requirements into functional and efficient solutions.

The overarching objectives of the internship seamlessly intertwine Salesforce proficiency, a deep comprehension of the discovery process, and the skillful application of this knowledge to create functionalities. The report provides a comprehensive overview of the internship journey, capturing the learning curve and practical applications in the dynamic environment of Salesforce and project discovery.

Keywords

Salesforce, Discovery, CRM

Índice

Lista de Figuras

Lista de Acrónimos

QA Quality Assurance

UI User Interface

US *User Stories*

WI Work Item

API Application Programming Interface

CRM Customer Relationship Management

DML Data Manipulation Language

ERP Enterprise Resource Planning

LWC *Lightning Web Components*

ONG Organizações Não Governamentais

SOW *Statement Of Work*

SaaS Software as a Service

SOQL Salesforce Object Query Language

SOSL Salesforce Object Search Language

VS Code Visual Studio Code

Capítulo 1

Introdução

1.1 Contextualização

Este documento foi desenvolvido no âmbito da unidade curricular de Dissertação ou Estágio em Engenharia Informática, inserida no segundo ano do Mestrado de Engenharia Informática, na Universidade da Beira Interior. O estágio foi realizado na organização Raise N' Go [1], em Castelo Branco, a qual foca o desenvolvimento Salesforce [2].

Para a realização deste projeto a estagiária, Rita Martins, foi integrada numa equipa que realiza tarefas para Organizações Não Governamentais (ONG), de forma a aprender e implementar soluções na área das CRM com a plataforma Salesforce.

Os orientadores foram o Professor Doutor Frutuoso Silva, por parte da Universidade da Beira Interior, e a Consultora Gabriela Levy, por parte da Raise N' Go.

1.2 Caracterização Geral da Organização

A Raise N' Go [1] foi fundada em 2016, motivada pela busca e entrega de soluções que ajudem os seus clientes a servir outros clientes. A organização começou por realizar projetos em Salesforce para organizações Portuguesas do terceiro setor e posteriormente, em 2017, expandiu-se internacionalmente. Em 2021, a mesma focou-se também no setor comercial. A figura 1.1 apresenta os marcos mais importantes da Raise N' Go.

Figura 1.1: Esta figura mostra os marcos mais importantes da empresa Raise N' Go.

O principal objetivo da Raise N' Go é fornecer aos doadores uma maneira fácil e otimizada dos mesmos comunicarem e armazenarem dados dos seus clientes atuais e potenciais

ao integrar a plataforma CRM, Salesforce.

A Raise N' Go foca-se em perceber os objetivos das Organizações de forma a conseguir implementar da melhor forma o CRM. Sempre que possível, a mesma realiza também integrações com outros sistemas como *Gateways* de pagamentos pontuais e regulares, soluções de *marketing automation*, sistemas de Enterprise Resource Planning (ERP) e faturação, *Websites*, *Webshops*, *Apps* e *Web-Services*, entre outros. Para além do serviço de consultadoria na implementação do CRM, a empresa tem ainda a capacidade de fornecer um serviço de suporte e desenvolvimento.

A empresa conta com mais de 100 projetos, em 10 países diferentes, sendo alguns deles para organizações como a UNICEF [3], Cruz Vermelha [4], Amnistia [5], Associação Salvador [6], Aldeias SOS [7], entre outras.

A figura 1.2 seguinte apresenta o logo da empresa.

Figura 1.2: Esta figura representa o logo da empresa Raise N' Go.

1.3 Objetivos do Estágio

Os objetivos do estágio englobam uma abordagem gradual e abrangente, focada na aprendizagem da plataforma Salesforce e práticas associadas. Inicialmente, procura-se uma imersão na plataforma, priorizando a assimilação de conhecimentos essenciais, a compreensão da arquitetura e a familiarização com as ferramentas da Salesforce. Paralelamente, destaca-se a aquisição de habilidades práticas na condução de descobertas em projetos, visando compreender procedimentos, metodologias e a importância da fase de *Discovery* no ciclo de vida de um projeto.

Numa etapa subsequente, os objetivos evoluem para a aplicação prática dos conhecimentos adquiridos, desafiando a estagiária a desenvolver funcionalidades alinhadas aos objetivos identificados durante a fase de *Discovery*. Esta etapa posterior visa não apenas a aplicação do conhecimento teórico, mas também a habilidade de traduzir efetivamente os requisitos identificados durante a descoberta em soluções práticas e eficientes.

Dessa forma, os objetivos do estágio buscam integrar de forma coerente o domínio da plataforma Salesforce, a compreensão do processo de *Discovery* e a aplicação prática desses conhecimentos na criação de funcionalidades alinhadas com as necessidades identificadas.

1.4 Propriedade Intelectual

Razões relativas ao direito de propriedade intelectual justificam limitações na descrição do trabalho desenvolvido no âmbito deste estágio, de forma a proteger os interesses do cliente e da empresa. Com isto, boa parte do processo de desenvolvimento, arquiteturas e informações não podem ser partilhadas neste documento.

1.5 Organização do Documento

De modo a refletir o trabalho realizado, este documento encontra-se dividido da seguinte forma:

- Primeiro Capitulo - Introdução - Apresenta a empresa na qual foi realizado o estágio e clarifica os objetivos do projeto e a organização do documento.
- Segundo Capítulo - Enquadramento - Apresenta as diversas etapas do processo de *Discovery* e apresenta também a evolução das CRM.
- Terceiro Capítulo - Ferramentas Utilizadas - Apresentas as ferramentas utilizadas no decorrer no estágio.
- Quarto Capítulo - Trabalho Desenvolvido - Apresenta algumas funcionalidades desenvolvidas bem como algumas práticas tomadas no decorrer do estágio.
- Quinto Capítulo - Conclusão - Apresenta as metas alcançadas e alguns problemas encontrados.

Capítulo 2

Enquadramento

2.1 Introdução

Este capítulo tem como finalidade dar a conhecer as diversas etapas do processo de *discovery*, baseadas no livro *"Salesforce Discovery 101: How Salesforce Partners Execute Discoveries to set Projects up for Success"*, escrito por Pei Mun Lim. [8]

Para além disso, o presente capítulo irá ainda apresentar a evolução das CRM e o aumento da adoção desta estratégia para melhorar as relações com os clientes.

2.2 *Discovery*

O processo de *Discovery* tem como objetivo delinear as necessidades de um cliente ao estruturar todos os requisitos quer das funcionalidades a serem implementadas como do próprio cliente.

Assim, nesta secção serão elaboradas as seguintes etapas do processo de *Discovery*:

1. Pré Requisitos;
2. Atribuições;
3. *Workshops*;
4. Metodologias;
5. Mapas de Processos;
6. *User Stories* (US);

As etapas serão abordadas de acordo com o livro escrito por Pei Mun Lim, uma consultora de Salesforce, líder de uma academia de excelência em entrega de projeto e consultadoria.

2.2.1 Pré Requisitos

Antes de começar o processo de *Discovery* de um projeto é importante que seja desenvolvido um *Statement Of Work* (SOW).

Um SOW tem como objetivo estabelecer as expetativas para o projeto e abordar aquilo a que a consultora se compromete fazer. Por outras palavras, pode-se definir um SOW como um acordo ou um contrato entre um cliente e um prestador de serviços que garante um entendimento claro e mútuo sobre o que será realizado

O SOW pode ser divido em várias tópicos:

- Escopo e Objetivo: Descreve os objetivos do projeto e o que deverá ser alcançado;
- Cronograma: Apresenta a estimativa de tempo de cada fase do projeto e as etapas importantes;
- Entregáveis: Descreve os produtos, resultados ou 'itens' a serem entregues no final de cada etapa e no final do projeto;
- Requisitos e Critérios de Aceitação: Especifica quais as métricas de avaliação das funcionalidades ou produtos ou os critérios que os mesmo devem atender.
- Responsabilidades: Define o papel que o cliente e o prestador de serviços deve desempenhar.
- Recursos: Especifica quais os recursos necessários para a execução do projeto tais como equipa e habilitações da mesma, equipamentos, entre outros.
- Orçamento e pagamento: Detalha o custo do projeto, as cláusulas de pagamento, condições contratuais, etc.
- Termos e Condições: Aspetos legais como acordos de confidencialidade, propriedade intelectual, termos de rescisão, entre outros.

É importante ressaltar que a estrutura e os tópicos de um SOW pode variar dependendo do tipo de projeto. Essencialmente, o objetivo é garantir que há um entendimento claro de ambas as partes envolvidas no projeto sobre o trabalho a ser executado.

2.2.2 Atribuições

Para garantir que o processo de *discovery* seja conduzido de forma eficaz é necessário fazer várias atribuições. As mais comuns são:

- *Functional Lead* - Responsável por supervisionar e coordenar todo o processo de *discovery*. É a pessoa que faz a ligação entre o cliente e a equipa de implementação, que conduz os *workshops*, entre outros.
- *Functional Consultant* - Responsável por trabalhar em estreita colaboração com o cliente para compreender as suas necessidades e traduzi-las em soluções viáveis.
- *Bussiness Analyst* - Responsável pela coleta de dados, análise e documentação dos requisitos de negócios para compreender as necessidades dos *stakeholders* e definir o âmbito do projeto.
- Facilitador - Responsável por conduzir os *workshops*, orientando as discussões, estimulando a participação ativa dos participantes e garantindo que os objetivos dos *workshops* são alcançados.

- *Stakeholders* - Indivíduos com interesses diretos ou indiretos numa organização, classificados como internos (funcionários, gestores diretos) e externos (clientes, fornecedores, parceiros de negócios). Os mesmos desempenham um papel crucial ao fornecer informações essenciais para orientar o âmbito do projeto, validando os entregáveis para garantir a sua adequação às necessidades, facilitando assim a comunicação com consultores. Participam ativamente em workshops para discutir, validar e esclarecer requisitos. Na fase de *Discovery*, uma gestão eficaz dos *stakeholders* pode proporcionar à equipa uma compreensão mais clara das expectativas do utilizador final, contribuindo assim para a criação de soluções e requisitos alinhados com as necessidades.

É importante salientar que a mesma pessoa pode assumir vários papeis dependendo da dimensão do projeto. Por exemplo, quando perante um pequeno projeto, um *Bussiness Analyst* pode assumir também o papel de facilitador.

Para que o processo de *discovery* seja conduzido de forma eficaz é fundamental que haja uma colaboração e uma definição clara dos diversos papeis. O trabalho conjunto de todas as atribuições descritas desempenha um papel crítico na compreensão detalhada das necessidades, na definição clara dos requisitos, no alinhamento das expetativas e na garantia que todas as tarefas serão executadas.

2.2.3 *Workshops*

Durante a fase de *Discovery* os *workshops* são úteis para explorar as necessidades, identificar requisitos e alinhar expetativas entre o cliente a o consultor.

Os *workshops* são geralmente pequenas reuniões nas quais se trazem assuntos claros e concisos de forma a chegar a um resultado mais rápido. Os mesmos são conduzidos por consultores responsáveis por garantir a participação ativa de todos os presentes e pelo controlo do tempo.

Quando a preparar um *workshop* é importante considerar alguns aspetos como estabelecer objetivos claros, definir regras básicas de conduta, determinar o material a ser utilizado para orientar a reunião (apresentações, documentos em PDF, entre outros) e antecipar possíveis dificuldades ou desafios que possam surgir durante o evento.

A figura 2.1 ilustra um guia abrangente que abarca todos os estágios de um *workshop*, desde a conceção e o propósito inicial, até a realização do próprio *workshop* e os resultados esperados que devem ser entregues após sua conclusão.

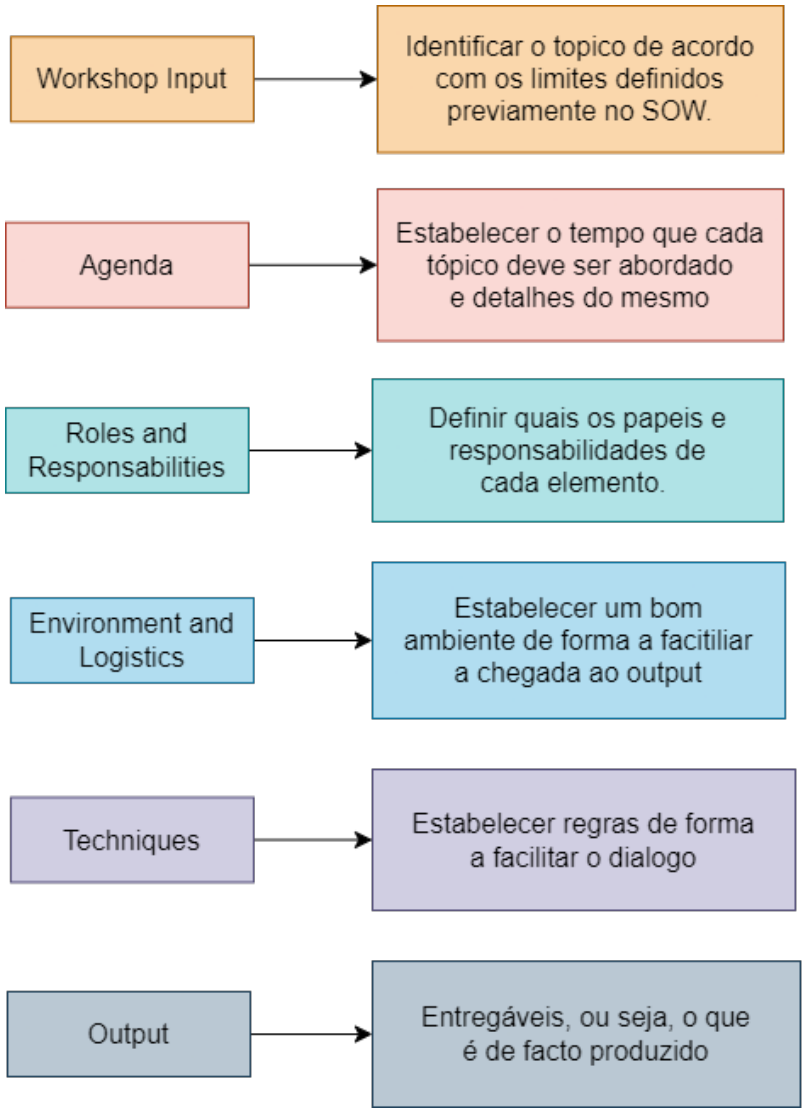

Figura 2.1: Esta figura exemplifica as diretrizes para a organização de um *workshop* eficiente.

Durante o *workshop* em si, a equipa da parte do cliente e do consultor colaboram, discutem ideias, reveem processos, identificam requisitos, elaboram estratégias e tomam decisões.

É após estas reuniões que o *output*, ou os entregáveis, são geralmente produzidos. Estes são os resultados documentados e consolidados das discussões e decisões que ocorrem durante a reunião em si. Para o desenvolvimento do mesmo há geralmente uma equipa dedicada ao projeto que tem como objetivo compilar e consolidar as informações coletadas nos *workshops* em formatos utilizáveis como documentos, relatórios, protótipos, etc.

Após a elaboração dos entregáveis os mesmos são partilhados com o cliente de forma a que os mesmos os possam rever e validar de acordo com critérios de aceitação previamente definidos no SOW.

É crucial ressaltar que todas as fases até à conclusão de um entregável final estão incorporadas no processo "As-Is". Este processo envolve a criação de protótipos dos entregáveis com base nas informações adquiridas do cliente ao longo dos *workshops*. Assim, é importante que haja uma compreensão detalhada do processo atual, ao identificar as atividades que serão executadas, quais as fragilidades, entre outros. Portanto, antes de implementar

qualquer mudança é então necessário compreender o funcionamento atual.

Posteriormente à definição do estado dos processos "As-Is", segue-se a definição do "To--Be", que se concentra na aprimoração dos entregáveis com base nos protótipos existentes e no *feedback* fornecido pelo cliente. Este processo envolve o redesenho do processo para uma versão otimizada e inovadora de modo a superar as limitações identificadas na fase "As-Is". O objetivo é então criar um modelo do processo alinhado com os objetivos do cliente, que leve a melhorias significativas.

Na secção 2.2.5 ambos os processos serão abordados numa perspetiva diferente.

2.2.4 Metodologias

Numa *Discovery* é importante adotar uma metodologia adaptada ao contexto do projeto a desenvolver. A metodologia deve ser escolhida após conversas com o cliente de forma a perceber qual a metodologia que melhor se adequaria ao mesmo.

Pode-se definir metodologias como uma *framework* usada para estruturar, planear e controlar o processo de desenvolvimento de um sistema informático. É importante adotar uma metodologia para que todas os trabalhadores envolvidos no projeto saibam o que estão a fazer, quais as ferramentas e modelos a utilizar e como os utilizar de forma correta.

Em projetos Salesforce é geralmente a metodologia *Hybrid* a escolhida para o desenvolvimento de um projeto, metodologia essa que é uma combinação da *Waterfall* [9] e *Agile* [10].

De uma forma geral, e para tentar compreender o conceito da metodologia *Hybrid* é necessário saber alguns aspetos das outras duas metodologias.

A metodologia *Waterfall* segue um modelo sequencial de desenvolvimento, no qual o processo é divido em fases, onde antes de avançar para uma fase seguinte é necessário concluir a fase anterior. Uma desvantagem desta metodologia é a inflexibilidade em lidar com mudanças uma vez que fazer alterações significativas a meio do processo pode ser complicado e levar a erros.

Já a metodologia *Agile* promove a entrega rápida, flexível e adaptável ao valorizar as mudanças e a entrega incremental de partes do projeto ao longo do tempo. Contrariamente à *Waterfall* esta metodologia é realizada por *sprints* onde em cada *sprint* é entregue e apresentado o trabalho desenvolvido ao longo de duas semanas.

Finalmente, a metodologia *Hybrid* combina os princípios do *Agile* e *Waterfall*. Ela é adotada em projetos que demandam flexibilidade para lidar com mudanças e interações frequentes com o cliente (referentes ao *Agile*), ao mesmo tempo em que requer uma estruturação detalhada e previsibilidade em certos aspetos do projeto (em consonância com o *Waterfall*). Por exemplo, um projeto pode incluir a entrega progressiva, feedback contínuo e colaboração com o cliente, ao mesmo tempo em que integra elementos mais estruturados e sequenciais, como a definição detalhada de requisitos no início do processo.

O ambiente Salesforce exige que haja flexibilidade em ajustar os requisitos à medida que as necessidades dos clientes mudam. Por esse motivo a metodologia Agile pode ser aplicada para que hajam entregas incrementais. Ao mesmo tempo, algumas partes do projeto podem exigir que os conceitos sejam mais estruturados, e requerem uma abordarem mais *Waterfall*.

Assim, a metodologia *Hybrid* é a melhor abordagem quando perante os aspetos referidos.

De forma a dar a conhecer melhor esta metodologia, segue-se o seguinte exemplo, o qual diz respeito a um projeto fictício de desenvolvimento de software para uma empresa que deseja lançar uma nova plataforma de comércio eletrónico. Neste cenário a metodologia Hybrid é adotada para combinar estratégias Agile com estratégias Waterfall. Os passos desse projeto seriam:

1. Planeamento Inicial: Análise inicial para identificar os requisitos principais e objetivos do projeto. Estruturação de uma visão geral do projeto com as metas e marcos principais.

2. Definição Detalhada de Requisitos: Utilização do Waterfall para definir mais detalhadamente os requisitos do sistema. Criação de documentação que inclui as especificações das funcionalidades, da User Interface (UI) e dos requisitos não funcionais.

3. Design e Estruturação do Projeto: Desenvolvimento da arquitetura do sistema com base nos requisitos definidos. Estruturação de fases especificas do projeto e identificação de possíveis dependências e riscos.

4. Início das Iterações Agile: Implementação de ciclos para o desenvolvimento de partes especificas do sistema. Realização de iterações curtas focadas nas funcionalidades prioritárias com entregas incrementais.

5. Comunicação Contínua com o Cliente: Agendamento de reuniões regulares com o cliente para revisão e *feedback* contínuo das funcionalidades. Ajuste dos requisitos com base nas interações com o cliente e nas mudanças identificadas durante as iterações Agile;

6. Mitigação de Riscos ao Longo do Projeto: Identificação proativa de potenciais riscos e elaboração de estratégias para a mitigação dos mesmos. Adaptação de estratégias de gestão de riscos com base na *Discovery* e no progresso do projeto.

7. Entregas Incrementais: Entrega de partes funcionais do sistema à medida que são concluídas durante as iterações Agile. Avaliação continua das entregas por parte o cliente, permitindo ajustes conforme necessário.

8. Finalização e Implementações: Conclusão do desenvolvimento iterativo e realização de testes abrangentes. Lançamento da plataforma com melhorias continuas e adaptações ao longo do projeto.

2.2.5 Mapas de Processos

Durante a fase de *discovery*, os mapas de processos representam visualmente os fluxos de trabalho, procedimentos e funcionalidades da organização ou do projeto. Os mesmos oferecem uma compreenssão clara de como os métodos serão ou estão a ser executados.

Numa fase inicial, fase essa inserida no processo ”As-Is” mencionado na secção 2.2.3, é elaborado um esboço inicial do mapa de processos. Este esboço inicial começa por identificar o objetivo do processo, ao obter informações em documentações, entrevistas, workshops e outras fontes de informação. Posteriormente, passa-se para a análise e coleta dos requisitos, nos quais os mesmos vão sendo refinados de acordo com o *feedback* do cliente nas sessões de *workshop*. Após assegurar que o âmbito do projeto ou da funcionalidade foi todo abrangido, é delineado o mapa de processos final, dentro do processo ”To-Be”. Nesta etapa são removidos e consolidados passos e atividades provenientes do esboço feito no processo ”As-Is”.

Esta transição entre o estado atual e o estado desejado permite aprimorar e otimizar processos ao considerar as melhorias discutidas ao longo do processo de *discovery*.

Na figura 2.2, estão delineados os passos a serem seguidos durante a construção de um mapa de processos.

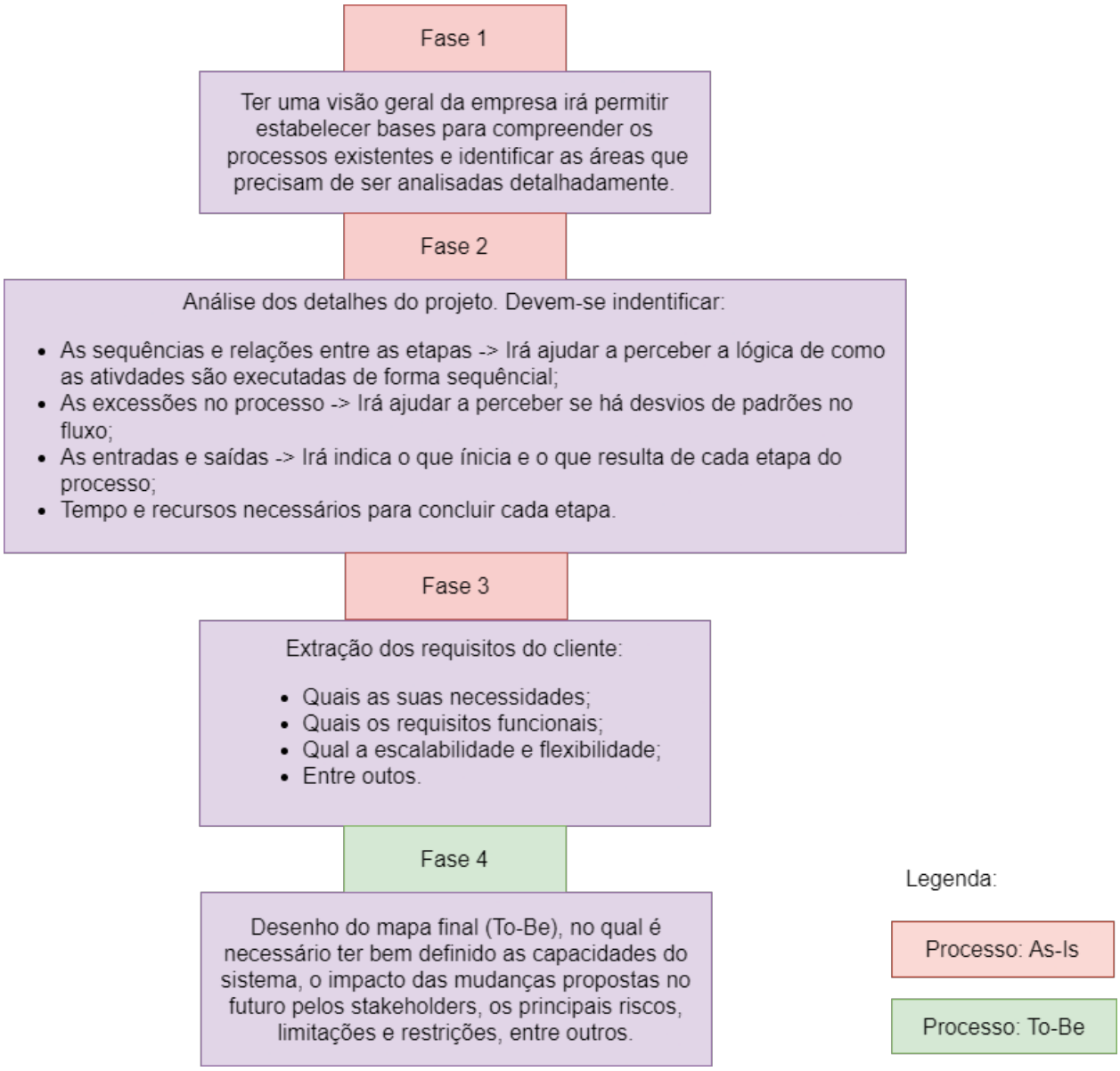

Figura 2.2: Esta figura ilustra a sequência de passos a serem seguidos durante a elaboração de um mapa de processos.

2.2.6 *User Stories*

No processo de *discovery*, as US são ferramentas ágeis usadas para capturar e comunicar requisitos no ponto de vista do utilizador.

As mesmas são narrativas claras e concisas, geralmente redigidas na perspetiva do utilizador final, e descrevem uma funcionalidade especifica que o sistem ou produto deve ter. A sua estrutura começa geralmente com a identificação do ator, descrevendo o que o mesmo deseja alcançar, seguido do beneficio resultante dessa ação.

A figura 2.3 representa a estrutura das US.

Figura 2.3: Esta figura representa a estrutura de uma *User Storie*.

As US são ainda complementadas por critérios de aceitação que contém todos os passos necessário para a conclusão bem sucedida da US. Esses critérios devem começar com o evento que desencadeia a US, seguido do resultado espetado após a execução desse mesmo evento.

Quando combinadas com os critérios de aceitação, as US fornecem uma visão detalhada das funcionalidades desejadas, o que irá facilitar o desenvolvimento de soluções que vão ao encontro das necessidades e requisitos dos clientes. Para além disso, as mesmas permitem ainda uma comunicação clara entre os stakeholders e as equipas do projeto, ajudando a alinhar as expetativas, prioridades e requisitos.

De notar que as US são geralmente escritas partindo dos mapas de processos, desenvolvidos numa fase anterior, uma vez que os mesmos ilustram uma compreensão detalhada das funcionalidades e operações da empresa, que pode ser usado como ponto de partida para identificar funcionalidades especificas e requisitos do sistema.

2.3 CRM

O CRM [11] representa uma abordagem completa que incorpora a gestão total das áreas de vendas, marketing, atendimento e todos os pontos de interação com o cliente, sendo frequentemente denominada de Gestão 360º.

Atualmente, atravessamos uma fase de transformação digital que tem influenciado e continua a influenciar a evolução das relações com os clientes. Esse contexto deu origem ao conceito de CRM, o qual enfatiza a importância da experiência do cliente por meio de estratégias, processos, ferramentas e tecnologias. As plataformas de CRM têm ganhado destaque, uma vez que centralizam as informações dos clientes, permitindo a gestão de contas, *leads* e oportunidades de vendas num único local.

Uma grande vantagem dos CRM é a transformação dos processos manuais, geralmente realizados em papel, em processos digitais, processos esses capaz de organizar as contas e contactos de forma acessível, em tempo real, o que acelera o processo de vendas.

Existem vários tipos de CRM:

- **CRM Local** - Refere-se a um CRM instalado num servidor físico dentro da empresa, necessitando de manutenção por parte equipa especializada para sua gestão e atualização.
- **CRM na Nuvem** - Refere-se a um CRM baseado em *Cloud Computing* (um sistema de computação fornecido através de uma rede segura de servidores remotos hospedados na *internet*), que não requer instalação em máquinas locais e dispensa a necessidade de uma equipa específica para manutenção. Esse CRM online também pode ser denominado como Software as a Service (SaaS), já que é gerido remotamente por especialistas. O SaaS pode ser definido como um modelo de distribuição de software no qual as aplicações são hospedadas remotamente por um fornecedor de serviços na nuvem e disponibilizado aos utilizadores via *internet*. Algumas das suas vantagens são:
  - Acessibilidade Remota - É possível aceder ao software a partir de qualquer local desde que haja conexão à *internet*, o que proporciona maior flexibilidade e mobilidade;
  - Manutenção e Atualizações Simplificadas - As atualizações são geralmente realizadas pelo fornecer de serviço o que leva à diminuição da carga de trabalho dos utilizadores;
  - Suporte Técnico - Geralmente os serviços SaaS incluem suporte técnico o que é uma grande vantagem para os utilizadores em caso de problemas ou duvidas;
  - Atualizações automáticas - As atualizações de software são automáticas e transparentes para os utilizadores, o que leva a que os mesmos tenham acesso a novas funcionalidades e melhorias;
  - Integração e Compatibilidade - Os serviços SaaS são desenhados de modo que sejam compatíveis e integráveis com outras ferramentas ou aplicações.

Capítulo 3

Ferramentas Utilizadas

3.1 Introdução

O propósito deste capítulo é elucidar as diversas ferramentas empregadas durante o estágio, com ênfase na plataforma central, o Salesforce. Para além desta, serão abordadas ferramentas essenciais para o desenvolvimento complementar ao Salesforce.

3.2 Ferramentas Utilizadas

3.2.1 Salesforce

Salesforce [2] é uma plataforma CRM usada por empresas para gerir interações com clientes e melhorar relações comercias. A mesma oferece uma vasta variedade de funcionalidades como vendas, serviços ao cliente, marketing, automação de processos e análise de dados.

Criada em 1999 por Marc Benioff, ex-executivo da Oracle, a Salesforce é uma plataforma americana conhecida globalmente por ser uma das maiores empresas de software, ao adotar o modelo de entrega de serviços SaaS.

A capacidade de personalização da plataforma é uma das suas principais características, uma vez que a mesma se pode adaptar e configurar conforme as necessidades de cada cliente. Além disso, a plataforma tem uma grande escalabilidade o que permite que a mesma cresça com os requisitos e tamanho da empresa.

Dependendo dos requisitos do projeto a implementação de funcionalidades no Salesforce pode seguir diversas metodologias, como a *Waterfall*, *Agile* e *Hybrid* (mencionadas na secção 2.2.4. A flexibilidade da plataforma permite que as empresas escolham a metodologia que mais se adequa às mesmas e aos requisitos do projeto.

O ecossistema do Salesforce, oferece ainda integrações com outras ferramentas e sistemas, bem como uma vasta gama de aplicações que se encontram disponíveis na Salesforce AppExchange, que consiste num *hub* que contém aplicações disponibilizadas pela própria Salesforce ou outros parceiros. Isso amplia as capacidades da plataforma ao oferecer soluções rápidas para diferentes necessidades comerciais.

Para além disso, o Salesforce fornece ainda recursos de treino, suporte e comunidades ativas para os utilizadores, garantindo assim maximização do potencial da plataforma.

Na imagem 3.1 está representado o *logo* da plataforma Salesforce.

Figura 3.1: Esta figura representa o logo da plataforma Salesforce.

3.2.2 Salesforce Developer Console

Dentro da plataforma Salesforce, existe uma ferramenta denominada de Developer Console [12]. Essa ferramenta foi projetada para auxiliar os desenvolvedores no processo de criação, depuração, testagem e personalização de aplicações.

A mesma permite o desenvolvimento de código na plataforma Salesforce sem a necessidade de instalar softwares adicionais. Assim os desenvolvedores podem criar componentes como Visualforce Pages (referidas na subsecção 3.2.4), Apex Classes (referidas na subsecção 3.2.5,) entre outros, diretamente na Developer Console.

A mesma é ainda composta por um editor de *querys*, no qual é possível executar comandos de pesquisa Salesforce Object Query Language (SOQL) e Salesforce Object Search Language (SOSL) aos dados armazenados no Salesforce. O SOQL tem como objetivo pesquisar dados de um único objeto ou de objetos relacionados no Salesforce. Já o SOSL tem como objetivo pesquisar dados através de termos existentes num determinado campo.

Na figura 3.2, está representada a Developer Console. Na área 1, está o editor de código que proporciona a capacidade de escrever, visualizar e modificar código. Na área 2, é possível observar um conjunto de abas que possibilitam a visualização de *logs*, erros, a escrita no editor de *querys*, entre outras funcionalidades.

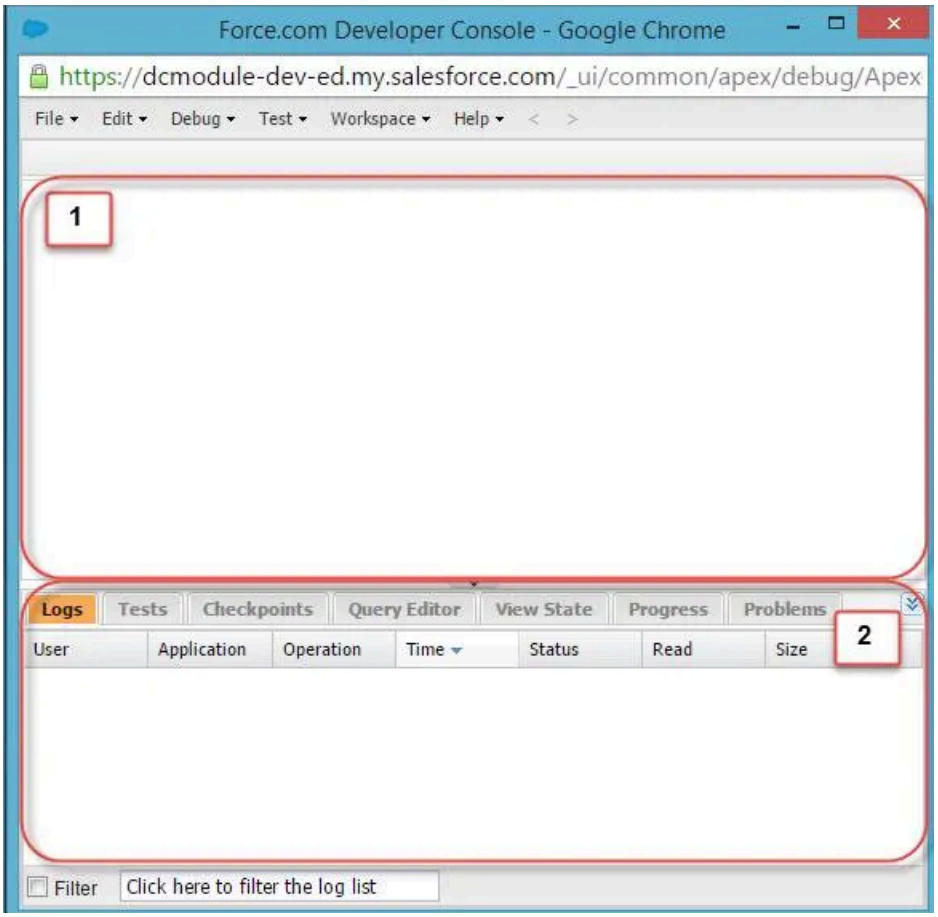

Figura 3.2: Esta figura representa a Developer Console. Fonte: https://trailhead.salesforce.com/today

3.2.3 Salesforce DevOps

O Salesforce DevOps [13] é uma plataforma que permite a entrega e integração continua na plataforma Salesforce. Na mesma é possível encontrar ferramentas de automação e integração de forma a melhorar a qualidade do código e a entrega de soluções de forma rápida e confiável.

O termo DevOps é uma combinação das palavras ”Desenvolvimento” (Dev) e ”Operações” (Ops). O Salesforce DevOps visa integrar essas duas áreas de forma a facilitar a entrega continua de valor aos clientes.

No contexto do Salesfoce, o Devops, permite que seja feito o controlo de versões (através do Git), automação de testes, entre outros. Tais funcionalidades permitem às equipas de desenvolvimento e operações a colaboração de forma eficaz ao ser possível corrigir problemas rapidamente garantindo assim que as atualizações e novas funcionalidades sejam entregues com segurança e qualidade.

3.2.4 VisualForce

Visualforce [14] é uma *framework* que viabiliza a construção de UI hospedadas nativamente na Lightning Platform. Baseada na linguagem HTML, essa *framework* dispõe de controladores padrão que simplificam operações na base de dados.

Dentro do Visualforce, cada *tag* corresponde a um componente da UI, como secções de páginas, campos ou *related lists*. O comportamento desses componentes pode ser controlado pela mesma lógica empregada em páginas padrão do Salesforce ou pode ser associado a uma lógica própria, definida por um controlador Apex vinculado a eles.

Além disso, a estrutura possibilita a utilização de Recursos Estáticos (*Static Resources*) do Salesforce, permitindo o carregamento de conteúdo para a página Visualforce. Esse conteúdo pode incluir arquivos, imagens e outros elementos necessários.

No contexto deste relatório, esta *framework* foi usada para construir páginas com campos e *display* personalizado.

3.2.5 Apex

Apex [15] é uma linguagem de programação orientada a objetos fortemente tipada que permite aos desenvolvedores executar fluxos e permite controlar transações nos servidores Salesforce em conjunto com chamadas a Application Programming Interface (API). O Apex, permite ainda desenvolver lógica relacionada com o negócio entre elas clique de botões, *updates* de *related records*e páginas Visualforce. O código Apex pode ser iniciado por pedidos *web* ou por *triggers* associados aos objetos.
Algumas características desta linguagem são:

- Integração - O Apex permite a manipulação de dados Data Manipulation Language (DML) incluindo chamadas *INSERT*, *UPDATE* e *DELETE* com integração de *DmlException*. Permite ainda consulta SOQL e SOSL que retornam listas de registos *sObject*, *looping* que permite o processamento em massa de vários registos, criação de chamadas personalizadas à API, entre outros;

- Fácil Utilização - O Apex é uma linguagem de programação muito semelhante a Java, pelo que a sua sintaxe a semântica é muito fácil de compreender e de fácil utilização na *Lightning Platform*;

- Fácil de Testar - O Apex oferece suporte integrado para a criação e execução de testes unitários. Os mesmos incluem resultados de teste que indicam qual o código que está coberto e garantem ainda que todo o código está funcional antes de qualquer atualização na plataforma;

- Ambiente *Multitenant* [16] - O Apex opera num ambiente *multitenant* que permite uma instância seja utilizada por várias pessoas diferentes ao mesmo tempo.

3.2.6 LWC

Os LWC [17] são uma *framework* de desenvolvimento de UI projetada pela Salesforce para aprimorar a criação de aplicações na mesma. O objetivo dos mesmos é a construção de uma UI através da estruturação de vários componentes modulares e reutilizáveis. Esses componentes podem ser desenvolvidos em HTML, Javascript, CSS e metadatos declarativos para definir o seu comportamento e aparência.

Algumas vantagens dos LWC são:

- Performance - Os LWC oferecem um carregamento rápido das páginas o que leva a uma experiência de utilizador mais responsiva;

- Integração Direta com o Salesforce - Os LWC estão nativamente integrados com a plataforma Salesforce, permitindo assim o acesso direto a dados, serviços e recursos da mesma;

- Facilidade de Manutenção e Teste - Os desenvolvedores de LWC podem recorrer a testes unitários para garantir a qualidade do código;

- Comunidade Ativa e Recursos de Aprendizagem - O Salesforce disponibiliza uma comunidade ativa na qual os desenvolvedores podem participar para partilhar recursos, exemplos de código, tutoriais e documentação acerca dos LWC.

Assim, os LWC são uma *framework* moderna e eficiente para criar UI na plataforma Salesforce, ao proporcionar aos desenvolvedores uma maneira flexível e eficaz de construir aplicações interativas na nuvem.

3.2.7 VS Code

Criado em 2015, o Visual Studio Code [18] é um editor de código-fonte desenvolvido pela Microsoft. Esta ferramenta apresenta características como:

- UI Simples - A interface apresentada pelo VS Code é intuitiva e amigável, aspeto esse que tona a edição de código mais fácil de usar;

- Extensões - Uma grande vantagem, do VS Code é o seu mercado de extensões, mercado esse que os desenvolvedores podem utilizar para obter funcionalidades extra como integrações com ferramentas de controlo de versões, depuração, entre outros;

- Terminal e Depuração Integrados - Permite aos utilizadores executar comandos diretamente no editor sem a necessidade de estar a alternar entre diferentes programas. Para além disso permite ainda rastrear e corrigir problemas no código de forma eficiente;

- Integração do *Git* - Por defeito o VS Code vem integrado com o *Git*, o que facilita o controlo de versões em projetos hospedados em repositórios *git*;

No contexto deste relatório, esta ferramenta foi utilizada para desenvolver os LWC, mencionados na secção 3.2.6.

Na imagem 3.1 está representado o *logo* da ferramenta VS Code.

Figura 3.3: Esta figura representa o logo da ferramenta VS Code.

Capítulo 4

Trabalho Desenvolvido

4.1 Introdução

Este capítulo descreve o trabalho realizado durante o período de estágio. Desde o desenvolvimento inicial das funcionalidades até à implementação de testes e entrega, o mesmo acompanha e documenta todas as etapas cruciais do ciclo de vida do desenvolvimento das funcionalidades.

4.2 *Formação*

O percurso da estagiária começou com uma formação em Salesforce, com o uso da plataforma Trailhead [19] que é uma plataforma de aprendizagem online oferecida pela Salesforce, projetada para capacitar indivíduos no desenvolvimento de habilidades em Salesforce. A mesma oferece uma ampla variedade de módulos de aprendizagem e projetos práticos que cobrem desde a introdução ao Salesforce até habilidades avançadas de desenvolvimento e administração.

A Salesforce dá a possibilidade de obter certificados, dando assim possibilidade aos indivíduos de demonstrar as suas competências na plataforma. Entre as certificações destacam-se a de *Associate*, *Administrator*, *Developer I*, entre outras. A figura 4.1 mostra alguns módulos de preparação para diversas certificações.

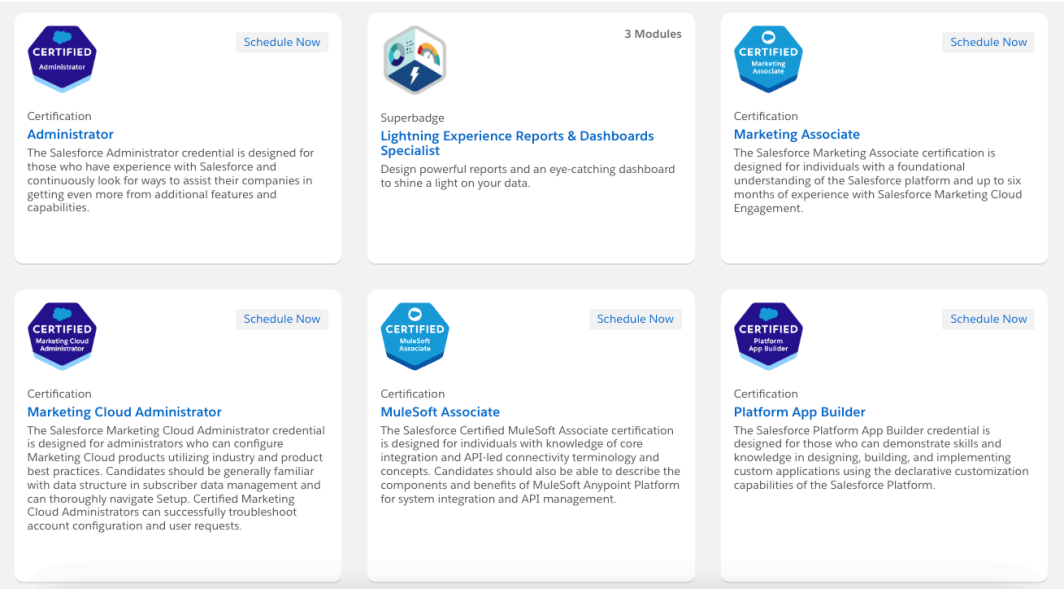

Figura 4.1: Esta figura representa módulos de aprendizagem para diversas certificações no Trailhead.

Inicialmente, foi sugerido à estagiária que estudasse para obter a certificação de *Associate*, que aborda todos os tópicos básicos do Salesforce. Para se preparar para essa certificação, ela seguiu um percurso no Trailhead que fornecia noções básicas da plataforma e permitia interagir pela primeira vez com o Salesforce. Essa etapa inicial foi concluída no primeiro mês do estágio.

A figura 4.2 representa os módulos de aprendizagem realizados para a preparação da certificação *Associate*.

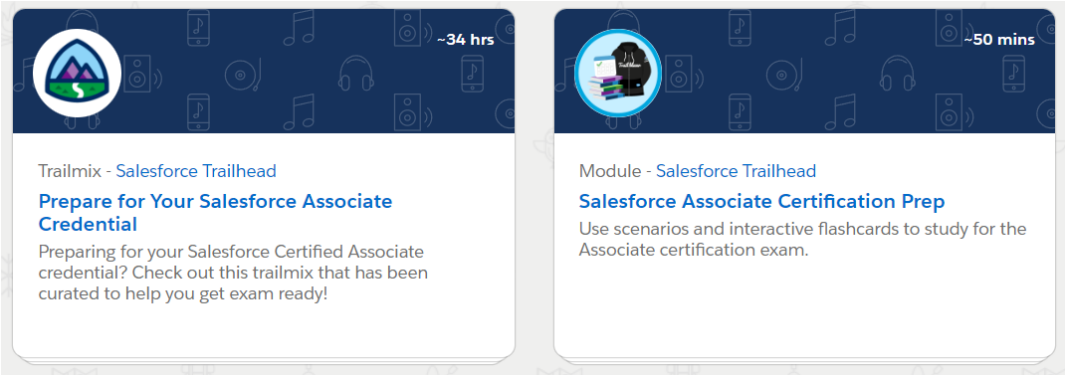

Figura 4.2: Esta figura representa os módulos de aprendizagem para a certificação de *Associate* no Trailhead.

Ao longo dos meses seguintes, a estagiária foi incentivada a estudar, especialmente nos períodos em que não tinha tarefas atribuídas. O objetivo era prepará-la para obter a segunda certificação, a de *Administrator*, que proporcionaria uma compreensão mais profunda da plataforma Salesforce.

No final do estágio, ela obteve com sucesso essa certificação, refletindo assim a sua evolução e aquisição de conhecimento ao longo do período de estágio.

4.3 *Integração*

Após o primeiro mês de estudo, a estagiária foi integrada num projeto. Essa integração começou com uma abordagem detalhada do projeto ao longo de duas semanas, período durante o qual a mesma participou em reuniões de formação feitas pelos colegas do projeto, explorou as funcionalidades existentes e dedicou tempo à leitura da documentação. Essas duas semanas de integração foram cruciais para permitir à estagiária adquirir um conhecimento abrangente do projeto. Considera-se que esse período foi suficiente para ela poder iniciar o seu percurso no projeto, ajudando a implementar novas funcionalidades e resolver problemas já existentes.

Após as duas semanas de integração, a estagiária começou a realizar tarefas simples para praticar e familiarizar-se com os vários aspetos do projeto, como a criação de objetos, campos, dependências e validações. Inicialmente, foi um desafio absorver todas essas informações, dada a complexidade e o tamanho do projeto, mas à medida que a mesma trabalhava em mais tarefas e funcionalidades, o projeto começou a tornar-se mais familiar e fácil de entender.

O apoio contínuo de uma equipa bem preparada, com disponibilidade para ajudar, esclarecer e orientar foi fundamental para a integração bem sucedida da estagiária no projeto.

4.4 *Metodologia de trabalho*

Nesta secção serão apresentadas as principais etapas do processo *Agile*, desde o planeamento e definição de requisitos até à entrega e revisão iterativa.

4.4.1 O que é o *Agile*

A metodologia *Agile* [20] baseia-se em princípios como a iteração incremental, a entrega continua de valor, a colaboração entre a equipa de desenvolvimento e o cliente e a adaptabilidade a mudanças. O foco está na comunicação e *feedback* continuo, permitindo assim realizar ajustes e correções ao longo do projeto.

Dentro das diversas metodologias *Agile*, no decorrer do estágio foi utilizado o *Scrum* [21] que tem como objetivo a divisão do projeto em *sprints* curtos com entregas frequentes e revisões constantes.

4.4.2 *Daily*

Na metodologia *Agile*, a *Daily* é um evento que acontece diariamente com o objetivo claro de manter a equipa sincronizada e focada no objetivo do *sprint*.

Nessa breve reunião, que geralmente não deverá durar mais de 15 minutos, cada membro da equipa informa as tarefas que concluiu no dia anterior, o que planeia fazer no dia atual e os possíveis impedimentos que podem atrapalhar o progresso. O gestor de projeto acompanha o progresso e ajusta o plano do *sprint*, se necessário, enquanto a equipa discute e resolve problemas em conjunto.

A principal vantagem da *Daily* é a comunicação e colaboração contínuas, garantindo que todos estejam informados sobre o andamento do projeto e que a resolução de problemas seja mais rápida e eficiente.

Para que a *Daily* seja eficaz, é importante que seja breve e objetiva, com participação ativa de todos os membros da equipa. O foco deve estar no progresso e nos impedimentos, com honestidade e transparência na comunicação dos desafios e dificuldades.

Ao implementar esta reunião de forma eficaz, as equipas *Agile* podem melhorar significativamente a comunicação, o foco e a colaboração, impulsionando o sucesso do projeto.

4.4.3 *Planning*

No início de cada *sprint*, que tem uma duração de duas semanas, a equipe reúne-se para a reunião de *Planning*. Essa reunião define o que será desenvolvido durante este período e como o trabalho será organizado.

Nessa reunião, o cliente fornece uma lista de funcionalidades que deseja ver abordadas no *sprint*, e em conjunto com a equipa, essas funcionalidades são abordadas e analisadas cuidadosamente.

Durante o *planning*, o cliente prioriza e seleciona do *backlog* (que é uma lista com todas USs e *bugs* do produto) os itens que pretende ver abordados no *sprint*, e estes são analisados em conjunto com a equipa de desenvolvimento.

Cada item é alvo de uma estimativa do tempo previsto que leva para desenvolver/corrigir, deste modo, é possível para o cliente perceber se ainda há espaço no *Sprint* para introduzir novas funcionalidades e também o nível de esforço e complexidade associado a cada tarefa.

No final da reunião de *Planning* a equipa tem um plano claro e detalhado para o *Sprint*, com as tarefas e responsabilidades definidas, as estimativas de tempo para cada tarefa/objetivo estabelecido(a). Isso garante um desenvolvimento organizado, eficiente e focado nos resultados.

4.5 *Desenvolvimento*

Nesta secção, será apresentado o processo de desenvolvimento de algumas funcionalidades do projeto, desde a conceção inicial até à solução final, que atenda às necessidades e expectativas do cliente.

4.5.1 Conceitos

De forma a explicar as funcionalidades desenvolvidas é importante referir a existência de 3 intervenientes:

- **Doadores** - Pessoas que realizam doações para o cliente;
- **Cliente** - Entidade que recebe os donativos;
- **Agência de Pagamentos** - responsável pela gestão da movimentação de fundos da doação realizada.

É também importante explicar o conceito de um **Batch Apex** em Salesforce, utilizado nas várias funcionalidades desenvolvidas. O **Batch Apex** é uma funcionalidade que permite processar grandes volumes de dados ao dividir o trabalho em pequenos lotes que são processados sequencialmente. Isso é útil quando se pretende fazer operações que envolvam uma grande quantidade de registos. Um **Batch Apex** é constituído por 3 métodos principais que devem ser implementados:

- ***Start*** - Responsável por retornar um objeto *Database.QueryLocator* que representa o conjunto de registos a serem processados. Este método é chamado no inicio da execução do *Batch* e é utilizado para definir o âmbito da execução.
- ***Execute*** - Responsável por executar a lógica que será aplicada a cada lote de registos. Este método é chamado para cada lote de registos retornados pelo método *Start*.
- ***Finish*** - Este método é executado no fim do *Batch* após todos os registos terem sido processados no método ***Execute***. Pode ser utilizado para realizar alguma tarefa final, se fizer sentido.

É importante referir que o *Batch Apex* **apresenta um limite de processamento de 200 registos por lote**.

Por fim, é importante destacar que nas funcionalidades a serem desenvolvidas vão ser realizadas operações **DML**, que se referem às operações que manipulam dados no Salesforce, como inserção, atualização e exclusão de registos.

Algumas destas operações, fazem uso de *queries*, que no contexto Apex são instruções escritas na linguagem SOQL, utilizadas para selecionar dados de objetos do *Salesforce*. Através das *queries*, é possível escolher campos de um objeto, filtrar os resultados com condições específicas e ordená-los conforme necessário.
As seguintes operações são as principais operações DML do Apex:

- **Inserção de Registos** - É feito com o uso do método *insert* do Apex. Exemplo:

```
Contact con = new Contact(Name='New Account');
insert con;
```

Listing 4.1: Trecho de código que mostra o insert de um registo

- **Atualização de Registos** - É feito com o uso do método *update* do Apex. Exemplo:

```
Contact con = [SELECT Id, Name FROM Contact LIMIT 1];
con.Name = 'Updated Name';
update con;
```

Listing 4.2: Trecho de código que mostra o update de um registo

- **Exclusão de Registos** - É feito com o uso do método *delete* do Apex. Exemplo:

```
Contact con = [SELECT Id FROM Contact LIMIT 1];
delete con;
```

Listing 4.3: Trecho de código que mostra o delete de um registo

- **Seleção/Consulta de Registos** - É feita com o uso do SOQL. Exemplo:

```
List<Contact> conList = [SELECT Id FROM Contact Where Name = '
    TestName'];
```

Listing 4.4: Trecho de código que mostra o select de um registo

- **Atualização em Lote** - É feita com o uso do método *update* da classe *Database*, que leva como primeiro parâmetro uma lista e como segundo um *boolean* que indica se inserimos todos ou nenhum. Exemplo:

```
Database.update(conList, false);
```

Listing 4.5: Trecho de código que mostra o update em lote

No Salesforce os desenvolvedores fazem o seu desenvolvimento num ambiente de trabalho isolado e independente, de modo a não comprometerem o ambiente utilizada pelo cliente. Essa instância isolada é denominada de ***org*** que é uma abreviação de *Organization*.

4.5.2 Início do desenvolvimento

No início de cada *sprint*, são atribuídas a cada elemento da equipa as US correspondentes. O desenvolvimento começa com a leitura e interpretação da US designada à estagiária. Após uma análise cuidadosa, a estagiária avança para o desenvolvimento.

Primeiramente, é criado um Work Item (WI), na plataforma Salesforce DevOps mencionada na secção 3.2.3. Este WI é responsável por enviar as alterações realizadas para o Git. Ao criar o mesmo, é necessário fornecer uma breve descrição da tarefa, seguida pela escolha do ambiente inicial (que será denominado de ***org***) onde o desenvolvimento será realizado.

De seguida, é realizado o processo de sincronização da *org* de desenvolvimento da estagiária com a *org* de produção, uma vez mais através do Salesforce DevOps. Este passo é crucial para garantir que a *org* tenha todas as atualizações e desenvolvimentos feitos por outros membros da equipa. Posteriormente, o ambiente de desenvolvimento é preparado, com a criação dos objetos necessários e as classes que serão utilizadas para concluir a tarefa.

Durante o desenvolvimento mantém-se uma comunicação ativa com os restantes membros da equipa de forma a esclarecer duvidas e superar obstáculos que possam surgir ao longo do processo.

Após cada desenvolvimento considerado significativo, é realizado um *commit* (que é uma transação) a partir do WI criado no início da tarefa. Neste processo, será primeiramente realizado um *pull* que irá obter as alterações feitas na *org*. De seguida, são selecionadas as classes e objetos modificados, sendo depois fornecida uma descrição das alterações antes de confirmar o *commit*. Este procedimento pode ser repetido conforme necessário até à conclusão do desenvolvimento, garantindo assim o controlo de versões e a possibilidade de reverter para uma versão anterior funcional caso ocorram problemas no código.

4.5.3 Organizações e Ambientes

Para o desenvolvimento de tarefas, são geralmente utilizados 3 ambientes diferentes. Inicialmente existe a *org* de desenvolvimento, onde a tarefa é realizada.

Após a sua conclusão, o desenvolvimento é passado para Quality Assurance (QA), que é uma *org* com uma cópia parcial dos dados da *org* de produção. Na mesma são realizados testes à funcionalidade desenvolvida tanto por outros elementos da equipa como por alguns membros do cliente, garantindo que não haja conflitos entre a nova funcionalidade e outras já desenvolvidas. Além disso, como esta *org* tem uma cópia parcial dos dados de produção, é possível realizar testes com dados reais, assegurando que a funcionalidade está adequadamente desenvolvida.

Por fim, o desenvolvimento é transferido para a Produção, que é a *org* utilizada pelo cliente para armazenar os dados dos seus próprios clientes. Este ambiente deverá ser o espelho do produto que será disponibilizado para o utilizador final, e é utilizado para garantir que as funcionalidades desenvolvidas estejam disponíveis e nos moldes esperados.

A figura 4.3 representa as *orgs* mencionadas. Na mesma é possível verificar os WI que se encontram de momento em cada ambiente.

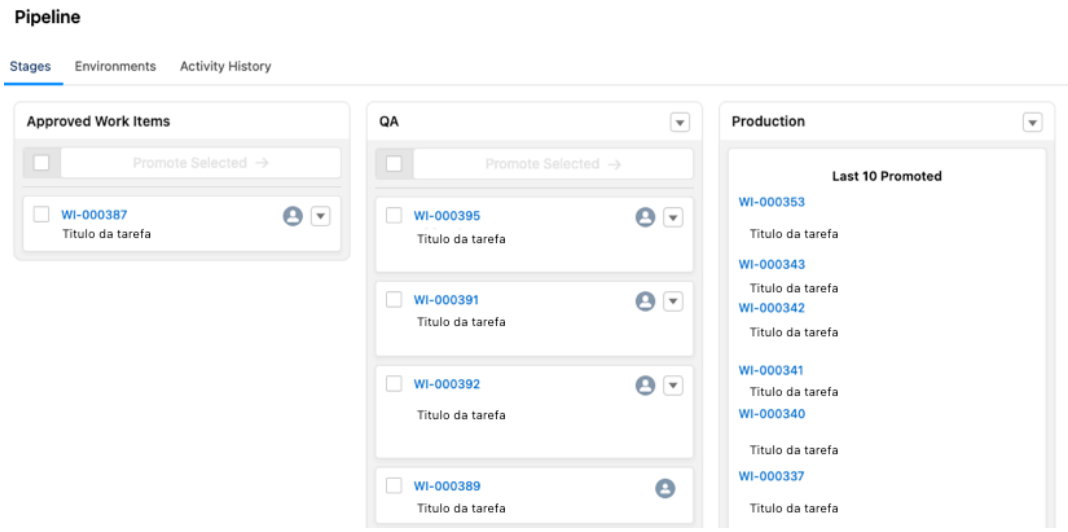

Figura 4.3: Esta figura representa pipeline com os diversos ambientes no Devops.

4.5.4 Desenvolvimento de uma funcionalidade exemplificativa

Para ilustrar a vertente técnica do desenvolvimento de uma funcionalidade em Salesforce será demonstrado um desenvolvimento exemplificativo. Este exemplo consiste na criação de uma funcionalidade de verificação de integridade de contactos e considera-se que um contacto está válido quando tem o último nome preenchido. Caso o mesmo não o tenha preenchido deve ser criado um *Log* que irá conter o erro (no caso será a falta de preenchimento do último nome) e o Identificador (Id) do respetivo contacto.

O exemplo será composto por um LWC que agrega dois componentes: um botão que, ao ser clicado, irá executar um *batch* Apex, e um texto que irá exibir a quantidade de contactos inválidos.

No *batch* Apex é executada uma *query* para obter, dos dados armazenados no Salesforce, o conjunto de contactos cujo último nome não esteja preenchido. Esta ação será realizada no método *Start*, que irá invocar uma função para retorna a *query* a executar, no formato de *String*. No excerto de código seguinte são apresentadas as funções *Start* e a função *getContactsQuery* que retorna a String mencionada.

```
public class InvalidContactsBatch implements Database.Batchable < sObject > {
    ...

    public Database.QueryLocator start(Database.BatchableContext BC) {
        return Database.getQueryLocator(getContactsQuery());
    }

    @AuraEnabled(cacheable=false)
    public static String getContactsQuery(){
        return "SELECT Id FROM Contact WHERE LastName = NULL";
    }

    ...
}
```

Listing 4.6: Trecho de código que mostra a função *LoadScene* utilizada para carregar uma nova cena

Posteriormente após obter os registos, os mesmos serão enviados para o método *Execute*, que itera sobre todos os contactos para criar um *Log* que irá conter o ID do contacto e uma mensagem de erro. Depois de todos os *logs* serem criados, serão inseridos no Salesforce de forma a poderem ser posteriormente consultados. No excerto de código seguinte será apresentada essa função.

```
public class InvalidContactsBatch implements Database.Batchable < sObject > {
    ...
    public void execute(Database.BatchableContext BC, List <Contact> scope) {
        List<IntegrationLog> logsToInsert = new List<IntegrationLog>();
        for(Contact contact : scope){
            IntegrationLog log = new IntegrationLog(
                ErrorMessage = 'Invalid Contact',
                ObjectId = contact.Id
            );

            logsToInsert.add(log);
        }

        insert logsToInsert;
    }
    ...
}
```

Listing 4.7: Trecho de código que mostra a função *LoadScene* utilizada para carregar uma nova cena

Por fim, no método *Finish* será exibida uma mensagem na consola que indica que o *batch* foi concluído. O excerto de código seguinte apresenta esse método.

```
public class InvalidContactsBatch implements Database.Batchable < sObject > {
    ...
    public void finish(Database.BatchableContext BC) {
        System.debug('Finish sending data to GC');
    }
    ...
}
```

Listing 4.8: Trecho de código que mostra a função *LoadScene* utilizada para carregar uma nova cena

Com o *batch* Apex construido será desenvolvido o LWC. Inicialmente o mesmo irá chamar a função *getContactsQuery*. De notar que para que o LWC possa invocar uma função do Apex a mesma deve estar anotada com ***@auraEnabled***. A função irá retornar uma *string* com a *query* a ser executada, que será utilizada no LWC por meio de outra função do Apex apresentada no excerto de código seguinte.

```
public class InvalidContactsBatch implements Database.Batchable < sObject > {
    ...
    @AuraEnabled(cacheable=false)
```

```
    public static List<Contact> executeQuery(String query) {
        return Database.query(query);
    }
    ...
}
```

Listing 4.9: Trecho de código que mostra a função *LoadScene* utilizada para carregar uma nova cena

A função que executa a *query* irá retornar para o LWC um conjunto de registos, cujo tamanho será verificado para exibir ao utilizador o número de contactos inválidos.

Após exibir essa informação, espera-se um clique no botão, que, ao ser acionado, chamará o *batch* Apex previamente construído. Para isso, é necessário criar uma função Apex que execute o *batch* e invocá-la no LWC da mesma forma como se chamou as funções anteriores.

4.5.5 Exemplo prático - Recuperações

As recuperações foi uma das funcionalidades desenvolvidas durante o estágio e consiste no cálculo e apresentação, para cada doador, do montante devido ao cliente pela agência de pagamentos.

Para o desenvolvimento desta funcionalidade foi atribuída uma US à estagiaria com o objetivo da funcionalidade e os respetivos critérios de aceitação.

Para fazer este cálculo pretendia-se o desenvolvimento de um *batch* que todos os dias a uma determinada hora calculasse o montante devido e atualizasse o registo do doador. Após a atualização, a ficha do doador deverá mostrar num campo de *input* não interativo o montante calculado seguido de uma *checkbox* marcada, também não interativa. Esta *checkbox* serve para indicar ao cliente que o calculo já foi efetuado para aquele doador.

Inicialmente, foi analisado um diagrama que acompanhava a US, o qual indicava a fórmula matemática a aplicar consoante o número de doações do doador. Por exemplo, se um doador realizou apenas uma doação, o montante correspondente a essa doação é integralmente atribuído ao cliente. No caso de duas doações, o cliente recebe 80% da média dessas doações e para três ou mais, o cliente recebe 70%.

Para realizar esse cálculo, foi elaborada uma *query* para selecionar do Salesforce todos os doadores que tenham pelo menos uma doação, e armazena o resultado numa lista. De seguida, foi feita uma segunda *query* para selecionar todas as doações cujo Id do doador esteja contido na lista de doadores. No excerto de código seguinte são apresentadas as duas *queries* previamente mencionadas.

```
List<Donor> donorIdsList = [SELECT Id FROM Donor WHERE numberOfDonations >= 1 AND
    checked = false];

List<Donation> listOfDonations = [SELECT Id, DonorId, Amount FROM Donation WHERE
    DonorId IN :donorIdsList];
```

Listing 4.10: Trecho de código que mostra a função *LoadScene* utilizada para carregar uma nova cena

Posteriormente agrupou-se num mapa as doações de cada doador. A chave do mapa são os Ids dos doadores e os seus valores consistem na lista das suas doações. Se o Id do doa-

dor já estiver presente no mapa, adiciona-se a doação à sua lista de doações, caso contrário, adiciona-se um novo elemento ao mapa contendo como chave o Id do doador e como valor uma lista com a doação em questão. O excerto de código seguinte demonstra o modo como foi construído o mapa.

```
Map<Id, List<Donation>> donationsByDonorId = new Map<Id, List<Donation>>();

for(Donation donation : listOfDonations){
    if(!donationsByDonorId.containsKey(donation.donorId)){
        donationsByDonorId.put(donation.donorId, new List<Donation>{donation});
    }else if(donationsByDonorId.get(donation.donorId).size()){
        List<Donation> donationsList = donationsByDonorId.get(donation.donorId);
        donationsList.add(donation);
        donationsByDonorId.put(donation.donorId, donationsList);
    }
}
```

Listing 4.11: Trecho de código que mostra a função *LoadScene* utilizada para carregar uma nova cena

Seguidamente, o mapa foi percorrido iterativamente. Em cada iteração, calculou-se o valor a recuperar com base no número de doações de cada doador. Criou-se um novo registo de doador e adicionou-se a uma lista que irá conter os dados dos doadores, com os valores calculados e o campo *checkbox* com valor *true*.

A atualização dos doadores no Salesforce foi feita com o uso da DML de atualização em lote. No fim da atualização, caso tenha ocorrido algum tipo de erro, é imprimida uma mensagem na consola com o erro ocorrido. O excerto de código seguinte ilustra o processo de atualização descrito.

```
List<Database.UpdateResult> results = Database.update( donorsToUpdate, false);

for(Database.UpsertResult result : results){
    if(!result.isSuccess()){
        System.debug('Error Updating Donor: ' + JSON.serializePretty(result.
            getErrors());
    }
}
```

Listing 4.12: Trecho de código que mostra a atualização dos doadores

4.5.6 Exemplo prático - Integração Agência

Para esta funcionalidade seguiu-se o mesmo processo de atribuição à estagiária de uma US. O objetivo seria criar membros a partir de doadores, resultando em novos registos que contêm informações dos doadores.

A estagiária foi desafiada a encontrar a melhor abordagem para esta tarefa. Inicialmente, considerou-se o desenvolvimento da funcionalidade utilizando um *Flow*. Um *Flow* no Salesforce é uma ferramenta de automação de processos que permite a criação de fluxos visuais para realizar tarefas específicas. Um exemplo simples de um *Flow* é ilustrado na fi-

gura 4.4, onde um conjunto de registos é recuperado com base numa condição e para cada registos é alterado um campo e por fim os registos no Salesforce são atualizados.

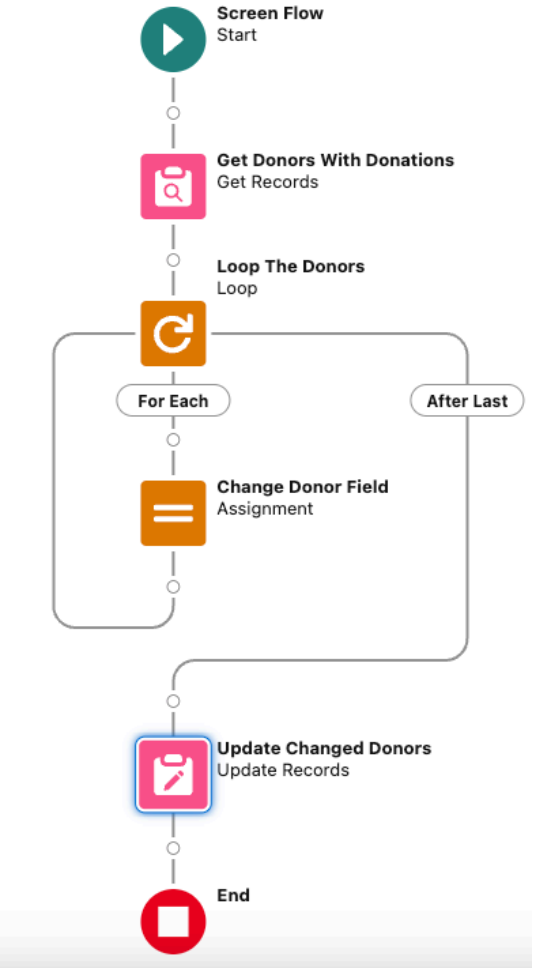

Figura 4.4: Esta figura representa um exemplo de um Flow Salesforce.

Embora o *Flow* parecesse uma solução viável, ele apresentava um limite de iteração de 2000 elementos. Portanto, caso houvesse mais de 2000 doadores para os quais os membros precisavam de ser criados, essa abordagem não seria adequada.

Após descartar a opção do *Flow*, optou-se por utilizar o *Batch* Apex como solução. O objetivo era executar automaticamente o processo agendando o *job* do *Batch* Apex ao usar o método *schedule*, selecionando todos os doadores que atendiam às condições necessárias para se tornarem membros.

Inicialmente, uma *query* foi elaborada para selecionar todos os doadores que atendiam aos critérios definidos. De seguida, os doadores selecionados foram iterados, e para cada um deles, um membro foi criado, contendo informações relevantes do registo do doador. Cada membro criado foi adicionado a uma lista para que pudessem ser inseridos no Salesforce em conjunto no final do processo.

Finalmente, foi criada uma função de *schedule* para agendar a execução do *job* do *Batch* Apex num momento futuro. Com essa função é possível especificar a data e a hora para iniciar a execução do *job*, garantindo assim uma execução controlada e planeada do processo. O excerto de código seguinte apresenta a função *schedule* na qual é chamado o *batch*. É im-

portante notar que o cabeçalho da classe do *batch* deve implementar a interface *Schedulable* de modo a que o método inicie a execução do *batch*.

```
public class CreateMembersBatch
            implements Database.Batchable<sObject>, Schedulable {

    public void execute(SchedulableContext sc) {
        Database.executeBatch( new CreateMembersBatch() );
    }
}
```

Listing 4.13: Trecho de código que mostra a função schedule utilizada para agendar a execução do batch

4.6 *Testes*

Nesta secção, será explorada a avaliação e validação das funcionalidades desenvolvidas. Os testes são uma parte fundamental do processo de desenvolvimento de *software* pois garantem que as funcionalidades produzem o resultado esperado e atendem aos requisitos estabelecidos. Aqui, serão explorados os diferentes tipos de testes, desde os testes unitários que verificam unidades individuais de código, até os testes de integração que garantem a interoperabilidade entre diferentes componentes do sistema.

4.6.1 O que são testes?

Os testes unitários referem-se à prática de escrever e executar testes para validar o comportamento esperado do código desenvolvido para uma determinada funcionalidade. No contexto da empresa, são realizados dois tipos de testes:

- Testes Unitários;
- Testes Manuais.

4.6.1.1 Testes Unitários

Os testes unitários são uma prática de desenvolvimento de *software* que visa testar unidades individuais de código de forma isolada, garantindo que cada uma delas opera conforme o esperado.

Esses testes são essenciais para assegurar que todos os componentes estejam integrados de maneira adequada e funcionem conforme o previsto. É recomendável que os testes cubram pelo menos 90% do código desenvolvido para uma determinada funcionalidade, garantindo assim que ela esteja pronta para avançar para o próximo ambiente. Esse valor é considerado suficiente pela equipa para garantir o funcionamento correto da funcionalidade.

Os testes unitários são escritos pelos desenvolvedores responsáveis pelas funcionalidades em Apex. Os mesmos conhecem o fluxo de trabalho esperado do código e sabem mais facilmente identificar os pontos críticos que precisam de uma cobertura de teste adequada.

Os resultados esperados de cada teste são assegurados com o uso da classe *Assert*. Os *Asserts* são empregues para verificar se as condições específicas são verdadeiras durante a execução do código. Se após algum desenvolvimento ou alteração um *Assert* falhar então é possível perceber que alguma das funcionalidades existentes não tem o comportamento pretendido e por esse motivo o desenvolvimento não pode ser entregue. Sem os testes estes erros não seriam detetados resultando em problemas e falhas no sistema.

Para além dos *asserts*, os testes unitários envolvem também o uso dos métodos *startTest* e *stopTest* da classe *Test*, responsáveis por definir o início e o fim de um bloco de código sob teste. Geralmente, esses métodos são utilizados para delimitar a execução de um *batch* ou de um método específico a ser testado. Além disso, os testes unitários incluem um método *init*, que é empregue para criar registos e configurar o contexto do teste antes da execução de cada um dos seus métodos.

Automatizados, esses testes podem ser facilmente executados com um simples clique num botão da Developer Console, conforme demonstrado na figura 4.5 pela marca 4. Na figura, é possível ainda observar que um marcador de verificação (marca 1) indica o sucesso do teste, enquanto uma cruz (marca 2) indica a falha do teste. Na marca 3, é possível verificar a cobertura dos testes para cada uma das classes, mostrando a percentagem e as linhas cobertas.

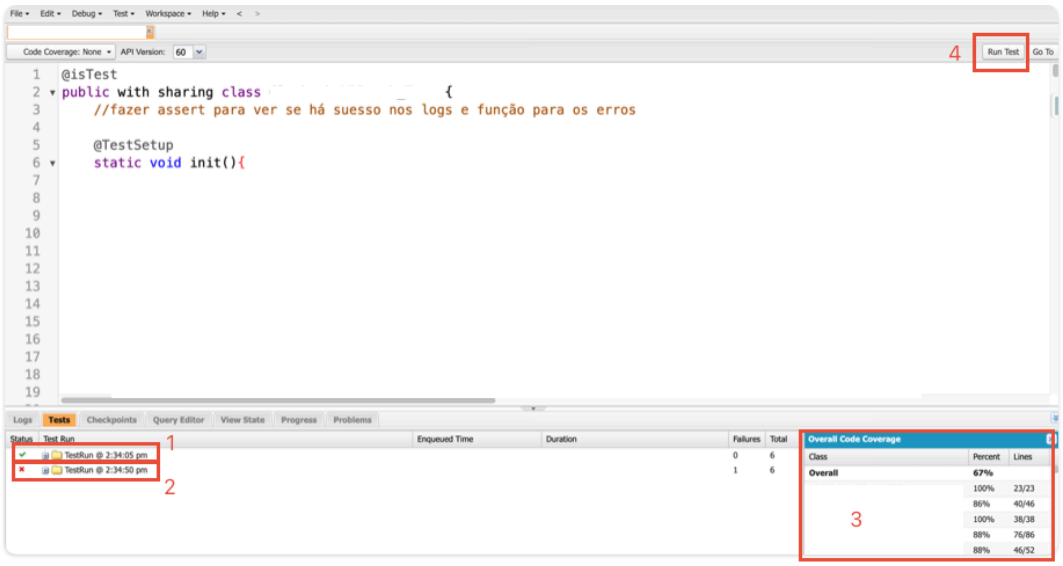

Figura 4.5: Esta figura o layout da Developer Console, usada para executar testes unitários.

O desenvolvimento de testes unitários é uma tarefa crucial, pois garante a qualidade do código e previne falhas que possam afetar as funcionalidades desenvolvidas. Além disso, os testes garantem o sucesso da implantação do código no ambiente de produção e servem como documentação, ao descrever o comportamento esperado de partes específicas do código.

4.6.1.2 Testes Manuais

Os testes manuais são uma etapa realizada após o desenvolvimento completo da funcionalidade e o seu sucesso nos testes unitários. Esses testes são conduzidos por membros da equipa, que, ao confirmarem o sucesso, repassam os testes para a equipa do cliente para

avaliação. Eles são executados manualmente, sem o auxílio de ferramentas automáticas, seguindo uma sequência de passos definidos.

Durante esse processo, os *testers* assumem o papel de utilizador final e exploram diferentes aspetos da funcionalidade, como desempenho e usabilidade. Os mesmos registam o comportamento do sistema, identificando possíveis falhas ou áreas de melhoria.

A realização desses testes permite que várias pessoas abordem a funcionalidade de maneiras diferentes, aumentando a probabilidade de detetar *bugs* e problemas de usabilidade.

Inicialmente os testes manuais são realizados pelo desenvolvedor na *org* de desenvolvimento. Quando o desenvolvimento é integrado na *org* de QA, é responsabilidade dos *testers* garantir a qualidade do desenvolvimento entregue. Após *feedback* positivo, a funcionalidade é então testada pelo cliente.

4.6.2 Exemplo Prático - Recuperações

Para testar a funcionalidade das recuperações foram realizados testes manuais e testes unitários.

Nos testes unitários, foram criados com recurso a operações DML três registos de doadores e seis registos de doações no método *init*. As doações foram associadas aos doadores de forma a que um doador tivesse uma doação, outro tivesse duas e o terceiro tivesse três. De seguida, foram criadas três classes de teste, cada uma com o objetivo de verificar se os campos dos doadores foram preenchidos corretamente após a execução do *batch*.
O teste inicia-se pela execução do *batch* responsável por calcular as Recuperações do desenvolvimento mencionada na secção 4.5.5. No fim da execução desse *batch*, foi selecionado, utilizando uma *query*, da lista de dados o doador cujo nome é "1Donation". Por fim, utilizando o método *areEqual* da classe *Assert*, validamos se o montante devolvido ao cliente tem valor 10 e se o campo ficou marcado como completo. O excerto de código seguinte representa a classe descrita.

```
@isTest
static void testAmount1Donation() {

    Test.startTest();
    Database.executeBatch(new CalculateBatch());
    Test.stopTest();

    Donor donor = [SELECT amount, isCompleted FROM Donor WHERE Name = '1Donation'
        LIMIT 1];

    Assert.areEqual(10, donor.amount);
    Assert.areEqual(true, donor.isCompleted);
}
```

Listing 4.14: Trecho de código que mostra uma classe de testes das recuperações

Nos testes manuais, o procedimento consistiu em criar três registos de doadores, cada um com um número específico de doações necessárias para completar todos os cálculos. De

seguida, executou-se o *batch* e os campos foram verificados para garantir que estavam preenchidos corretamente. Se os mesmos estivessem preenchidos corretamente, a funcionalidade passava nos testes manuais.

4.6.3 Exemplo Prático - Integração Agência

Para testar a funcionalidade da integração com a agência foram realizados testes manuais e testes unitários.

Nos testes unitários foi criado, com recurso a operações DML, um registo de doador com determinadas informações no método *init*. De seguida, foi criada uma classe de teste, com o objetivo de verificar se o membro foi criado após a execução do *batch*.

Inicialmente foi executado o *batch* para criar o membro a partir dos dados do doador. Seguidamente selecionou-se da lista de dados todos os ids dos membros existentes. Como durante este teste foi criado apenas um membro, se tudo funcionar corretamente espera-se que o número de membros existentes seja um. Esta verificação é feita com recurso ao método *areEqual* da classe *Assert*.

```
@isTest
static void testMembersCreation() {

    Test.startTest();
    Database.executeBatch(new CreateMembersBatch());
    Test.stopTest();

    List<Member> member = [SELECT Id FROM Member];

    Assert.areEqual(1, member.size());
}
```

Listing 4.15: Trecho de código que mostra uma classe de testes da integração com a agência

Para além da classe de teste mencionada, foi criada uma segunda classe com o objetivo de cobrir a parte do código responsável pelo agendamento do método *schedule* do *batch*. Para agendar a execução do *batch* primeiro é criada uma instância desse *batch* que será depois agendada com recurso ao método *schedule* da classe *System*. Este método recebe três parâmetros em que o primeiro parâmetro com a descrição do agendamento, um segundo parâmetro com a data e a frequência da execução deste agendamento e como terceiro parâmetro o *batch* a executar.

A estrutura da data e frequência consiste nos seguintes 6 campos:

- Segundos - Valor de 0 a 59;
- Minutos - Valor de 0 a 59;
- Horas - Valor de 0 a 23;
- Dia do Mês - Valor de 1 a 31;
- Mês - Valor de 1 a 12;

- Dia da Semana - Valor de 1 a 7.

Alguns valores podem assumir o símbolo * que significa que são executados repetidamente ou o símbolo ? que significa que podem assumir qualquer valor.

No caso do excerto de código seguinte, a estrutura da data e frequência é "0 0 * * * ?" que significa que este *batch* será executado aos 0 segundos e 0 minutos, de hora a hora, todos os dias, todos os meses e em qualquer dia da semana.

```
@isTest
public static void testBatchSchedule(){
    Test.startTest();
    CreateMembersBatch reminder = new CreateMembersBatch();

    System.schedule('Schedule is Running','0 0 * * * ?', reminder);
    Test.stopTest();
}
```

Listing 4.16: Trecho de código que mostra uma classe de testes que cobre a classe Schedule

Este teste tem como único objetivo garantir a cobertura de código do agendamento do *batch* mas este agendamento não é verdadeiramente executado pois está dentro do contexto de teste.

Nos testes manuais, o procedimento consistiu em criar um registo de doador elegível para servir de base à criação de um membro. De seguida, executou-se o *batch* e verificou-se se foi inserido um novo membro relacionado ao doador. Em caso de sucesso, a funcionalidade passava nos testes manuais.

4.7 *Entrega*

Nesta secção, serão explorados os passos para uma entrega eficiente e eficaz. Desde a preparação do ambiente de produção até à implementação das funcionalidades desenvolvidas, esta secção abrange todo o processo de entrega. Além disso, é também abordada a etapa de aprovação onde se garante que os desenvolvimentos atendem aos requisitos e expectativas do cliente antes de serem lançadas para produção.

4.7.1 Primeiros passos

Após a conclusão de uma tarefa, é importante realizar um *commit* final no WI criado no passo 4.5.2, como demonstrado na figura 4.6.

Inicialmente, devem ser selecionados todos os itens que foram criados e desenvolvidos (marca 1). Seguidamente, é necessário inserir uma breve descrição do que foi realizado (marca 2) e clicar no botão *Commit* (marca 3). Após o *commit* ser realizado, deve-se clicar no botão de *review* para que a funcionalidade seja aprovada (marca 4).

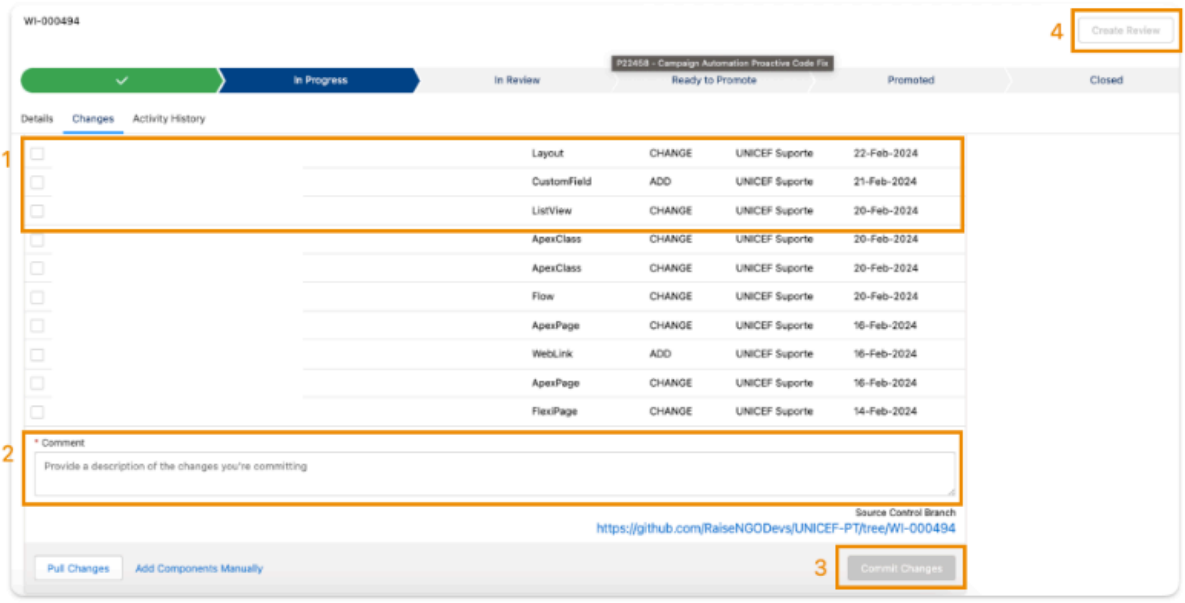

Figura 4.6: Esta figura representa os passos necessários a realizar para finalizar o *commit* da funcionalidade.

4.7.2 Aprovação

A aprovação da funcionalidade é conduzida pelo *Tech Lead*. Este é encarregado de rever o código desenvolvido, e assegurar que o código está em conformidade com as normas e funcional. Após essa revisão, o *Tech Lead* autoriza a promoção do WI e encaminha-o para o ambiente de QA, dando assim a tarefa como concluída.

Se a funcionalidade não for aprovada pelo *Tech Lead*, a sua integração é bloqueada. De seguida, é agendada uma reunião entre o desenvolvedor responsável pela tarefa e o *Tech Lead*, com o objetivo de identificar e esclarecer as melhorias necessárias a realizar.

4.7.3 Ambientes e integração

Inicialmente, a funcionalidade é encaminhada para o ambiente de QA, onde é submetida a testes rigorosos. Após a conclusão bem sucedida de todos os testes, a mesma irá ser enviada para o ambiente de produção.

Dado que novos desenvolvimentos (como campos), por padrão, não têm permissões ativas quando integrados no ambiente de produção, é crucial garantir que todas as permissões necessárias sejam concedidas. Esta atribuição de permissões garante que o cliente pode usufruir da funcionalidade sem quaisquer restrições.

Capítulo 5

Conclusão

5.1 Introdução

Na conclusão deste relatório, é possível destacar tanto as metas alcançadas ao longo do estágio quanto os desafios enfrentados durante o processo. Durante este período, a estagiária conseguiu atingir marcos importantes de aprendizagem e desenvolvimento ao consolidar o seu conhecimento na plataforma Salesforce e ao adquirir experiência valiosa no ambiente profissional.

É importante destacar que cada implementação é única e cada cliente apresenta novos desafios a serem superados. Por isso, é fundamental ter um bom conhecimento da ferramenta de trabalho, neste caso a plataforma Salesforce, para conseguir adaptar a solução às necessidades específicas do cliente, respeitando ao mesmo tempo os limites do sistema.

Ao analisar as metas alcançadas e os problemas enfrentados, é possível extrair lições valiosas e *insights* que contribuirão para o crescimento contínuo da estagiária e para o aprimoramento das práticas organizacionais.

5.2 Problemas Encontrados

Durante o estágio, a parte mais desafiante foi encontrar soluções que fossem eficientes, respeitando os limites do sistema e respondendo às necessidades do cliente. Foi muitas vezes necessário ver além daquilo que o cliente exigia e entender a fundo a funcionalidade necessária para colmatar essa demanda.

Durante o estágio, a parte mais desafiante foi encontrar soluções eficientes que respeitassem os limites do sistema e atendessem às necessidades do cliente. Compreender as necessidades reais dos clientes, que nem sempre eram claramente expressas, foi crucial para o sucesso das implementações.

Embora o desempenho do sistema fosse uma preocupação constante, influenciado por fatores como o volume de dados, a complexidade das consultas e as customizações, o verdadeiro obstáculo estava na adaptação às expectativas dos clientes. Frequentemente, as exigências dos mesmos apresentavam desafios técnicos significativos, exigindo um profundo entendimento das funcionalidades do Salesforce e uma habilidade de adaptar o sistema de maneira criativa e eficaz.

Em cada uma dessas situações, a colaboração com os colegas de equipa foi fundamental para encontrar soluções eficazes que permitissem alcançar os objetivos do projeto. O trabalho conjunto e a troca de ideias desempenharam um papel crucial na superação desses desafios e na entrega bem sucedida das soluções ao cliente.

5.3 Metas Alcançadas

Sendo o Salesforce uma poderosa plataforma de CRM, altamente customizável e escalável, a mesma torna-se uma ferramenta adequada para organizações e empresas de todos os tamanhos. O ecossistema do Salesforce oferece uma vasta gama de integrações e aplicações que ampliam as suas funcionalidades com apenas alguns cliques. O grande desafio está em conhecer a plataforma e adaptá-la às necessidades específicas de cada cliente.

Durante o estágio, a estagiária adquiriu um vasto conhecimento da plataforma Salesforce, alcançando um marco significativo na jornada de aprendizagem. A mesma ficou com uma base sólida ao envolver-se ativamente com as práticas e conhecimentos adquiridos. A mesma considera também que foi possível aplicar os conceitos teóricos e práticos aprendidos na universidade, trazendo assim valor na resolução de problemas e na discussão de funcionalidades.

Este período foi marcado pela exploração de novas metodologias, conceitos avançados e pela aplicação prática dessas habilidades. O conhecimento e a experiência com a metodologia Agile/Scrum é de extrema importância, pois a mesma permite uma gestão de projetos mais eficiente e adaptável às mudanças, aumentando a produtividade e a qualidade dos resultados entregues. Para além disso, sendo esta uma das mais utilizadas no mercado de trabalho [22] [23] a experiência com a mesma poderá vir a ter um grande valor.

Além disso, a estagiária teve a oportunidade valiosa de mergulhar no mundo profissional. Ao enfrentar desafios diários, a mesma descobriu o que significa estar no ambiente de trabalho. A experiência proporcionou uma compreensão mais profunda das dinâmicas corporativas, incluindo a interação com colegas, a gestão de prazos e a adaptação a novas situações.

Ao ter a oportunidade de trabalhar lado a lado com profissionais experientes, com muito conhecimento na área das ONG e na plataforma Salesforce, a estagiária adquiriu bastante conhecimento não só da plataforma Salesforce, como também da área das ONG, área essa de grande foco e extrema importância nos dias de hoje.

Por fim, esta experiência foi valiosa para conhecer mais a fundo uma das ferramentas líderes de mercado e desenvolver competências de interpretação, pensamento crítico e entendimento das necessidades dos clientes. Esta experiência traz valor não apenas para a implementação em questão mas para toda a futura experiência que a estagiária venha a ter.

Bibliografia

[1] R. N. Go, “Raise N’ Go,” 2024, [Online] https://raisengo.com/. 1

[2] Salesforce, “Data + AI + CRM + Trust = more sales and happier customers.” 2023, [Online] https://www.salesforce.com/eu/?ir=1. 1, 15

[3] UNICEF, “UNICEF,” 2024, [Online] https://www.unicef.pt/?altTemplate=StartView. 2

[4] C. Vermelha, “Cruz Vermelha,” 2024, [Online] https://www.cruzvermelha.pt/. 2

[5] Amnistia, “Amnistia,” 2024, [Online] https://www.amnistia.pt/. 2

[6] A. Salvador, “Associação Salvador,” 2024, [Online] https://associacaosalvador.com/. 2

[7] A. SOS, “Aldeias SOS,” 2024, [Online] https://www.aldeias-sos.org/. 2

[8] Salesforce, “Discovery 101,” 2023, [Online] https://www.amazon.com/Salesforce-Discovery-101-Discoveries-Projects/dp/B0BNJHYTYH. 5

[9] Adobe, “Waterfall Methodology: A Complete Guide,” 2023, [Online] https://business.adobe.com/blog/basics/waterfall. 9

[10] Asana, “What is Agile methodology?” 2023, [Online] https://asana.com/pt/resources/agile-methodology. 9

[11] Salesforce, “CRM: Definição e Conceitos.” 2023, [Online] https://www.salesforce.com/br/crm//#crm-definicao-e-conceitos-scroll-tab. 13

[12] ——, “Developer Console,” 2024, [Online] https://help.salesforce.com/s/articleView?id=sf.code_dev_console.htm&type=5. 16

[13] ——, “Salesforce DevOps,” 2024, [Online] https://developer.salesforce.com/developer-centers/devops. 17

[14] ——, “Visualforce,” 2024, [Online] https://help.salesforce.com/s/articleView?id=sf.pages_about.htm&type=5. 17

[15] ——, “What is Apex?” 2024, [Online] https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_what_is_apex.htm. 18

[16] T. Cursos, “O que é uma aplicação multitenant?” 2024, [Online] https://www.treinaweb.com.br/blog/o-que-e-uma-aplicacao-multitenant. 18

[17] Salesforce, “Introducing Lightning Web Components,” 2024, [Online] https://developer.salesforce.com/blogs/2018/12/introducing-lightning-web-components. 18

[18] V. S. Code, “Code editing Redefined.” 2024, [Online] https://code.visualstudio.com/. 19

[19] Trailhead, “Trailhead,” 2024, [Online] https://trailhead.salesforce.com/. 21

[20] Atlassian, “Agile,” 2024, [Online] https://www.atlassian.com/agile. 23

[21] ——, “Scrum,” 2024, [Online] https://www.atlassian.com/br/agile/scrum. 23

[22] Salesforce, “World’s 1 CRM,” 2024, [Online] https://www.salesforce.com/in/campaign/worlds-number-one-CRM/. 40

[23] ISDI, “Why is Salesforce the world’s number one CRM?” 2024, [Online] https://isdicrm.com/why-is-salesforce-the-worlds-number-one-crm/. 40

Glossário

LaTeX	Conjunto de macros para o processador de textos TeX, utilizado amplamente para a produção de textos matemáticos e científicos devido à sua alta qualidade tipográfica.