Relatório de Estágio

Jorge Alexandre Costa Martins

Curso Técnico Superior Profissional em Análise de Dados

set | 2023

GUARDA
POLI
TÉCNICO

POLI TÉCNICO GUARDA

Escola Superior De Tecnologia e Gestão

O CAMINHO PARA O DESENVOLVIMENTO EM SALESFORCE

RELATÓRIO DE ESTÁGIO
PARA OBTENÇÃO DO DIPLOMA DE TÉCNICO(A) SUPERIOR
PROFISSIONAL EM ANÁLISE DE DADOS

Jorge Alexandre Costa Martins
Setembro / 2023

POLI TÉCNICO GUARDA

Escola Superior de Tecnologia e Gestão

O CAMINHO PARA O DESENVOLVIMENTO EM SALESFORCE

RELATÓRIO DE ESTÁGIO
PARA OBTENÇÃO DO DIPLOMA DE TÉCNICO(A) SUPERIOR PROFISSIONAL EM ANÁLISE DE DADOS

Professor(a) Orientador(a): Paulo Alexandre Andrade Vieira

Jorge Alexandre Costa Martins

Setembro / 2023

POLI TÉCNICO GUARDA

Ficha Técnica

Estagiário

Nome: Jorge Alexandre Costa Martins

Aluno nº: 1705885

LinkedIn: https://www.linkedin.com/in/Jorge-a-c-martins

Curso: Técnico Superior Profissional (CTeSP) Análise de Dados

Estabelecimento de Ensino

Politécnico da Guarda, Escola Superior de Tecnologia e Gestão

Av. Doutor Francisco Sá Carneiro nº 50, 6300-559 Guarda, Portugal

Telefone: 271 220 100

Email: estg-geral@ipg.pt

Instituição de Acolhimento

Fidizzi, Lda

Rua da Sota 2A, 3000-392 Coimbra, Portugal

Telemóvel: +351 912 794 916

Email: hello@fidizzi.com

Website: https://fidizzi.com

POLI TÉCNICO GUARDA

Duração do estágio

Início: 15/02/2023

Fim: 30/06/2023

Orientador na ESTG

Nome: Paulo Alexandre Andrade Vieira

Grau Académico: Doutor em Informática e Automação

Tutor na Instituição de Acolhimento

Nome: Daniel Sousa Baptista

Grau Académico: Licenciatura em Matemática Aplicada

POLI TÉCNICO GUARDA

Resumo

O presente relatório descreve o decurso do estágio curricular do curso de Técnico Superior Profissional (CTeSP) em Análise de Dados da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda realizado na empresa FIDIZZI.

Este estágio teve como objetivo o estudo da plataforma de CRM Salesforce, de maneira que o estagiário conseguisse ter uma visão geral das funcionalidades e potencialidades desta plataforma, e a capacidade para implementar algumas soluções ao nível das funções de administrador, consultor e programador. Para tal, o estágio foi dividido em duas fases.

A primeira fase consistiu na aprendizagem da plataforma Salesforce, recorrendo à plataforma de ensino Trailhead onde foi seguido um roteiro de conteúdos (Trailmix) definido pela entidade acolhedora.

A segunda fase consistiu na implementação e apresentação de uma App Salesforce de gestão de eventos, onde foi posto em prática grande parte do conhecimento adquirido. Foram feitas configurações e definidas permissões de acesso aos dados. Foram também desenvolvidas funcionalidades e automações recorrendo a recursos *low-code*/*no-code* e implementadas soluções e testes em código Apex. No final foi feita a apresentação da App à entidade acolhedora.

Palavras-chave: Salesforce, Trailhead, CRM, App, Objetos, Apex, Triggers

POLI TÉCNICO GUARDA

Abstract

This report describes the course of the curricular internship of the Data Analysis Higher Professional Technical Degree of the School of Technology and Management of the Polytechnic Institute of Guarda carried out in the company FIDIZZI.

This internship aimed at training in the CRM Salesforce Platform, so that the intern could gain an overview of the platform's functionalities and potential, and the ability to implement some solutions at the administrator, consultant, and developer levels. To this end the internship was divided into two phases.

The first phase consisted of studying the Salesforce platform using the Trailhead learning platform, following a "content roadmap" (Trailmix) that was defined by the host organization.

The second phase had the objective of putting into practice the acquired knowledge by implementing a Salesforce App for event management and also make a presentation of the App. Configurations were made, data access permissions were defined, functionalities and automations were developed using low-code/no-code resources, and solutions and tests were implemented in Apex code. Finally, the App was presented to the host organization.

Keywords: Salesforce, Trailhead, CRM, App, Objects, Apex, Triggers

POLI TÉCNICO GUARDA

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

API: **A**pplication **P**rogramming **I**nterface (Interface de Programação de Aplicações)

App: **App**lication (Aplicação)

CEO: **C**hief **E**xecutive **O**fficer (Diretor executivo)

CRED: **C**reate, **R**ead, **E**dit, **D**elete (Criar, Ler, Editar, Eliminar)

CRM: **C**ustomer **R**elationship **M**anagement (Gestão da Relação com o Cliente)

CTeSP: **C**urso **Té**cnico **S**uperior **P**rofissional

DML: **D**ata **M**anipulation **L**anguage (Linguagem de Manipulação de Dados)

ER: **E**ntidade-**R**elacionamento

HTML: **H**yper**T**ext **M**arkup **L**anguage (Linguagem de Marcação de HiperTexto)

IDC: **I**nternational **D**ata **C**orporation

IDE: **I**ntegrated **D**evelopment **E**nvironment (Ambiente de Desenvolvimento Integrado)

IU: **I**nterface do **U**tilizador

LWC: **L**ightning **W**eb **C**omponents

OWD: **O**rganization **W**ide **D**efaults (Predefinições para Toda a Organização)

Org: Salesforce **Org**anization (Organização Salesforce)

SOQL: **S**alesforce **O**bject **Q**uery **L**anguage (Linguagem de Consulta de Objetos Salesforce)

SQL: **S**tructured **Q**uery **L**anguage (Linguagem de Consulta Estruturada)

TI: **T**ecnologia da **I**nformação

VCS: **V**ersion **C**ontrol **S**ystem (Sistema de Controlo de Versão

1. Introdução

A quantidade de dados gerados diariamente e a sua disponibilidade nunca foi tão grande, os avanços tecnológicos deram às empresas ferramentas e recursos poderosos, capazes de recolher e processar dados numa escala nunca antes vista e em tempo recorde. O grande desafio das empresas agora é conseguir transformar grandes quantidades de informação em insights de criação de valor [1]. Esta necessidade é verificada pelo aumento da procura de *software* corporativo especialmente *software* de Gestão de Relacionamento com o Cliente (em inglês *Customer Relationship Management* - CRM), uma vez que o foco nos clientes tem uma importância fundamental em ambientes de negócios digitais no destaque da concorrência [2]. Os *softwares* de CRM podem ajudar sincronizando as comunicações do cliente entre vários departamentos. Vendas, marketing e atendimento ao cliente podem estar todos em sintonia. Eles funcionam como um sistema de registo de contactos e contas ao longo do ciclo de vida do cliente. Com as ferramentas CRM as empresas podem acompanhar, automatizar, analisar e otimizar todas as interações com os clientes. Na sua essência, um CRM é uma base de dados de informações de contacto e histórico de interação para cada contacto individual [3].

Segundo a plataforma de avaliação de tecnologia empresarial TrustRadius[1], o *software* de CRM com maior quota de mercado é oferecido pela empresa Salesforce [4]. Um estudo do IDC[2] para a Salesforce prevê a criação de 9.3 milhões de novos postos de trabalho (diretos e indiretos) até 2026 [5], assim sendo, a formação e certificação na utilização desta plataforma será uma mais-valia.

[1] A TrustRadius (https://www.trustradius.com/) é uma plataforma que disponibiliza avaliações e classificações verificadas de diferentes tipos de produtos.

[2] O IDC (International Data Corporation) é uma empresa que fornece informações de mercado, serviços de consultoria e eventos para os mercados das tecnologias de informação, telecomunicações e tecnologias de consumo (https://www.idc.com/).

1.1 Enquadramento

Este relatório descreve o estágio curricular do Curso Técnico Superior Profissional (CTeSP) de Análise de Dados do Politécnico da Guarda, ministrado entre os anos letivos 2021/22 e 2022/23.

Esta componente curricular foi realizada em contexto de trabalho, em horário laboral entre o dia 15 de fevereiro e o dia 30 de junho, perfazendo 750 horas na empresa FIDIZZI em contexto remoto.

1.2 Caracterização Instituição de Acolhimento

A instituição acolhedora do estágio foi a FIDIZZI (https://fidizzi.com), fundada em 2018, esta empresa com sede em Coimbra é parceira da Salesforce e presta serviços e consultoria em Tecnologias de Informação (TI). Tem como missão ajudar os clientes a maximizar os seus investimentos em Salesforce e em 2022 teve um volume de negócios de cerca de 335.000€.

A FIDIZZI é uma microempresa que funciona em modo de trabalho remoto, é feita uma reunião diária *on-line* com toda a equipa e, caso haja necessidade, os seus colaboradores reúnem sempre que necessário também de maneira *on-line*. Uma vez por mês há uma reunião presencial no seu escritório em Coimbra. A empresa conta com uma equipa de profissionais especializados e credenciados em Salesforce e é composta por um arquiteto Salesforce que é também o CEO, dois programadores e seis consultores, que desenvolvem soluções que passam principalmente pela Salesforce *Cloud*, *Service Cloud*, *Experience Cloud*, *Revenue Cloud* e *Marketing Cloud*. A figura 1 mostra a estrutura organizacional da empresa.

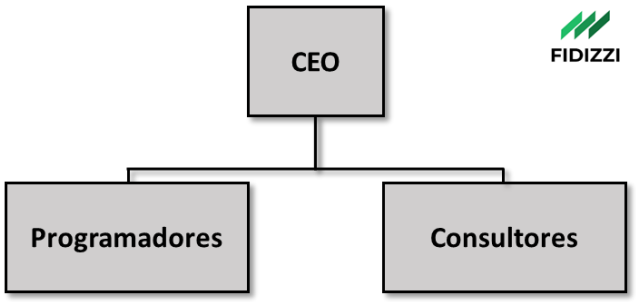


Figura 1 - Organograma da FIDIZZI.

Os serviços oferecidos pela FIDIZZI passam pela consultoria, implementação, customização, manutenção e desenvolvimento de Apps Salesforce para vários tipos de indústria, formação na plataforma de utilizadores finais e também oferece soluções a nível da migração e integração de sistemas *on-premise* (instalados nos servidores dos clientes) ou alojados em *cloud*.

1.3 Objetivos

A primeira fase do estágio teve como objetivo a aprendizagem da plataforma Salesforce usando a plataforma de ensino Trailhead, seguindo um roteiro de conteúdos teórico-práticos selecionados pela empresa, divididos em três temas, administrador, consultor e programador pretendendo assim dar ao estagiário uma visão geral das funcionalidades e aplicações deste *software* de CRM. Os objetivos para a segunda fase foram a realização e apresentação de um projeto final que consistiu no desenvolvimento de uma aplicação (App) Salesforce de gestão de eventos. A implementação desta App seguiu os requisitos definidos pela empresa. Com esta implementação pretendeu-se consolidar os conteúdos teórico-práticos da fase de aprendizagem, bem como desenvolver competências a nível da capacidade de pesquisa, aspeto fundamental nesta área. Outras competências como autonomia, capacidade de resolução de problemas, comunicação e organização foram também adquiridas durante o estágio.

Os conhecimentos adquiridos ao longo do curso de Análise de Dados foram importantes para a realização deste estágio uma vez que forneceram conceitos basilares que permitiram alcançar os objetivos do trabalho.

1.4 Estrutura do documento

O presente relatório é composto por nove capítulos, no primeiro capítulo é feita uma breve introdução sobre *software* CRM e, em particular, a solução oferecida pela empresa Salesforce. É também feita a contextualização do relatório, a caracterização da instituição de acolhimento e a exposição dos objetivos do trabalho.

No segundo capítulo é feito o estado da arte. Descreve-se a plataforma Salesforce, apresentam-se outras plataformas CRM e são analisadas as quotas de mercado.

No terceiro capítulo é abordada a metodologia de trabalho, apresentando diagramas de Gantt para as duas fases do estágio.

O quarto capítulo apresenta as tecnologias usadas ao longo do estágio.

O quinto capítulo corresponde à primeira fase do estágio, é feita uma descrição em traços gerais a plataforma de aprendizagem Trailhead e os principais temas abordados em cada etapa (Introdução, administrador, consultor e programador).

Para a segunda fase do estágio, uma vez que tem um carácter mais específico, foram dedicados três capítulos onde é exposto com mais detalhe o trabalho realizado. Assim, no sexto capítulo refere-se o projeto final, nomeadamente à fase de configuração da App Salesforce a implementar, o sétimo capítulo é referente à implementação de funcionalidades da App e o oitavo capítulo corresponde à apresentação da App à entidade acolhedora.

No nono capítulo são apresentadas as apreciações finais sobre o trabalho realizado e sugestões de trabalho futuro.

2. Estado da Arte

2.1 Salesforce

Salesforce é uma empresa americana fundada em 1999 com sede em São Francisco, oferece soluções tecnológicas em *cloud* principalmente centradas na Gestão de Relações com o Cliente (CRM), permitindo que as empresas tenham uma visão completa de seus clientes. Quando a tecnologia Salesforce é implementada, os funcionários dos departamentos de marketing, vendas, atendimento ao cliente e tecnologia da informação (TI), entre outros, podem partilhar uma única visão do cliente em qualquer dispositivo e em qualquer parte do mundo, desde que possuam uma ligação à internet.

As ferramentas e serviços Salesforce são comumente referidos como Clouds. Como se pode ver na figura 2, a plataforma oferece seis tipos principais de *clouds*: *Sales Cloud*, Marketing *Cloud*, *Commerce Cloud*, *Service Cloud*, *Experience Cloud* e *Analytics Cloud*.

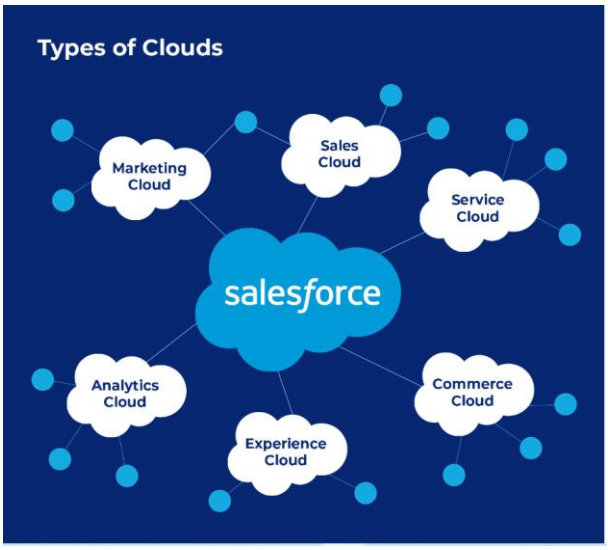


Figura 2 - Tipos de *Clouds* do Salesforce.

Fonte: https://www.scalefocus.com/blog/6-types-of-salesforce-clouds.

Estas *clouds* permitem a criação de Apps usando modelos pré-estabelecidos (*templates*) ou criando novos modelos. Mediante o tipo de negócio, é usada a *cloud* que mais se ajusta aos objetivos do cliente.

É possível desenvolver Apps usando programação com linguagens imperativas, mas é também possível construir e configurar Apps em *low-code*/*no-code* usando ferramentas *point-and-click* ou *drag-and-drop*, o que permite que utilizadores sem conhecimento profundo em programação possam desenvolver soluções.

Uma grande vantagem da Salesforce diz respeito à aprendizagem. É disponibilizada de forma gratuita uma plataforma *on-line* de aprendizagem - Trailhead - onde se pode adquirir conhecimento das várias funcionalidades e opções disponíveis bem como ter acesso a documentação e vídeos tutoriais mais técnicos sobre cada tema. O Trailhead permite também o acesso à comunidade de utilizadores (Trailblazers) que interagem entre si colocando questões e oferecendo respostas.

O Salesforce não tem versões de produção gratuitas, mas oferece uma versão de teste por trinta dias, contudo tem algumas limitações. Existem vários tipos de versões disponíveis que podem agrupar ou não diferentes *clouds*, cada versão tem vários níveis de licenças (Starter, Professional, Enterprise por exemplo) dependendo das funcionalidades que se pretendam e que influenciam o preço.

2.2 Outros softwares CRM

Existem *software*s de CRM gratuitos ou versões "*freemium*" disponíveis como o Zoho CRM (zoho.com) ou o Odoo (odoo.com), que podem ser uma boa escolha para novos utilizadores se familiarizarem com o uso de um sistema CRM. Existem, claro, desvantagens destas plataformas pois terão funcionalidades mais limitadas e, por vezes, um limite para o número de registos de contactos ou de utilizadores da plataforma. Normalmente têm menos integrações nativas (se as houver) com sistemas de terceiros. Estas plataformas são mais adequadas para pequenas empresas ou *startups*, especialmente se houver a opção de atualizar para uma versão paga à medida que vão crescendo [3].

De acordo com um estudo realizado pela TrustRadius (2021) referente a quotas de mercado dos CRMs mais usados (figura 3), verificou-se que os mais representativos eram o Salesforce (salesforce.com), seguido pela Microsoft Dynamics (dynamics.microsoft.com), Zoho CRM e Hubspot CRM (hubspot.com) [4].

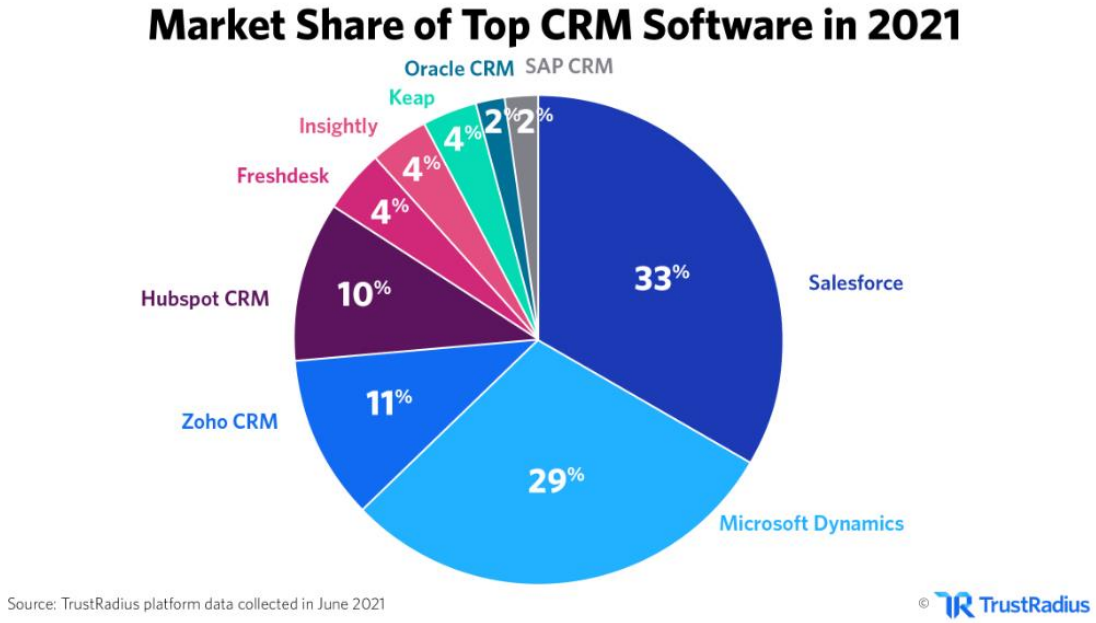


Figura 3 - Quotas de mercado de diferentes CRMs.

Fonte: https://www.trustradius.com/vendor-blog/crm-statistics-trends.

Num artigo de 2020 pela empresa Liminal[3] onde foi feita a comparação das soluções disponibilizadas pela Salesforce e pela Microsoft (Tabela 1), pode verificar-se que as funcionalidades dos dois produtos são semelhantes. Existe uma vantagem do Microsoft Dynamics em poder ser disponibilizado não só em cloud mas também como uma plataforma *on-premise* (instalado em servidores dos clientes) ou híbrida. Ambos permitem integração de ferramentas da Microsoft, mas no caso do produto da Microsoft esta integração é nativa tendo por isso óbvia vantagem. No entanto, o Salesforce é mais indicado na integração com tecnologias externas [6].

[3] A Liminal (https://liminal.pt) é uma empresa especializada em Tecnologias de Marketing.

Tabela 1 - Comparação entre Microsoft Dynamics e Salesforce.

	MICROSOFT DYNAMICS 365 SALES	SALESFORCE
Pricing	De 54,80€ a 130€/user/mês	De 22€ a 270€/user/mês
Reporting/Analytics	✓	✓
Contas & Contactos	✓	✓
Lead Management	✓	✓
Sales forecasting	✓	✓
Marketing Automation	Necessita de um add-on	Necessita de um add-on
Social CRM	✓	✓
Gestão de território	✓	✓
Sales performance management	✗	✓
Customer self-service portal	✓	✓
App Marketplace	✓	✓
Conferencing/IM	✓	✓
Gestão de parceiros	✗	✓
Workflows personalizados	✓	✓
Tracking de tempo	✗	✓
Cloud-based	✓	✓
On-premise	✓	✗
App nativa (mobile)	✓	✓
Controlo de acessos	✓	✓
API	Web Services API	SOAP API
Suporte	✗	✓

Fonte: https://liminal.pt/martech-magazine/salesforce-vs-microsoft-dynamics-365-sales-crm.

3. Metodologia

A metodologia de gestão de projetos é uma combinação estritamente definida de práticas relativas à lógica, aos métodos e aos processos que determinam a melhor forma de planear, desenvolver e controlar um projeto ao longo do processo contínuo da sua implementação e conclusão [7].

Para responder à ineficiência dos processos tradicionais de desenvolvimento de *software*, nomeadamente a dependência de extensa documentação e supervisão, surge em 2001 o Manifesto Ágil, elaborado por dezassete programadores de renome, com o objetivo de agilizar processos e ajudar a incentivar práticas de trabalho mais focadas no utilizador [8].

Este manifesto estabelece quatro valores orientadores [9]:

- As pessoas e as relações são mais importantes que os procedimentos e ferramentas;
- O funcionamento do *software* é mais importante do que ter a documentação completa;
- Colaborar com o cliente é mais importante do que negociar contratos;
- A capacidade de responder a alterações é mais importante do que seguir um plano.

O Manifesto Ágil enuncia também doze princípios [9]:

- Garantir a satisfação do cliente através da entrega rápida e contínua de *software* funcional;
- Mudanças ao âmbito do projeto são bem-vindas mesmo que cheguem em fases avançadas no desenvolvimento;
- Entregar *software* funcional com frequência;
- Deve haver cooperação constante entre clientes e a equipa;
- Os projetos devem ser criados em torno de indivíduos motivados, isto é, se a equipa tiver o ambiente e o apoio de que necessitam, ela vai conseguir atingir os objetivos;

- O método mais eficiente e eficaz de comunicar para a equipa e dentro dela é a conversa cara-a-cara;
- Ter um *software* funcional é a principal medida de progresso do projeto;
- Os processos ágeis promovem o desenvolvimento sustentável. Ou seja, os patrocinadores, programadores e utilizadores devem ser capazes de manter um ritmo constante indefinidamente;
- A atenção contínua à excelência técnica e ao bom *design* aumenta a agilidade;
- Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial;
- As melhores arquiteturas, requisitos e projetos emergem de equipas auto-organizadas;
- A equipa analisa regularmente como pode tornar-se mais eficaz, ajustando assim o seu comportamento.

Ao contrário de metodologias tradicionais, a metodologia ágil não tem como base a assunção de que os requisitos representam a restrição fixa a respeitar e que a partir desta restrição são estimados os recursos e o tempo necessários para desenvolver as funcionalidades (requisitos). São antes os recursos e o tempo destinados ao projeto que compõem as restrições fixas, estando os requisitos dependentes destes dois fatores [10].

No contexto da metodologia ágil a abordagem mais utilizada tem por nome *Scrum* e define uma estratégia flexível e de fácil implementação no desenvolvimento de produtos e serviços. Esta abordagem distingue-se principalmente por [11]:

- Em vez de uma abordagem sequencial, a equipa desenvolve os produtos progressivamente, sendo posteriormente melhorados de forma iterativa (*sprints*) e incremental;
- Ao invés de uma estrutura hierarquizada, as equipas auto-organizam-se;
- As equipas trabalham no mesmo local físico.

A figura 4 mostra a estrutura simplificada da metodologia *Scrum* que consiste em vários processos que vão sendo executados até ao fim do projeto [12]:

- O *Product Backlog* - é uma lista, ordenada por nível de prioridade, que contém os requisitos que se pretendem implementar, esta lista pode ser aprimorada no decurso do projeto;
- *Sprint* - é o trabalho realizado num intervalo de tempo definido que engloba vários componentes:
    - *Sprint Planning* - é o início do *Sprint* e consiste numa reunião de planificação do mesmo dando resposta a três questões fundamentais:
        - qual o objetivo a atingir no *Sprint* (*Sprint Goal*)
        - que itens do *Backlog* se vão incluir no *Sprint*
        - Como vai ser feito o trabalho selecionado;
    - *Sprint Backlog* - é o plano resultante do passo anterior contendo as respostas às três questões;
    - *Daily Scrum* - é uma breve reunião diária ao longo de todo o *Sprint* que tem como propósito verificar o progresso do trabalho em função dos objetivos traçados, identificar impedimentos que possam existir e, caso seja necessário, reajustar o *Sprint Backlog*;
    - *Increment* - é o resultado do trabalho feito no *Sprint* somado com todo o trabalho feito em sprints anteriores devidamente testado, garantindo que todos os Incrementos funcionam em conjunto;
    - *Sprint Review* - é uma reunião com o objetivo de mostrar o trabalho realizado no *Sprint* e determinar futuros ajustamentos;
    - *Sprint Retrospective* - é o momento final do *Sprint* onde é discutido o que correu bem, o que pode ser melhorado e que ações se podem tomar para melhorar.

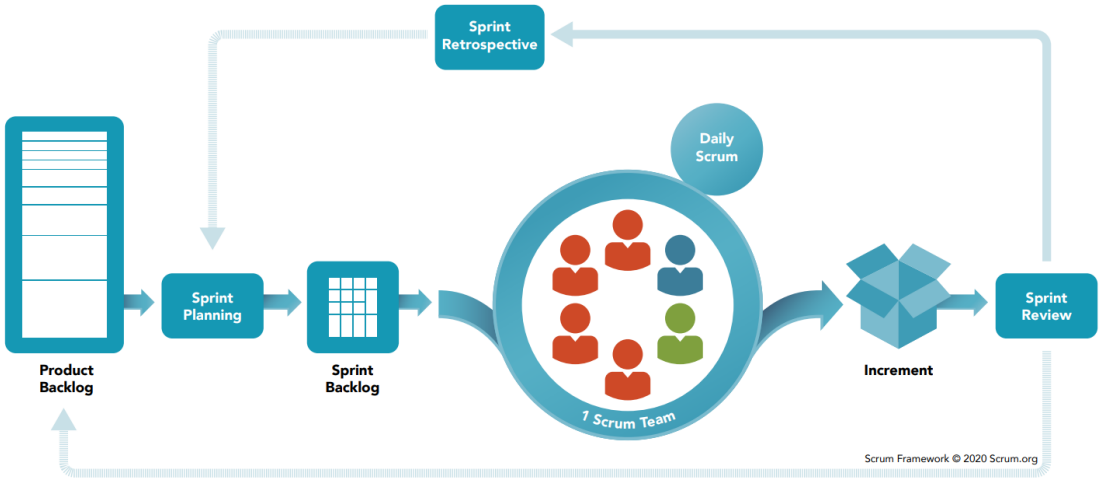


Figura 4 - Estrutura da metodologia *Scrum.*

Fonte: https://www.scrum.org/learning-series/what-is-scrum.

Assim sendo, num paradigma de desenvolvimento ágil os custos do desenvolvimento, o risco do projeto e até o tempo que leva à introdução dos produtos no mercado são reduzidos substancialmente [10].

3.1 Metodologia Usada

A metodologia usada no estágio foi uma adaptação da metodologia ágil *Scrum*.

Os objetivos de alto nível foram estipulados pelo tutor da entidade acolhedora e dividiram-se em duas fases, a primeira consistia na aprendizagem da plataforma Salesforce e a segunda passava pela criação e apresentação de uma App Salesforce. Para a primeira fase houve uma primeira reunião onde foram apresentadas as metas a atingir, foi elaborado um roteiro (Trailmix) para seguir na plataforma de aprendizagem Trailhead (https://trailhead.salesforce.com) de acordo com um cronograma com quatro etapas (*Sprints*). Durante esta fase houve reuniões três vezes por semana com o tutor onde era feito o acompanhamento da evolução do estagiário em função dos objetivos traçados. Eram também expostas as dificuldades e dúvidas que o estagiário ia encontrando e eram feitos ajustamentos ao calendário caso fosse necessário, nomeadamente, o alargamento

do tempo de estudo em determinados assuntos ou mesmo a eliminação de conteúdos que constavam no roteiro inicial. No final de cada etapa era feita uma reunião (*Sprint Review*) onde eram discutidos em termos gerais os conteúdos estudados e aferia-se a necessidade de fazer ajustamentos ao Trailmix ou ao calendário. No fim desta primeira fase houve uma reunião onde foi feito um balanço de todas as etapas (Sprint Retrospective).

Na segunda fase do estágio foram apresentados os requisitos para a implementação de uma App Salesforce (*Product Backlog*), mediante estes requisitos foram definidas as etapas (*Sprints*) a cumprir e a sua calendarização. Cada etapa tinha o seu objetivo e o conjunto de itens dos requisitos que seriam implementados (*Sprint Backlog*). Apesar de não haver reuniões diárias (*Daily Scrums*), o trabalho foi acompanhado com reuniões duas vezes por semana com o tutor e, sempre que necessário e se houvesse disponibilidade, com os programadores da empresa. Durante esta fase houve a necessidade de alterar os requisitos da implementação de modo a cumprir o calendário do estágio. A última etapa desta fase foi a apresentação da App, mostrando todas as configurações e funcionalidades implementadas, para isso foi feita uma apresentação em vídeo.

No final da segunda fase foi feita uma reunião de balanço onde o estagiário pôde fazer uma reflexão sobre os aspetos que correram bem e onde poderia melhorar. Neste momento foi sugerido um plano de continuação de estudos pós-estágio, por parte do tutor, visando a obtenção de certificação de programador (*developer)* em Salesforce.

3.2 Etapas de Desenvolvimento

A figura 5 mostra o diagrama de Gantt com as etapas da fase de aprendizagem da plataforma Salesforce, a data de início e o número de dias dispendidos.

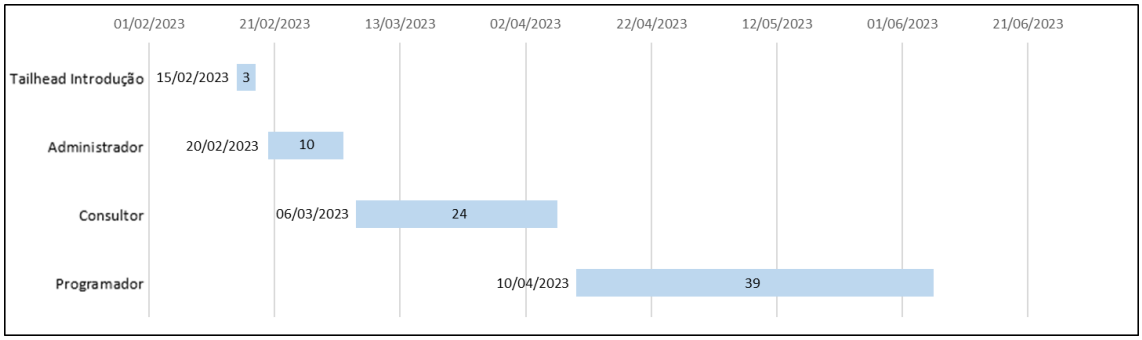


Figura 5 - Etapas da fase de aprendizagem da plataforma Salesforce.

Na figura 6 estão definidas as etapas da fase de desenvolvimento da App num diagrama de Gantt exibindo a data de início e o número de dias dispendidos.

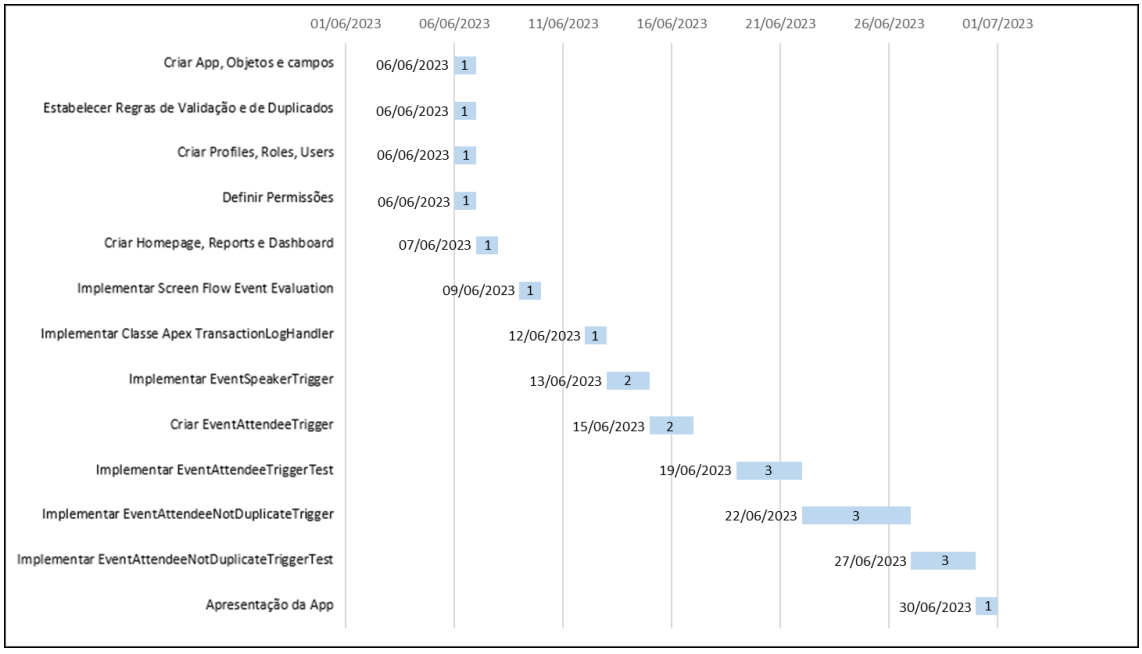


Figura 6 - Etapas da fase de desenvolvimento da App Salesforce.

4. Tecnologias Usadas

Trailhead

É uma plataforma de aprendizagem da Salesforce (https://trailhead.salesforce.com) onde são disponibilizados gratuitamente, mediante a criação de uma conta, conteúdos teórico-práticos que estão agrupados por temas.

Salesforce Developer Edition Org

Em Salesforce uma Org (abreviação de *Organization*) é o espaço virtual dedicado a um cliente individual do Salesforce onde são alojados todos os seus dados e Apps, cada Org está separada de todas as outras [13]. Em termos gerais, existem duas categorias de Orgs, são elas: instâncias de produção e instâncias de desenvolvimento [14]. A Developer Edition Org (figura 7) é uma instância de desenvolvimento gratuita e permite, por exemplo, desenvolver, testar e implementar aplicações, criar código Apex, páginas Visualforce [15].

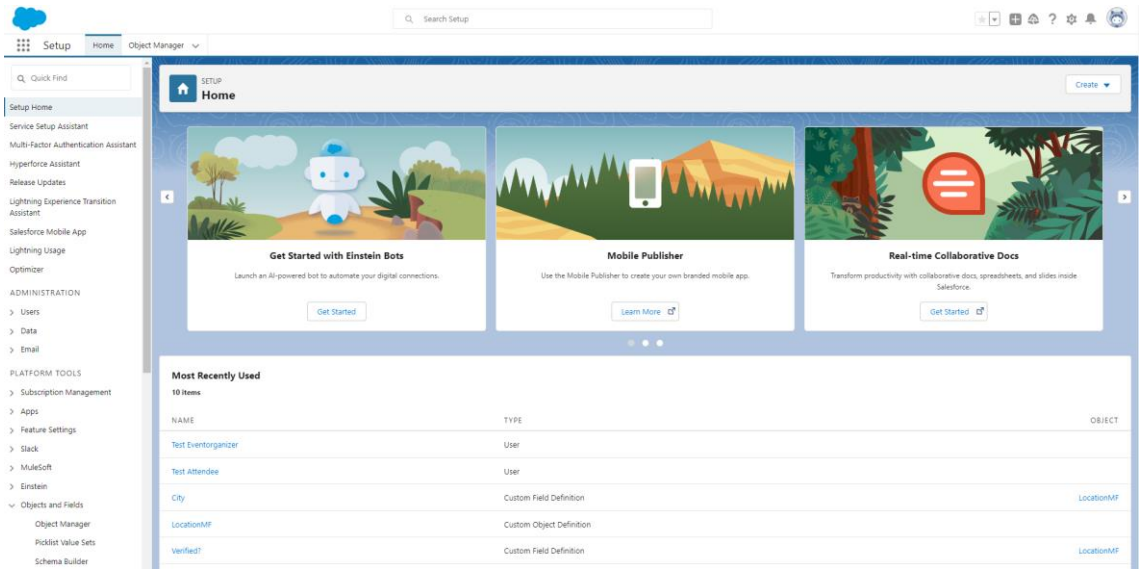


Figura 7 - Interface de utilizador de uma Org Salesforce na página de *Setup*.

SOQL

É uma linguagem de consulta nativa do Salesforce (*Salesforce Object Query Language*) que permite pesquisar os dados nas Orgs de Salesforce [16] muito semelhante a SQL (*Structured Query Language*).

Apex

É uma linguagem de programação orientada a objetos nativa do Salesforce semelhante a Java que permite executar instruções de controlo de fluxo e transações em servidores Salesforce em conjunto com chamadas para a API. O código Apex pode ser iniciado por pedidos ao serviço Web e por *triggers* (código Apex que é acionado quando se executa uma operação de manipulação de dados como inserção, edição ou eliminação) [17].

Developer Console

É o ambiente de desenvolvimento integrado (IDE) nativo do Salesforce, possui um conjunto de ferramentas usadas para criar, depurar, testar aplicações e executar consultas aos dados nas Orgs de Salesforce [18].

Microsoft Visual Code

É o editor de código-fonte recomendado para programadores de Salesforce, é gratuito e *open-source*, desenvolvido para Windows, Linux e macOS. A Salesforce disponibiliza gratuitamente extensões (Salesforce Extension Pack) para este Ambiente de Desenvolvimento que simplificam a experiência de programação [19].

Salesforce CLI

É uma interface de linha de comando, gratuita e *open-source*, que permite simplificar o desenvolvimento quando se trabalha numa Org de Salesforce [20]. Ela é usada para criar ambientes para desenvolvimento e teste, sincronizar código-fonte entre as Orgs e Sistemas de Controlo de versões (VCS), permite também executar testes e controlar o ciclo de vida das aplicações de Salesforce [21].

Visualforce

É uma estrutura de desenvolvimento *web* que permite a construção de interfaces de utilizador personalizadas para Apps móveis e de *desktop* na plataforma Salesforce [22]. Esta estrutura inclui uma linguagem de marcação baseada em *tags* semelhante a HTML e possui um conjunto de controladores *standard* do lado do servidor que simplificam as operações básicas da base de dados, como consultas e inserções [23].

JavaScript

É uma linguagem de programação responsável pela parte gráfica de aplicações *web*. Uma característica importante desta linguagem é que ela corre no lado do cliente (no *browser*) o que se traduz na poupança de recursos no lado dos servidores [24]. Esta linguagem é usada em *Lightning Web Components* (LWC) que é uma estrutura da interface do utilizador usada para criar páginas customizadas e funcionalidades na plataforma Salesforce.

HTML

É a linguagem de marcação *standard* usada para a criação de páginas *web*, ela descreve a estrutura das mesmas. Consiste em vários elementos que indicam ao *browser* como exibir o conteúdo das páginas [25]. Esta linguagem é também usada em LWC na criação de páginas customizadas e funcionalidades na plataforma Salesforce mas pode também ser embebida em código Apex em métodos que o permitam.

5. Aprendizagem da Plataforma Salesforce (1ª fase)

A fase de aprendizagem do Salesforce foi baseada na plataforma Trailhead que é disponibilizada gratuitamente pela Salesforce. Nela são disponibilizados conteúdos teórico-práticos que podem estar agrupados em trilhas, módulos ou projetos. Cada trilha tem vários módulos e/ou projetos, cada módulo está dividido em unidades e cada projeto está dividido em passos (figura 8). Existe também outra categoria, superbadges, que consistem em desafios práticos aplicados a problemas empresariais mais complexos.

Figura 8 - Organização dos conteúdos na plataforma Trailhead.

A entidade acolhedora fez uma seleção de trilhas, módulos, projetos e superbadges, de modo que o estagiário pudesse ter uma visão geral da plataforma na perspectiva de administrador, consultor e programador. A esta seleção chama-se, genericamente, Trailmix e, mediante a criação de uma conta na plataforma, pode ser acedida em: https://trailhead.salesforce.com/users/dbaptista/trailmixes/fidizzi-shark-bootcamp-extra.

No fim de cada unidade há um momento de avaliação, este pode ser um pequeno questionário ou um desafio prático onde o utilizador é guiado passo a passo. A avaliação tem de ser superada e no final é atribuída uma pontuação. No caso dos projetos, cada

passo implementado dá direito a uma pontuação, e só se pode avançar para o passo seguinte concluindo o passo anterior. Os desafios práticos e projetos são realizados numa *Developer Edition* Org especial - Trailhead Playground - que permite ganhar experiência na plataforma pois possui os recursos e funcionalidades do Salesforce além de um conjunto de dados de treino.

Quando se ultrapassam todas as unidades de um módulo ou todos os passos de um projeto, o Trailblazer (utilizador) ganha um emblema. A conjunção da pontuação com o número de emblemas ganhos estabelece o *rank* do Trailblazer numa escala que começa em "Scout" e termina em "All Star Ranger".

Esta fase do estágio foi dividida em quatro etapas (tabela 2), em seguida é feito um resumo dos temas que foram abordados.

Tabela 2 - Etapas da fase de aprendizagem da plataforma Salesforce com o número de trilhas, módulos, projetos, unidades/passos e superbadges.

Etapa	Trilhas	Módulos	Projetos	Unidades/Passos	Superbadges
Salesforce Introdução	-	11	-	13	-
Administrador	2	13	3	68	-
Consultor	4	22	21	172	4
Programador	2	16	10	119	2
Total	**8**	**62**	**34**	**372**	**6**

A primeira etapa teve o objetivo de dar uma visão resumida das plataformas Salesforce e Trailhead, do ecossistema da Salesforce e das funções de administrador, consultor e programador.

Na segunda etapa o estudo passou pela arquitetura do Salesforce, configurações, modelo de dados, conceitos básicos como objetos (equivalentes a entidades no modelo ER), campos (equivalentes a atributos no modelo ER), relacionamentos entre objetos e registos. Também foram abordados temas como importação/exportação de dados, validação, segurança e ferramentas de análise de dados (*reports* e *dashboards*).

A etapa de consultor teve foco nas *clouds* de *Sales* e *Service*, temas como a gestão de utilizadores, criação, configuração e customização de objetos e Apps Salesforce foram consolidados resolvendo vários projetos. Nesta etapa foram também abordados vários tipos de automação de processos recorrendo à ferramenta *Flow Builder*. Esta etapa além de ter vários módulos e projetos contou também com alguns superbadges.

A etapa de programador abordou três tipos de programação: Apex, Visualforce e LWC.

No estudo da linguagem de programação Apex foram abordados temas como a criação de classes e métodos, estruturas e controlo do fluxo de dados. Foi também estudada a linguagem de consulta SOQL semelhante a SQL que, em conjunto com Apex, permitem a manipulação dos registos na base de dados. Para além da construção de classes Apex também foi estudada a implementação de *triggers* e unidades de teste tanto para classes como para *triggers*. O IDE usado foi a *Developer Console* nativa do Salesforce.

O estudo da estrutura de desenvolvimento *web* Visualforce foi de nível básico, tendo passado pelo aprofundamento das funcionalidades da *Developer Console*, construção de páginas Visualforce simples usando uma linguagem de marcação semelhante a HTML e implementação de controladores em Apex responsáveis pelas funcionalidades destas páginas.

Por último foi abordada a estrutura *Lightning Web Components* (LWC). Os projetos e módulos abordados passaram pela criação de componentes de nível básico. Para tal foi necessária a instalação da interface de linha de comandos do Salesforce (Salesforce CLI) e configuração do IDE Visual Studio Code com a instalação da extensão *Salesforce Extension Pack*. Estas ferramentas permitem o desenvolvimento de componentes e funcionalidades customizadas (entre outros) e a sua implantação nas Orgs. Para o estudo desta estrutura do Salesforce (usada na criação de páginas e funcionalidades customizadas) o estagiário teve de se familiarizar com as linguagens de programação HTML e JavaScript uma vez que são as principais linguagens para programar em LWC. Devido à sua complexidade e ao calendário do estágio esta etapa não foi muito aprofundada.

6. Configuração de uma App Salesforce (2ª fase)

Na segunda fase do estágio pretendeu-se implementar parte de uma App Salesforce seguindo os requisitos impostos pela entidade acolhedora, sendo que um desses requisitos consistiu em implementar algumas funcionalidades sugeridas pelo próprio estagiário. A App MAXFIT tem como finalidade a gestão de eventos de uma empresa, ela deverá permitir o registo de eventos, organizadores, palestrantes e participantes bem como as inscrições dos palestrantes e participantes nos eventos.

Na implementação de uma App Salesforce existem vários aspetos a considerar, tais como objetos, campos, tipos de dados, regras de validação, regras de duplicados, perfis de utilizadores e permissões. A implementação foi feita numa Developer *Edition* Org.

6.1 Modelo de Dados

O modelo relacional dos dados da aplicação MAXFIT é composto por oito objetos como mostram as figuras 9 e 10. O Salesforce permite vários tipos de objetos, entre eles objetos *standard* ou padrão e objetos customizados. Os objetos padrão são nativos em cada Org, isto é, estão incluídos na versão/instância de cada cliente/utilizador do Salesforce. Embora estes objetos já contenham campos e relacionamentos entre si, podem ser customizados adicionando, por exemplo, novos campos e relacionamentos. Os objetos customizados são criados de raíz quando não existe um objeto *standard* que possa satisfazer o que é pretendido implementar. Neste trabalho foram apenas usados objetos customizados.

Tal como os objetos, os campos de um objeto podem ser *standard* ou customizados. Todos os objetos customizados possuem um conjunto de campos *standard* por defeito aquando da sua criação, entre eles, “ID”, “Criado por”, “Modificado pela última vez por”, “Nome” e “Dono” (este último campo só é criado se o objeto não for filho de outro objeto). Quando é criado um registo, é o próprio *software* que atribui um valor (único) para o campo “ID” correspondente à chave primária.

No que toca a relacionamentos entre objetos, podem distinguir-se dois[4] tipos diferentes apesar de serem ambos relacionamentos de um para muitos. O relacionamento *Master-Detail* (pai-filho) permite um vínculo forte entre dois objetos, o objeto filho herda as permissões do objeto pai, se um registo do pai for eliminado, o registo correspondente no filho é também eliminado. Além disso é possível criar um campo (*Roll-Up Summary*) no pai onde podem ser calculados valores (soma, mínimo, máximo, contagem) do conjunto de registos do objeto filho. O relacionamento *Lookup* cria um vínculo não tão forte, ele não permite a eliminação de registos em cascata e não permite a criação de campos do tipo *Roll-Up Summary*.

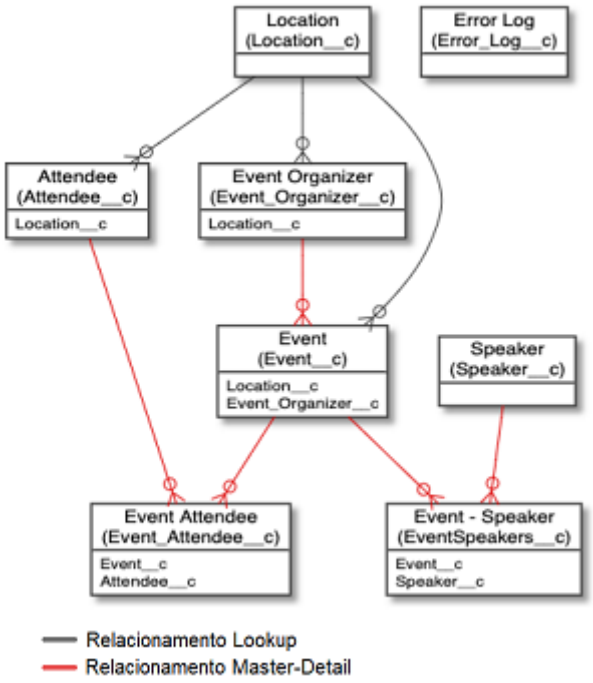


Figura 9 - Modelo de Dados da App MAXFIT.

[4] Existem mais tipos de relacionamentos que não foram abordados neste trabalho.

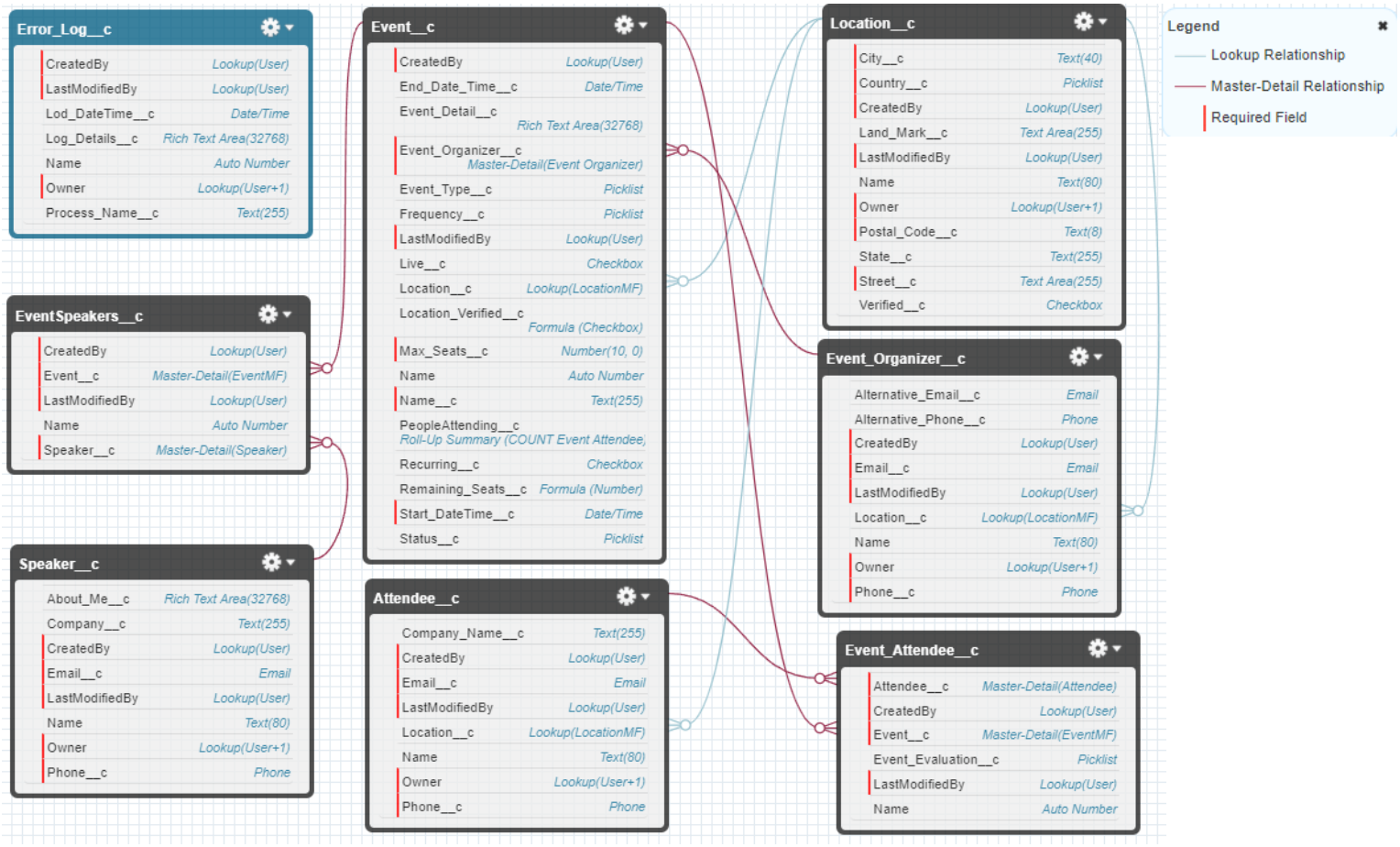


Figura 10 - Modelo de Dados da App MAXFIT gerado pelo Salesforce

6.2 Dicionário de Dados

As tabelas de 3 a 10 descrevem todos os objetos da App, serão descritos apenas a chave primária, o Nome (ambos campos *standard*) e os campos customizados, em Salesforce os relacionamentos *Lookup* e *Master-Detail* são considerados campos uma vez que apresentam valores para cada registo.

Nesta plataforma os objetos e campos têm dois nomes, a *Label* e o API *Name*. O primeiro é o nome apresentado na interface do utilizador e o segundo é o nome usado programaticamente em Apex ou em qualquer uma das APIs. O Nome API para objetos e campos customizados tem sempre o sufixo __c.

Tabela 3 - Dicionário de Dados do objeto "LocationMF".

Object Label	LocationMF		API Name	Location__c		
Descrição	Lista de localizações					
Campos						
API Name	Field Label	Descrição	Tipo de Dados	Tamanho	Valores válidos	Obrigatório
Id	Record ID	Chave primária	Text	18	Alfanumérico	Automático
Name	Name	Nome da localização	Text	80		Sim
Street__c	Street	Nome da rua	Text	255		Sim
City__c	City	Nome da cidade	Text	40		Sim
Postal_Code__c	Postal Code	Código Postal	Text	8		Sim
Country__c	Country	País	Picklist		(*)	Sim
Land_Mark__c	Land Mark	Pontos de referência	Long Text Area	255		Não
State__c	State	Estado	Text	255		Não
Verified__c	Verified?	Indica se a localização foi ou não verificada	Checkbox			Não

(*) Lista extensa de países fornecidas pela entidade acolhedora

Tabela 4 - Dicionário de Dados do objeto “Event Organizer”.

Object Label	Event Organizer		API Name	Event_Organizer__c		
Descrição	Lista de organizadores de eventos e dados pessoais					
Campos						
API Name	Field Label	Descrição	Tipo de Dados	Tamanho	Valores válidos	Obrigatório
Id	Record ID	Chave primária	Text	18	Alfanumérico	Automático
Name	Name	Nome do organizador	Text	80		Não
Email__c	Email	Email	Email	80	parte-local@domínio	Sim
Phone__c	Phone	Número de telefone	Phone	40		Sim
Alternative_Email__c	Alternative Email	Email alternativo	Email	80	parte-local@domínio	Não
Alternative_Phone__c	Alternative Phone	Número de telefone alternativo	Phone	40		Não
Location__c	Location	Chave estrangeira do objeto LocationMF	Relacionamento Lookup			Não

Tabela 5 - Dicionário de Dados do objeto “System Event”.

Object Label	System Event		API Name	Error_log__c		
Descrição	Log de erros					
Campos						
API Name	Field Label	Descrição	Tipo de Dados	Tamanho	Valores válidos	Obrigatório
Id	Record ID	Chave primária	Text	18	Alfanumérico	Automático
Name	log#	Código do erro	Auto Number	6	De LOG-000001 a LOG-999999	Automático
Lod_DateTime__c	Lod Date/Time	Data/Hora do erro	Date/Time			Não
Log_Details__c	Log Details	Descrição do erro	Rich Text Area	32768		Não
Process_Name__c	Process Name	Nome do processo	Text	255		Não

Tabela 6 - Dicionário de Dados do objeto "EventMF".

Object Label	EventMF		API Name	Event__c		
Descrição	Lista de eventos					
Campos						
API Name	Field Label	Descrição	Tipo de Dados	Tamanho	Valores válidos	Obrigatório
Id	Record ID	Chave primária	Text	18	Alfanumérico	Automático
Name	Event #	Código de evento	Autonumber	6	De EVT-000001 a EVT-999999	Automático
Location__c	LocationMf	Chave estrangeira do objeto LocationMF	Relacionamento Lookup			Não
Event_Organizer__c	Event Organizer	Chave estrangeira do objeto Event Organizer	Relacionamento Master-Detail			Sim
Name__c	Name	Nome do evento	Text	255		Sim
Start_DateTime__c	Start Date/time	Data/Hora de início	Date/Time			Sim
End_Date_Time__c	End Date/Time	Data/Hora de fim	Date/Time			Não
Recurring__c	Recurring?	Indica se o evento é recorrente	Checkbox			Não
Max_Seats__c	Max Seats	Número máximo de lugares	Number	10	Número inteiro [0;9999999999]	Sim
Live__c	Live?	Indica se o evento está ativo	Checkbox			Não
PeopleAttending__c	# People Attending	Contagem do número de registos do objeto Event Attendee para o mesmo evento	Roll-Up Summary			Automático
Remaining_Seats__c	Remaining Seats	Indica o número de lugares vagos num evento	Formula (Number)			Automático
Event_Type__c	Event Type	Indica o tipo de evento	Picklist		In-person; Virtual	Não
Frequency__c	Frequency	Indica a frequência do evento, se este for recorrente	Picklist		Daily; Weekly	Não
Location_Verified__c	Location Verified?	Toma o mesmo valor do registo do objeto LocationMF quando é selecionada uma localização	Formula (Checkbox)			Automático
Status__c	Status	Indica a fase em que está o evento	Picklist		Created;Published; In Progress; Completed; Postponed; Cancelled	Não

Tabela 7 - Dicionário de Dados do objeto “Speaker”.

Object Label	Speaker		API Name	Speaker__c		
Descrição	Lista de palestrantes e dados pessoais/profissionais					
Campos						
API Name	Field Label	Descrição	Tipo de Dados	Tamanho	Valores válidos	Obrigatório
Id	Record ID	Chave primária	Text	18	Alfanumérico	Automático
Name	Name	Nome do palestrante	Text	80		Sim
Email__c	Email	Email	Email	80	parte-local@domínio	Sim
Phone__c	Phone	Número de telefone	Phone	40		Sim
Company__c	Company	Nome da empresa	Text	255		Não
About_Me__c	About Me	Breve descrição do palestrante	Rich Text Area	32768		Não

Tabela 8 - Dicionário de Dados do objeto “Attendee”.

Object Label	Attendee		API Name	Attendee__c		
Descrição	Lista de participantes e dados pessoais/profissionais					
Campos						
API Name	Field Label	Descrição	Tipo de Dados	Tamanho	Valores válidos	Obrigatório
Id	Record ID	Chave primária	Text	18	Alfanumérico	Automático
Name	Name	Nome do participante	Text	80		Sim
Email__c	Email	Email	Email	80	parte-local@domínio	Sim
Phone__c	Phone	Número de telefone	Phone	40		Sim
Company__c	Company	Nome da empresa	Text	255		Não
Location__c	LocationMF	Chave estrangeira do objeto LocationMF	Relacionamento Lookup			Não

Tabela 9 - Dicionário de Dados do objeto “Event - Speaker”.

Object Label	Event - Speaker		API Name	EventSpeaker__c		
Descrição	Lista de palestrantes atribuídos a eventos (objeto de junção)					
Campos						
API Name	Field Label	Descrição	Tipo de Dados	Tamanho	Valores válidos	Obrigatório
Id	Record ID	Chave primária	Text	18	Alfanumérico	Automático
Name	Event Speaker	Código de registo de palestrante num evento	Auto Number	4	De A-0001 a A-9999	Automático
Event__c	Event	Chave estrangeira do objeto EventMF	Relacionamento Master-Detail			Sim
Speaker__c	Speaker	Chave estrangeira do objeto Speaker	Relacionamento Master-Detail			Sim

Tabela 10 - Dicionário de Dados do objeto “Event Attendee”.

Object Label	Event Attendee		API Name	Event_Attendee__c		
Descrição	Lista de participantes inscritos em eventos (objeto de junção)					
Campos						
API Name	Field Label	Descrição	Tipo de Dados	Tamanho	Valores válidos	Obrigatório
Id	Record ID	Chave primária	Text	18	Alfanumérico	Automático
Name	Attendee #	Código de inscrição de participante num evento	Auto Number	5	De ATT-00001 a ATT-99999	Automático
Event__c	Event	Chave estrangeira do objeto EventMF	Relacionamento Master-Detail			Sim
Attendee__c	Attendee	Chave estrangeira do objeto Speaker	Relacionamento Master-Detail			Sim

6.2.1 Roll-Up Summary

No objeto “EventMF” o campo “#People Attending” é do tipo *Roll-Up Summary*. O valor devolvido corresponde à contagem de registos existentes no objeto “Event Attendees” (inscrições de participantes em eventos) para o mesmo evento. A figura 11 mostra como é configurado este campo na interface do utilizador.

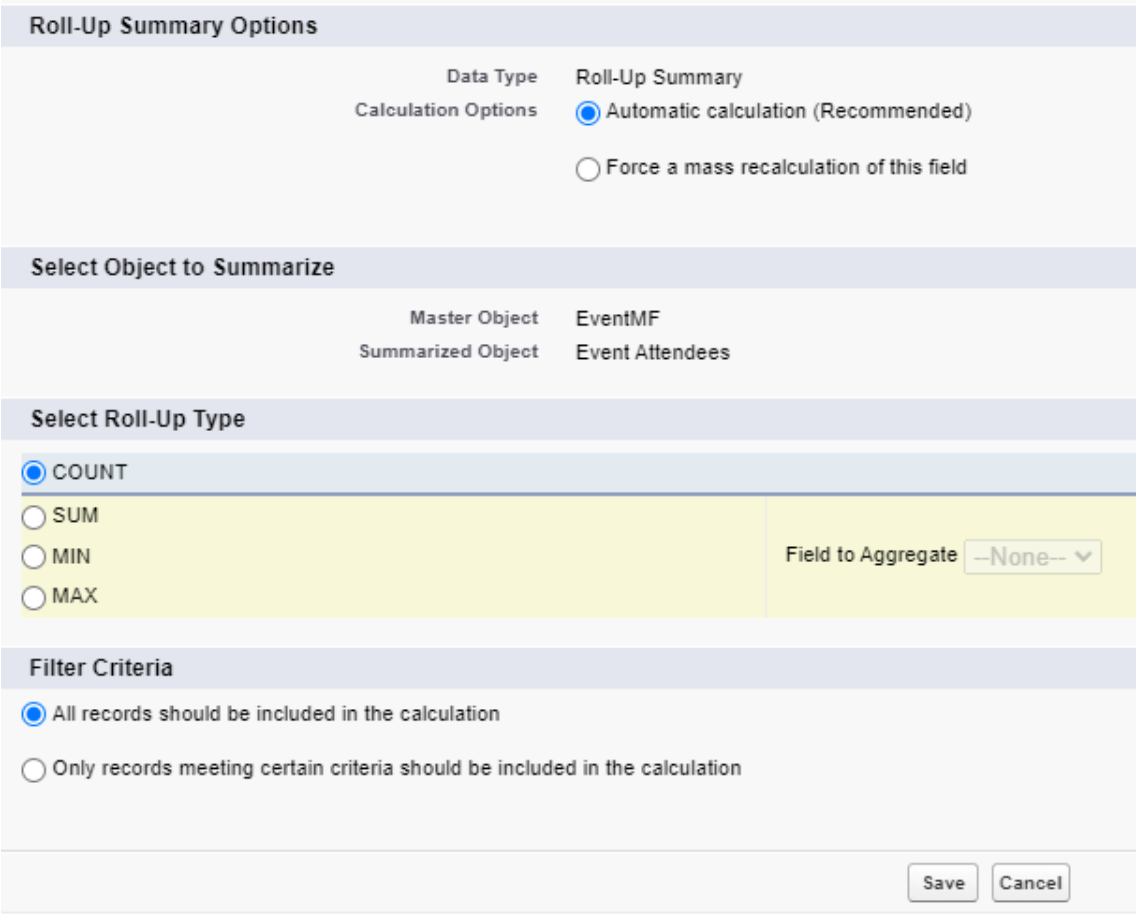


Figura 11 - Configuração do *Roll-Up Summary* do campo "#People Attending".

6.2.2 Fórmulas

No objeto "EventMF" existem dois campos cujo tipo de dados tem a designação de "*Formula*", são eles "Remaining_Seats__c" e "Location_Verified__c". O primeiro destes campos devolve um valor do tipo *Number* enquanto o segundo devolve um valor do tipo *Checkbox* (verdadeiro ou falso).

A figura 12 mostra a interface de utilizador na página de configurações dedicada à gestão de objetos relativa ao objeto "EventMF" e campo "Remaining Seats" mostrando a fórmula utilizada para o cálculo dos lugares disponíveis, que mais não é do que a subtração dos valores numéricos dos campos "Max Seats" e "# People Attending".

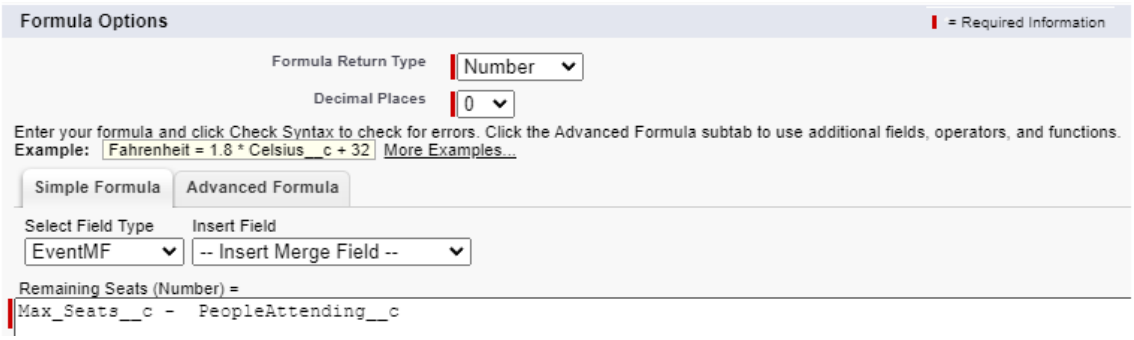


Figura 12 - Fórmula de cálculo para a obtenção do valor do campo “Remaining Seats”.

O campo “Location Verified?” devolve um valor do tipo *checkbox*, isto é, um valor booleano, se a *checkbox* estiver selecionada o valor correspondente é “verdadeiro” se não estiver selecionada devolve “falso”. A fórmula implementada (figura 13) referencia o valor do campo “Verified?” do objeto “LocationMF”.

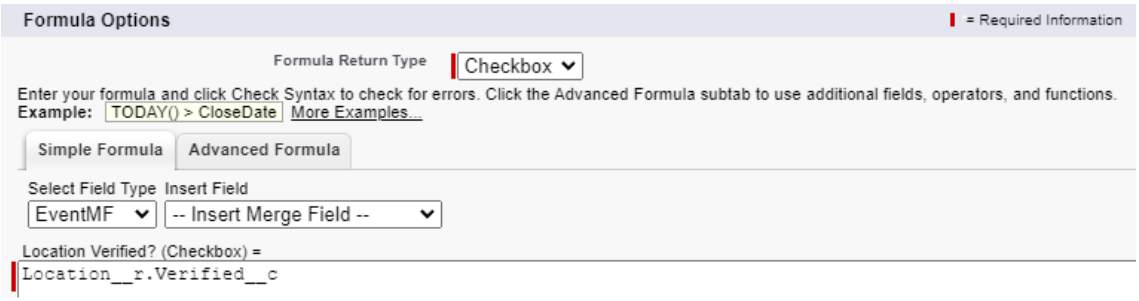


Figura 13 - Fórmula de cálculo para a obtenção do valor do campo “Location Verified?”.

6.3 Regras de validação de dados

Quando é criado um campo num objeto, é selecionado um tipo de dados específico, ao fazer esta seleção já se está a introduzir um tipo de critério de validação de dados, mas o Salesforce tem uma funcionalidade que vai mais além do que este critério de validação, são chamadas de regras de validação, que podem ser aplicadas em qualquer campo e têm critérios mais finos de configuração. Estas regras verificam se os dados inseridos pelo utilizador respeitam os padrões especificados para que o registo possa ser guardado com sucesso e haja consistência nos dados.

Uma regra de validação pode conter uma fórmula ou expressão que avalia os dados a inserir em um ou mais campos, retornando um valor booleano (verdadeiro ou falso). Se este valor for "verdadeiro" é mostrada uma mensagem de erro ao utilizador.

Foram implementadas sete regras de validação de dados, cinco delas relativas ao objeto "EventMF", uma para o objeto "Event Attendee" e outra para o objeto "Event Speaker".

6.3.1 Regras de validação no objeto "EventMF"

A primeira regra que faz com que, caso se pretenda inserir uma data e hora de término de um evento (pois este campo não é de preenchimento obrigatório), este valor tenha de ser, pelo menos, vinte e quatro horas posterior ao valor do campo referente à data e hora do início do evento. A figura 14 mostra o resultado da introdução de dados inválidos neste campo, os detalhes desta regra podem ser consultados no Anexo A.

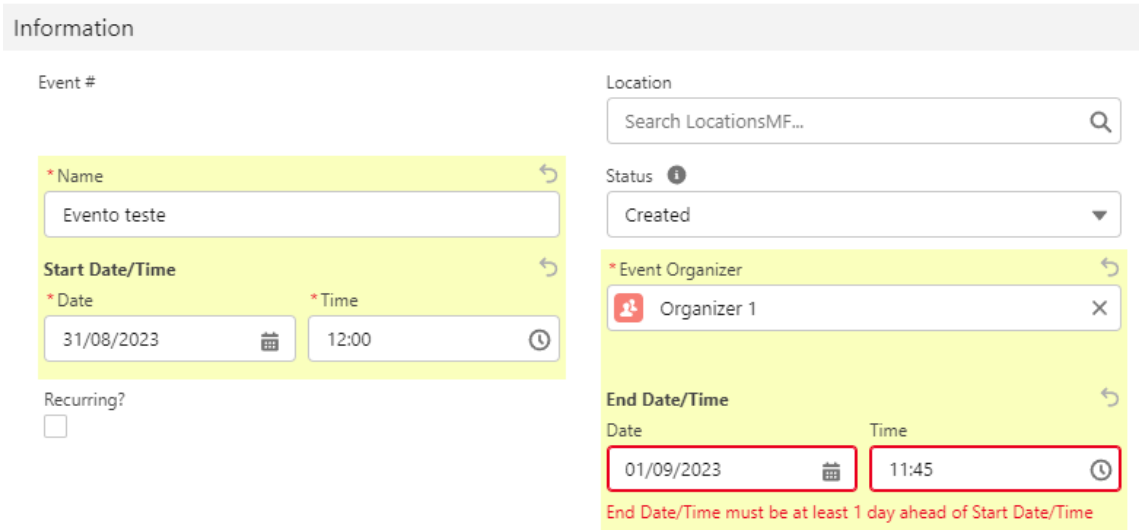


Figura 14 - Ativação da regra de validação do campo "End Date/Time".

Foram também implementadas duas regras de validação que relacionam os valores dos campos "Event Type" e "Location". Se o tipo de evento for "In-Person" então terá de ser escolhida forçosamente uma localização (figura 15).

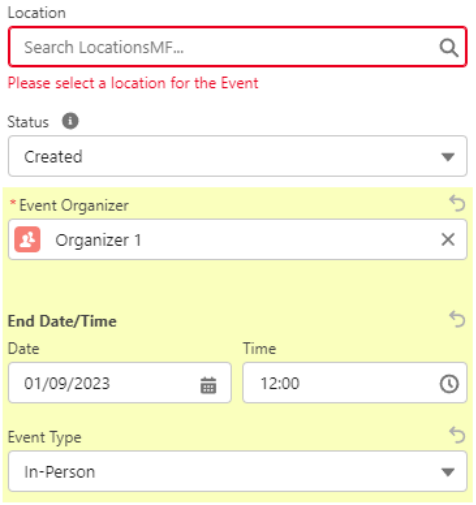


Figura 15- Ativação da regra de validação para eventos do tipo "In-Person".

Caso o tipo de evento seja do tipo "Virtual" então não pode ser definida nenhuma localização no evento (figura 16). Os detalhes destas regras podem ser consultados no Anexo A.

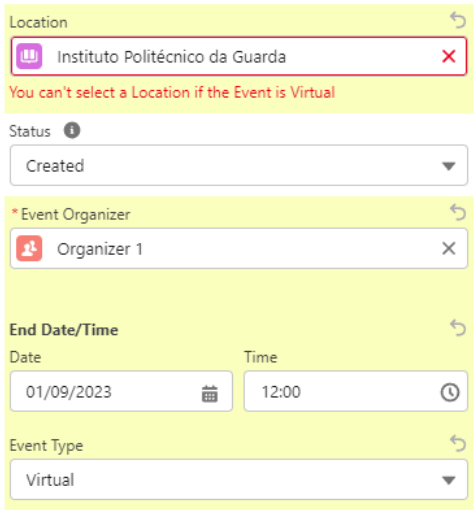


Figura 16 -Ativação da regra de validação para eventos do tipo "Virtual".

As figuras 17 e 18 mostram a ativação das regras de validação que relacionam os valores dos campos "Recuring?" e "Frequency". Se um evento é definido como recorrente (selecionando a *checkbox* deste campo), então terá que ser especificada a frequência do

mesmo (“Daily” ou “Weekly”). Caso o evento não seja recorrente, o utilizador não pode definir a frequência. Os detalhes destas regras podem ser consultados nos Anexo A.

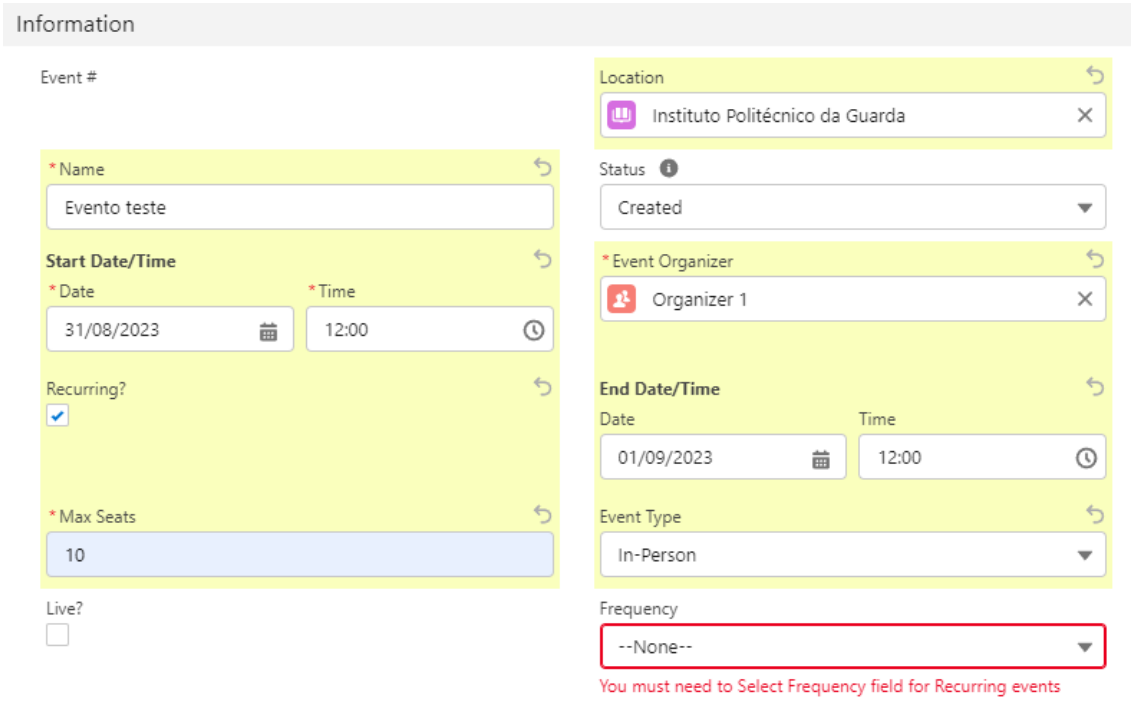


Figura 17- Ativação da regra de validação para eventos recorrentes.

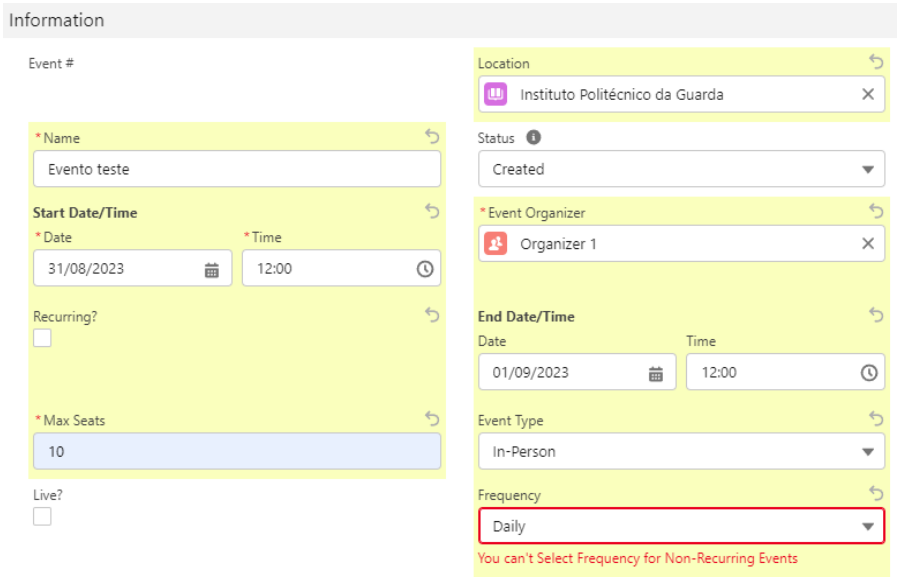


Figura 18 - Ativação da regra de validação para eventos não recorrentes.

6.3.2 Regras de validação no objeto "Event Speaker"

Neste objeto foi implementada uma regra de validação (Anexo A) que impede o utilizador de guardar um registo (figura 19) quando a *checkbox* do campo "Live?" do objeto "EventMF" não está selecionada ou se a data e hora do término do evento já tenha passado.

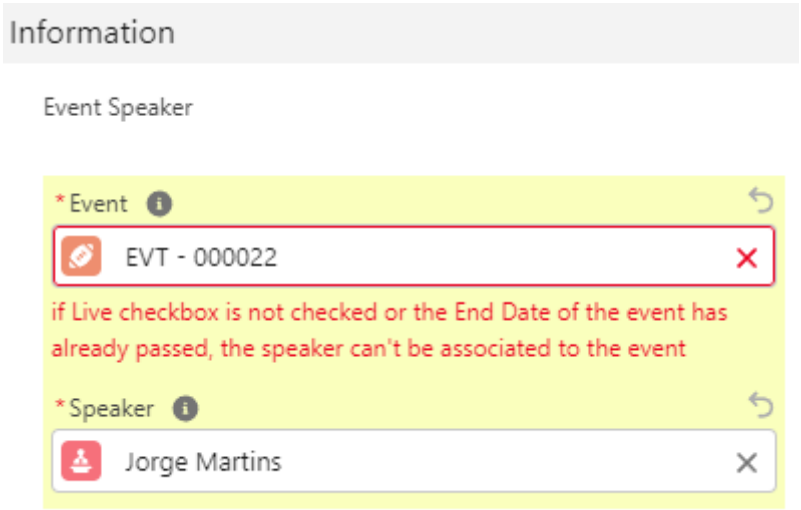


Figura 19 - Ativação da regra de validação para o objeto "Event Speaker".

6.3.3 Regras de validação para o objeto "Event Attendee"

Esta regra de validação (Anexo A) impede a inscrição de um participante num evento caso a data de término já tenha passado, ou a checkbox do campo "Live?" do objeto "EventMF" não esteja selecionada, ou não existam lugares disponíveis. A figura 20 mostra a mensagem de erro lançada na interface de utilizador quando a regra é ativada.

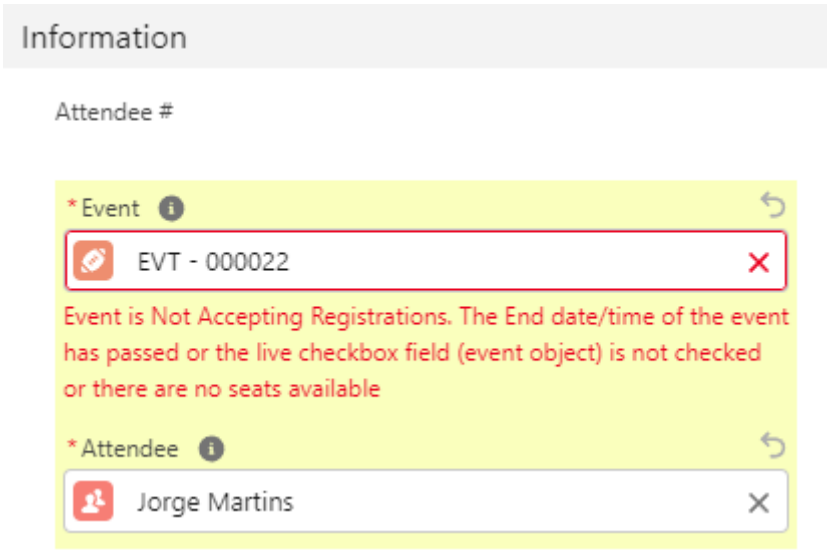


Figura 20 - Ativação da regra de validação para o objeto "Event Attendee".

6.4 Regras de Duplicidade de dados

Estas regras servem para definir critérios para que não seja permitido introduzir registos duplicados na base de dados. Neste trabalho foram implementadas regras para que não houvesse registos nos objetos "Event Organizer", "Speaker" e "Attendee" que tivessem o mesmo valor nos campos "Email" e "Phone". Para caso do objeto "Attendee" também se incluiu o campo "Name".

Para se implementar este tipo de regra é necessário criar primeiro regras de correspondência onde são especificados os campos que se querem comparar entre registos do mesmo objeto e como se querem comparar (Anexo B). O passo seguinte é criar a regra de duplicidade para cada objeto, atribuindo uma regra de correspondência e definindo se a comparação de valores é feita na criação e/ou atualização do registo. No Anexo C são apresentadas as regras de duplicidade implementadas. A título de exemplo é mostrado na figura 21 a interface de utilizador quando a regra de duplicidade é ativada na criação ou atualização de um registo no objeto "Attendee" nos outros dois casos o resultado é idêntico.

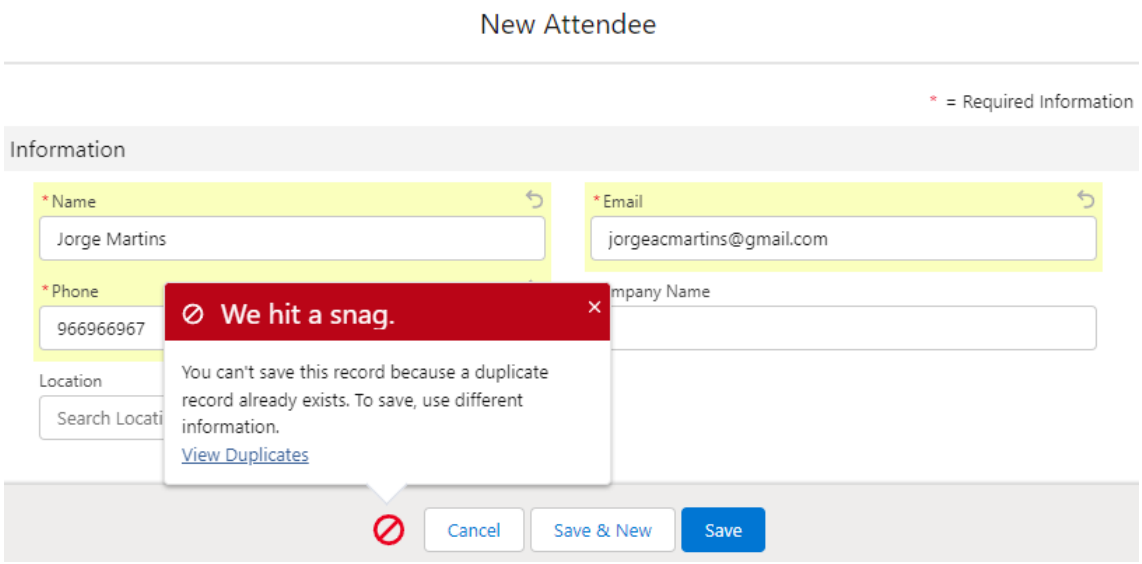


Figura 21 - Ativação da regra de duplicidade para o objeto "Attendee".

6.5 Permissões de acesso aos dados

Um aspeto fundamental em Salesforce é a configuração de acesso aos dados, isto é, definir "quem pode ver o quê". Existem vários níveis de permissões que serão abordados seguidamente.

6.5.1 Perfis

Um utilizador do *software* CRM da Salesforce tem de ter forçosamente um perfil atribuído. Os perfis em Salesforce são os responsáveis (entre outros aspetos) pela segurança ao nível dos objetos da Org, determinando que objetos os utilizadores podem ver e se podem criar, editar ou eliminar registos próprios nesses objetos. Estas permissões são comummente chamadas de CRED (*Create*, *Read*, *Edit* e *Delete*). Assim, para a criação de perfis tem de se aceder a *Profiles* no menu *User* na página de *Setup* da interface de utilizador (IU) como mostra a figura 22. Foram criados três perfis genéricos, "Event Organizer", "Speaker" e "Attendee", que interagem com a App. Para além destes perfis existe um perfil, "System Administrator", que é criado por defeito em cada Org.

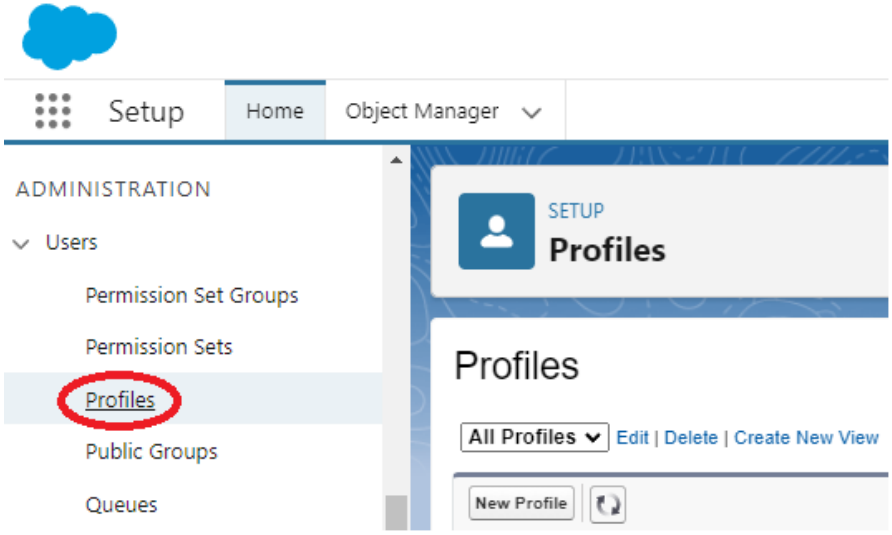


Figura 22 - Página de Setup para a criação de novos perfis.

Foram depois criados três utilizadores (*users*) tipo (“Eventorganizer Test”, “Speaker Test” e “Attendee Test”) acedendo a *Users* no menu *Users* da página de *Setup* da IU (figura 23). A cada utilizador foi atribuído o perfil correspondente.

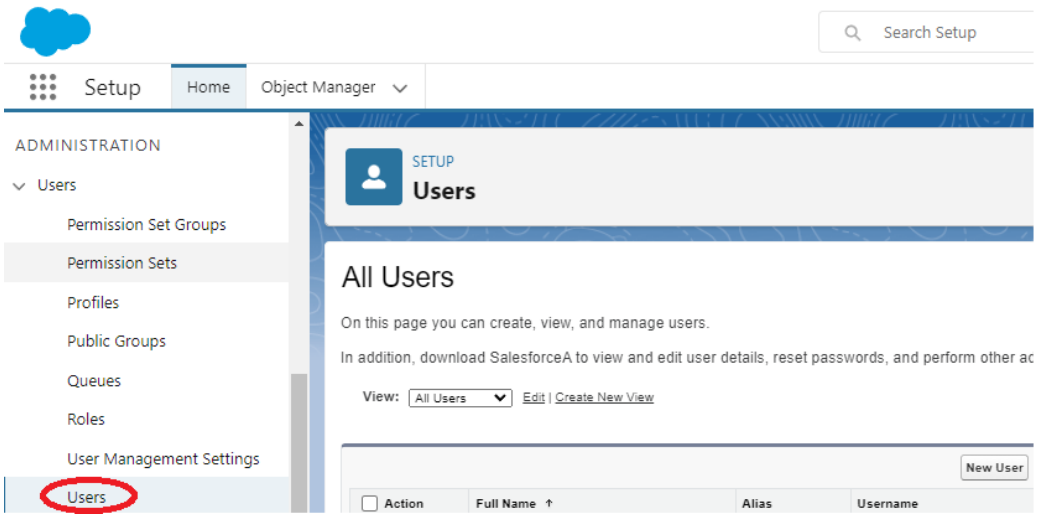


Figura 23 - Página de Setup para a criação de novos utilizadores.

A tabela 11 mostra as permissões CRED definidas para os três tipos de perfis, o perfil “System Administrator” possui todas as permissões ao nível de objetos.

Tabela 11 - Permissões a nível dos objetos para cada perfil de utilizador.

OBJETO	PERFIL		
	Event Organizer	Speaker	Attendee
EventMF	CRED	R	R
Event Organizer	CRE	R	R
Speaker	CRE	CRED	R
Attendee	R	X	CRE
Location	CRED	R	CRE
Event Attendee	CRED	X	CR
Event Speaker	CRED	CRE	R
System Event	X	X	X

C - Criar; R - Ver; E - Editar; D - Eliminar; X - Sem Acesso

Na figura 24, a título de exemplo, pode ver-se a IU com as permissões ao nível dos objetos da App para o perfil Event Organizer.

Estas definições são configuradas em *Profiles* no menu de *Users* na página de *Setup* da IU.

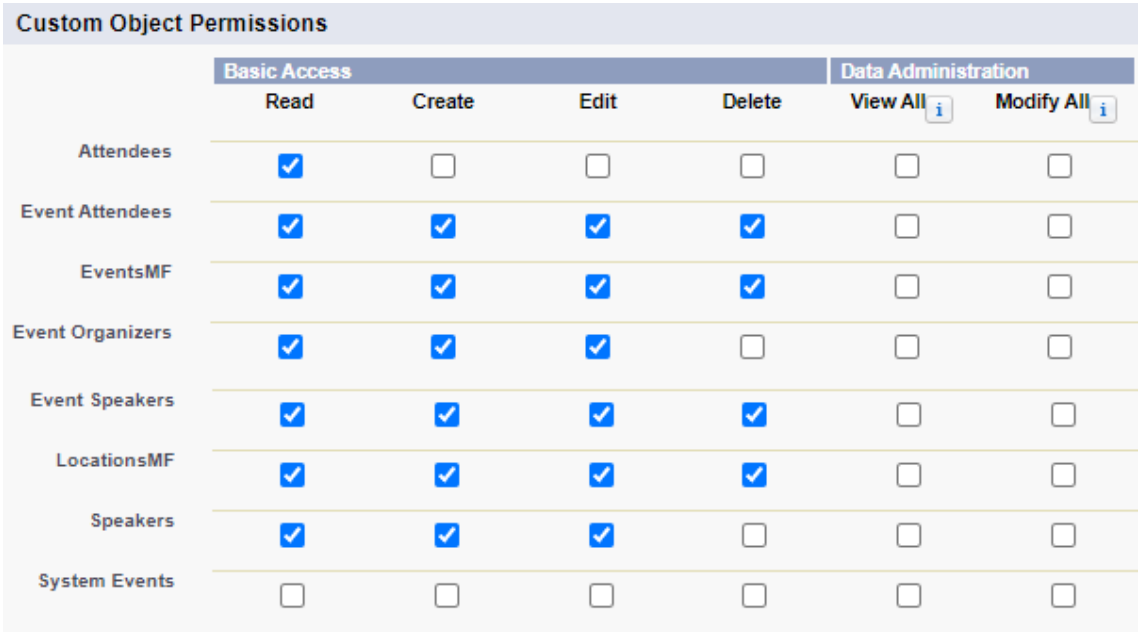


Figura 24 - Permissões de acesso aos objetos da App MAXFIT para o perfil “Event Organizer”.

6.5.2 Predefinições para toda a Organização (Organization Wide Defaults - OWD)

Estas permissões especificam o nível básico de acesso que os utilizadores têm aos registos uns dos outros e podem ser definidas como:

- Privado (não é permitido o acesso aos registos a outros utilizadores que não ao utilizador dono dos registos);
- Controlado pelo pai (o acesso aos registos de um objecto é o mesmo que está definido para os registos do objeto pai);
- Leitura pública (o acesso aos registos permite apenas a visualização dos mesmos);
- Leitura/Escrita pública (o acesso aos registos permite a leitura e a modificação dos mesmos);

Este tipo de permissão não se sobrepõe às permissões aos objetos definidas nos perfis. A tabela 12 mostra as permissões OWD definidas para os objetos da App.

Tabela 12- Permissões ao nível dos registos da App MAXFIT.

OBJETO	OWD
EventMF	Controlado pelo pai
Event Organizer	Leitura pública
Speaker	Privado
Attendee	Privado
Location	Leitura pública
Event Attendee	Controlado pelo pai
Event Speaker	Controlado pelo pai
System Event	Leitura/Escrita pública

Estas permissões são definidas acedendo a *Sharing Settings* no menu *Security* na página de *Setup* da IU.

6.5.3 Hierarquia de Funções e Regras de Partilha

Outro nível a considerar no que toca à partilha de dados é a definição de uma hierarquia de funções, isto é, do papel que o utilizador tem na estrutura da organização. Assim, é possível permitir que os superiores hierárquicos tenham acesso aos registos dos seus subordinados independentemente das permissões OWD. Estes papéis são atribuídos a cada utilizador. Assim, o utilizador "Eventorganizer Test" tem o papel "Organizer", o utilizador "Speaker Test" tem o papel "Speaker" e o utilizador "Attendee Test" tem o papel "Attendee". A figura 25 mostra as três funções hierárquicas (em inglês: *roles*) definidas na opção *Roles* do menu *Users* da página de *Setup* da IU.

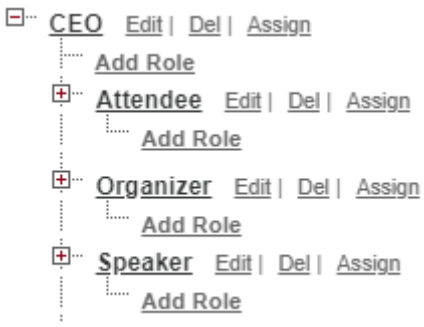


Figura 25 - Hierarquia de funções na App MAXFIT.

No caso dos objetos *standard* esta permissão está sempre ativa, no caso dos objetos customizados (como é o caso da App MAXFIT) a permissão pode ou não estar ativa. A ativação desta permissão é feita nos *Sharing Settings* no menu *Security* na página de *Setup* da IU (figura 26) neste trabalho estas permissões estão sempre ativas.

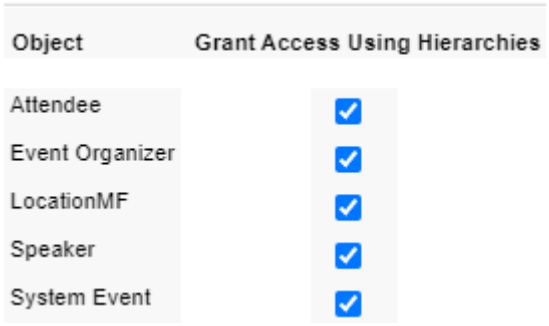


Figura 26- Ativação/Desativação das permissões Hierárquicas na IU.

As regras de partilha estendem as permissões OWD para determinados grupos de utilizadores ou papéis (*roles*) de modo que estes possam ter acesso a registos de outros utilizadores que normalmente não poderiam ter. Foram implementadas quatro regras de partilha (tabela 13).

Tabela 13 - Regras de partilha da App MAXFIT.

OBJETO	REGRA DE PARTILHA	
	Função Hierárquica	Tipo de Acesso
Speaker	Organizer	Leitura/Escrita
Attendee	Organizer	Leitura/Escrita
Event Organizer	Speaker	Leitura/Escrita
Event Organizer	Attendee	Leitura/Escrita

As regras sobre os objetos “Speaker” e “Attendee” fazem com que sejam partilhados os dados dos registos destes objetos com o utilizador com função hierárquica “Organizer” (que neste caso é o utilizador “Eventorganizer Test”). Isto permite que este utilizador possa criar registos nos objetos “Event Speaker” e “Event Attendee”.

As regras sobre o objeto “Event Organizer” (figura 27) permitem aos utilizadores com papel “Speaker” e “Attendee” criarem registos nos objetos “Event Speaker” e “Event

Attendee" respetivamente usando os registos do objeto "EventMF" criados por outros utilizadores. Isto é possível porque o objeto "EventMF" é filho do objeto "Event Organizer" (relacionamento Master-Detail), ou seja, ele herda as permissões do objeto pai. Apesar de as regras darem permissões de leitura/escrita, estas não se sobrepõem às permissões ao nível dos objetos (criadas no perfil) logo, por exemplo, o utilizador "Eventorganizer Test" não vai conseguir editar os registos do objeto "Attendee", apenas os pode ver.

Figura 27 - Regras de partilha para os registos do objeto "Event Organizer".

6.6 Criação da App

Em Salesforce uma App é uma janela na qual se pode aceder a um conjunto de separadores (*tabs*), cada um deles constituído por diferentes tipos de informações e funcionalidades. Uma *tab* pode estar associada a um objeto permitindo assim o acesso, edição ou introdução de registos. Cada instância de Salesforce contém por defeito várias Apps *standard* que podem ser personalizadas, mas é possível também construir novas Apps customizadas de raíz.

Neste trabalho foi criada uma App de nome MAXFIT com os oito objetos já descritos, recorrendo à ferramenta App Manager onde, em apenas cinco passos se consegue criar uma App Salesforce. No primeiro passo definem-se alguns detalhes como o nome e imagem da App. No segundo passo são configurados o tipo de navegação, tipo de dispositivo (móvel e/ou *desktop*) e tipo de experiência de configuração. O terceiro passo só é aplicável a Apps para *desktop*, permitindo configurar uma barra de ferramentas no rodapé, no quarto passo são selecionadas as *tabs* que vão compor a App. Finalmente, no

quinto passo são definidos os perfis de utilizador que podem aceder à App. A figura 28 mostra as tabs da App para cada perfil de utilizador (cada perfil tem acesso a um conjunto de tabs diferentes).

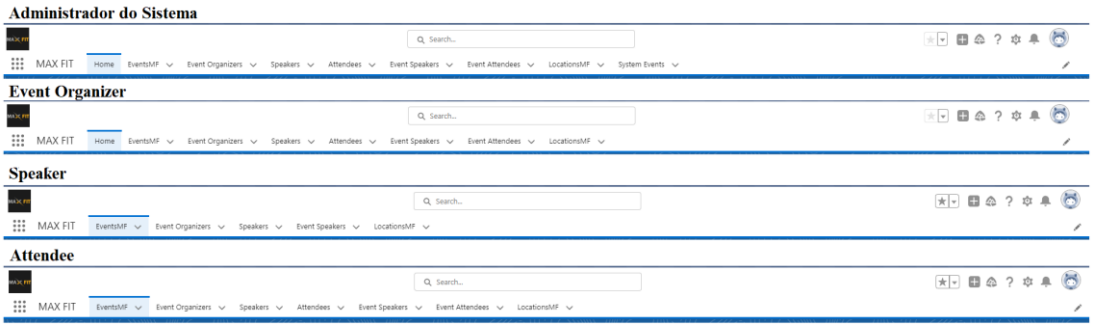


Figura 28 - Tabs da App MAXFIT para cada perfil de utilizador.

7. Desenvolvimento de funcionalidades da App (2ª fase)

No que toca ao desenvolvimento de funcionalidades em Salesforce podemos considerar as que são implementadas recorrendo a ferramentas *low-code/no-code* (*flows* por exemplo) e as que são desenvolvidas recorrendo a código Apex e *Lightning Web Components*. Neste projeto foram desenvolvidas funcionalidades recorrendo a ferramentas *low-code/no-code* e a código Apex.

7.1 Reports e Dashboards

Os *reports* e *dashboards* são funcionalidades chave do Salesforce para a análise e visualização de dados, com eles consegue-se ver os dados mais relevantes obtendo assim uma visão geral que permita tomar decisões informadas.

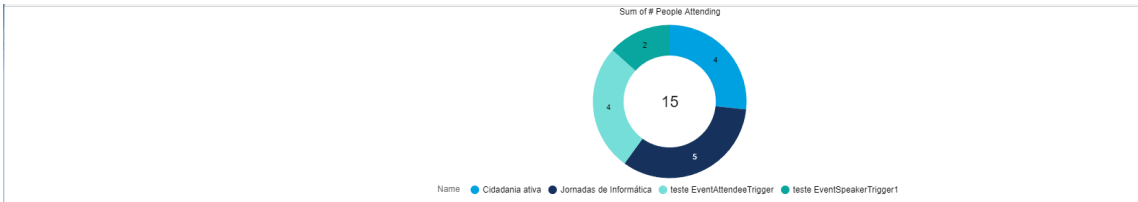
Um *report* é uma representação tabular da lista dos registos que satisfazem determinado critério definido pelo utilizador (podendo incluir também um gráfico personalizado) e que pode ser filtrada ou agrupada com base em qualquer campo.

Um *dashboard* consiste na representação visual de um ou mais *reports* que podem incluir tabelas ou vários tipos de gráficos, facilitando assim a análise dos dados.

Neste trabalho foram desenvolvidos dois *reports* e um *dashboard* que foi incorporado a uma nova *tab* ("Home") da aplicação apenas acessível ao utilizador com o perfil de "Event Organizer" ou "System Administrator".

7.1.1 Report Attendees in Event

Este *report* (figura 29) agrega os dados de alguns campos dos objetos "EventMF" e "Event Attendee" numa tabela, agrupando-os por evento e filtrando-os de maneira que só sejam mostrados eventos futuros. É gerado também um gráfico mostrando o número de participantes por evento (desde que haja participantes inscritos).

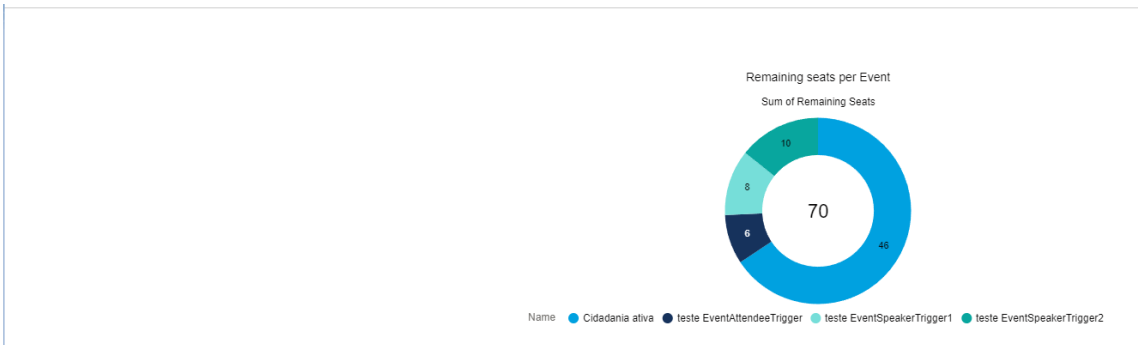


Name	Event Organizer: Name	Event #	Attendee #	Attendee: Name	Status	# People Attending	Remaining Seats	Start Date/Time	End Date/Time	Event Type	Location: Name
Cidadania ativa	Organizer 3	EVT - 000021	ATT - 00052	Attendee2	Created	4	46	14/09/2023, 12:00	-	In-Person	Câmara Municipal da Guarda
			ATT - 00051	Attendee6	Created	4	46	14/09/2023, 12:00	-	In-Person	Câmara Municipal da Guarda
			ATT - 00054	Pedro Attendee	Created	4	46	14/09/2023, 12:00	-	In-Person	Câmara Municipal da Guarda
			ATT - 00053	Attendee 1	Created	4	46	14/09/2023, 12:00	-	In-Person	Câmara Municipal da Guarda
Jornadas de Informática	Organizer created by admin	EVT - 000020	ATT - 00048	Attendee3	Created	5	0	25/09/2023, 12:00	29/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
			ATT - 00047	Attendee2	Created	5	0	25/09/2023, 12:00	29/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
			ATT - 00049	Attendee4	Created	5	0	25/09/2023, 12:00	29/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
			ATT - 00050	Attendee5	Created	5	0	25/09/2023, 12:00	29/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
			ATT - 00046	Attendee 1	Created	5	0	25/09/2023, 12:00	29/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
teste EventAttendeeTrigger	Organizer 2	EVT - 000019	ATT - 00043	Jorge Martins	Created	4	6	15/09/2023, 12:00	16/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
			ATT - 00055	Attendee2	Created	4	6	15/09/2023, 12:00	16/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
			ATT - 00056	Attendee5	Created	4	6	15/09/2023, 12:00	16/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
			ATT - 00057	Attendee4	Created	4	6	15/09/2023, 12:00	16/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda

Figura 29 - Report “Attendees in Event”.

7.1.2 Report Upcoming Events

Este *report* (figura 30) também só apresenta eventos futuros e contém dados de vários campos dos objetos “EventMF” e “Event Speaker”. É gerado um gráfico que mostra o número de lugares ainda disponíveis por cada evento (se não houver lugares disponíveis o evento não é representado).



Name	Speaker: Name	Status	# People Attending	Remaining Seats	Start Date/Time	End Date/Time	Event Type	Location: Name
Cidadania ativa (2)	Speaker 2 (1)	Created	4	46	14/09/2023, 12:00	-	In-Person	Câmara Municipal da Guarda
	Speaker 3 (1)	Created	4	46	14/09/2023, 12:00	-	In-Person	Câmara Municipal da Guarda
Jornadas de Informática (1)	Speaker 1 (1)	Created	5	0	25/09/2023, 12:00	29/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
teste EventAttendeeTrigger (1)	Jorge Martins (1)	Created	4	6	15/09/2023, 12:00	16/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
teste EventSpeakerTrigger1 (1)	Speaker1 created by speaker user (1)	Created	2	8	29/09/2023, 12:00	30/09/2023, 12:00	In-Person	Instituto Politécnico da Guarda
teste EventSpeakerTrigger2 (1)	- (1)	Created	0	10	29/09/2023, 12:00	01/10/2023, 12:00	Virtual	-
Total (6)			15	70				

Figura 30 - Report “Upcoming Events”.

7.1.3 Dashboard Upcoming Events

O *dashboard* implementado (figura 31) vai buscar informação aos dois *reports* consistindo em quatro tabelas e dois gráficos que mostram de uma forma sumária informação relativa a eventos, palestrantes e participantes.

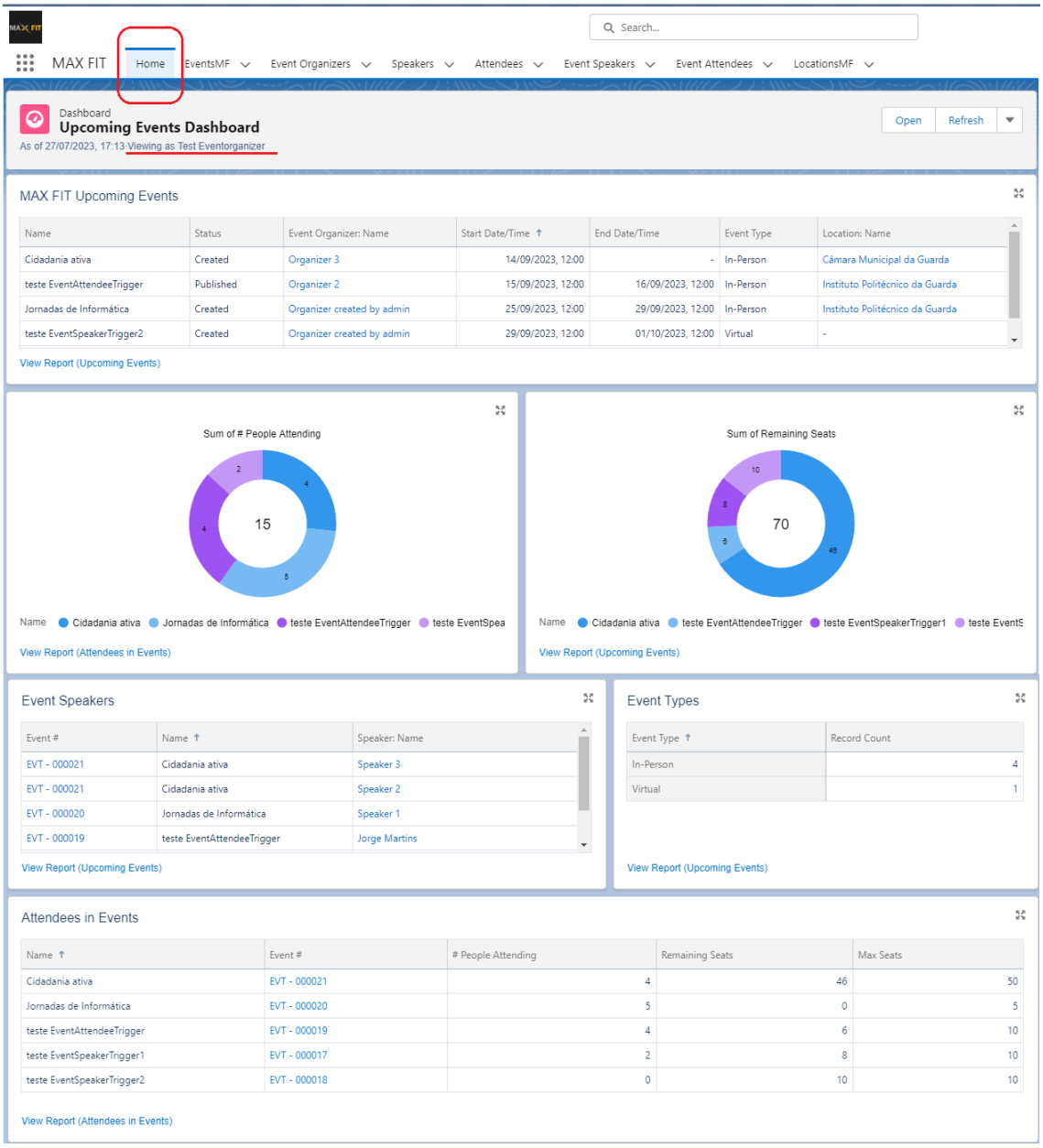


Figura 31- Dashboard “Upcoming Events”.

7.2 Flows

Um *flow* em Salesforce é uma ferramenta que automatiza uma sequência de atividades necessárias para atingir um determinado objetivo, para isso, ele recolhe dados e depois faz algo com eles. Os *flows* são construídos recorrendo ao *Flow Builder*, que é a interface declarativa, nativa do Salesforce, para a criação de flows individuais e que consegue criar lógica semelhante a código sem usar uma linguagem de programação, é uma solução *no-code*. Existem vários tipos de *flows*, neste trabalho foi construído um *Screen Flow* que é caracterizado por ter um elemento que é exibido na IU e que requer algum tipo de input por parte do utilizador.

A finalidade do "Event Evaluation Flow" (figura 32) é a de permitir que um utilizador participante possa avaliar um evento em que está inscrito.

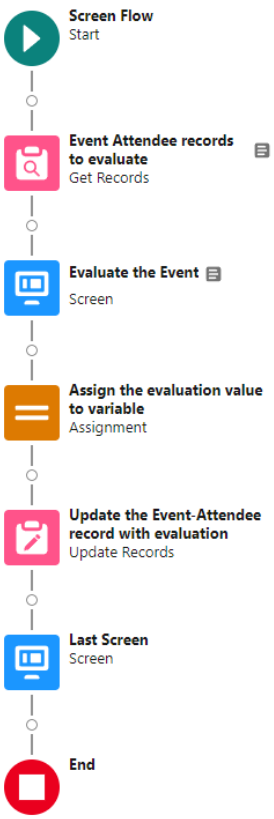


Figura 32 - Estrutura do Screen Flow "Event Evaluation".

Na página que corresponde à inscrição do participante num evento (registo no objeto "Event Attendee") é mostrado um primeiro painel (figura 33) onde o utilizador pode escolher uma pontuação para o evento (figura 34).

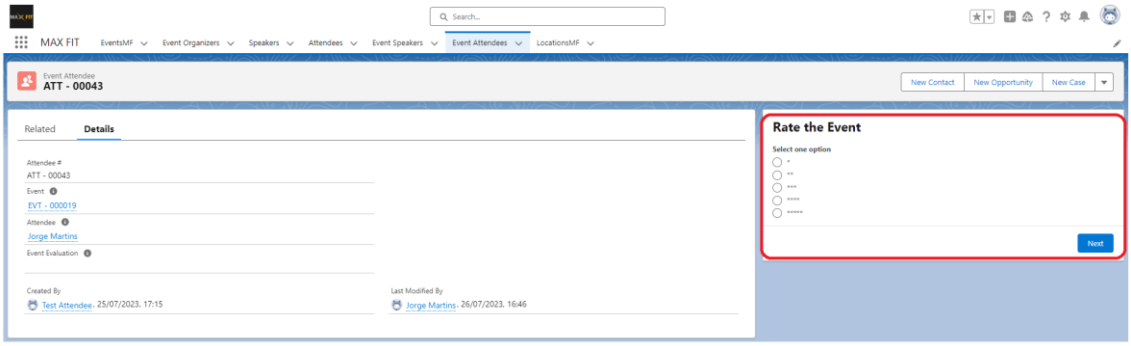


Figura 33 - Interface do utilizador participante mostrando um registo e o painel do flow.

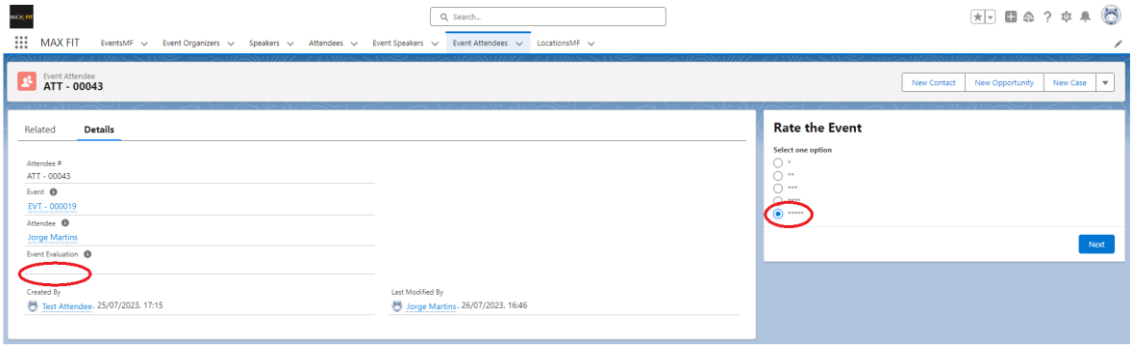


Figura 34 - Interface do utilizador participante mostrando um registo e o painel referente ao "Event Evaluation Flow" com avaliação do evento ainda não submetida.

Uma vez submetida a avaliação, é mostrado um segundo painel que dá duas opções ao utilizador, voltar atrás ou terminar (figura 35). Se o utilizador optar pela opção de terminar, o painel desaparece e é atribuído o valor da avaliação ao campo "Event Evaluation" do objeto "Event Attendee" (figura 36).

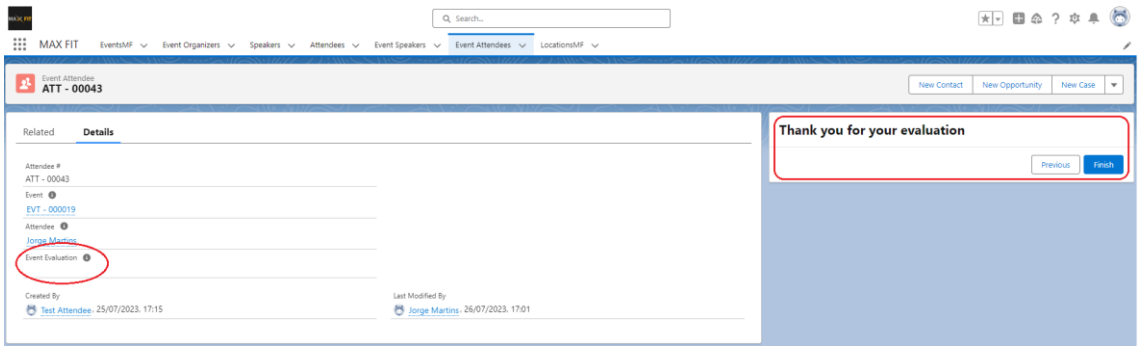


Figura 35 - Interface do utilizador participante mostrando um registo e o segundo painel referente ao "Event Evaluation Flow" com avaliação do evento submetida.

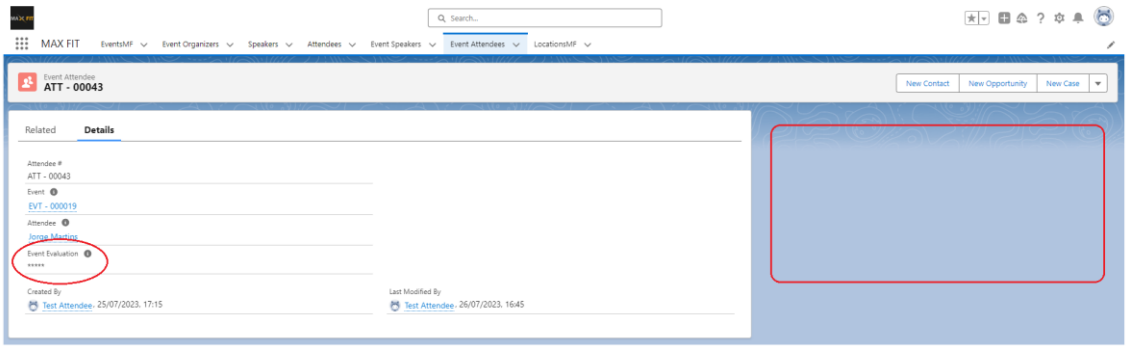


Figura 36 - Interface do utilizador participante mostrando um registo após atribuição do valor da avaliação ao campo "Event Evaluation".

Para que isto seja possível o *flow* vai buscar o registo onde é feita a avaliação e mostra o primeiro painel, estas etapas são feitas logo no momento em que se acede a um registo do objeto "Event Attendee" que não tenha já uma avaliação. Quando o utilizador seleciona o valor da classificação e clica no botão "Next" esse valor é atribuído a uma variável que por sua vez vai ser atribuída ao valor do campo "Event Evaluation" do objeto "Event Attendee" por uma operação de DML de atualização de registo. Por fim é mostrado um segundo painel que permite que o utilizador volte atrás e reveja a sua classificação ou termine o processo. Uma vez terminado o *flow*, a classificação é mostrada no campo "Event Evaluation" e o painel desaparece.

7.3 Classe Apex - Transaction Log Handler

À semelhança do Java, em Apex é possível criar classes, estas podem ser um conjunto de outras classes, métodos definidos pelo utilizador, variáveis, tipos de exceção e código de inicialização estático. Neste trabalho foi implementada a classe Apex "Transaction Log Handler" com o objetivo de criar registos no objeto "System Event". É pretendido que esta classe seja incorporada noutras classes (*triggers*) e, caso exista algum erro ou exceção é criado um registo contendo detalhes do erro, data/hora do erro e o nome do processo onde o erro ocorreu. Esta classe contém três métodos, um para lidar com as exceções do sistema criando uma lista com valores para atribuir aos campos do objeto "System Event", outro para lidar com erros que possam ocorrer nos métodos dos *triggers* criando também uma lista com valores para atribuir aos campos do objeto "System Event" e um último método para inserir os valores das listas criadas na base de dados, isto é, no objeto. O código desta classe Apex encontra-se no Anexo D e foi desenvolvido recorrendo ao IDE nativo do Salesforce, a Developer Console.

7.4 Triggers

Um *trigger* em Salesforce é código Apex que é executado antes ou depois de operações de manipulação de dados (DML) como *insert*, *update* ou *delete*. Assim, é possível criar automação de tarefas que seriam impossíveis de implementar apenas com recurso à IU.

Foram implementados três *triggers*, um no objeto "Event Speaker" e dois no objeto "Event Attendee" recorrendo à Developer Console.

7.4.1 Event Speaker Trigger

Este *trigger* tem como função impedir que seja criado ou atualizado um registo que atribui um palestrante a um evento, se esse mesmo palestrante já estiver atribuído a um evento que comece no mesmo dia à mesma hora. Quando se tentar introduzir um registo que cumpra esta condição é mostrado um aviso de erro na IU (figura 37).

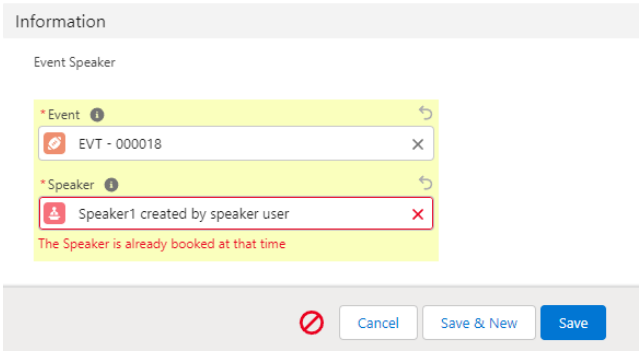


Figura 37 - Resultado da ativação do "Event Speaker Trigger".

O código deste *trigger* pode ser consultado no Anexo E.

7.4.2 Event Attendee Trigger

A função deste *trigger* é a de enviar um email de confirmação, com um formato pré-definido ao participante, quando este submete a inscrição num evento, ou seja, quando cria um registo no objeto "Event Attendee". Esta funcionalidade foi implementada usando uma classe Apex (*handler*) que é invocada pelo *trigger*. Esta classe também invoca a classe "Transaction Log Handler" que lida com os erros e exceções. A figura 38 mostra o email enviado.

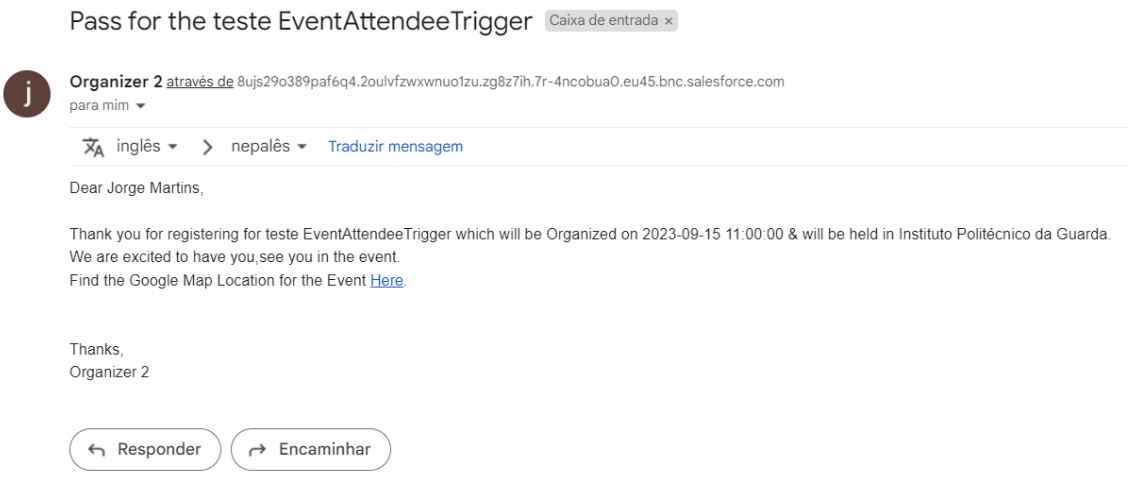


Figura 38 - Resultado da ativação do "Event Attendee Trigger".

Os códigos do *trigger* e da sua classe *hanlder* podem ser consultados no Anexo F e G respetivamente.

7.4.3 Event Attendee Trigger Test

Antes de implementar um *trigger* numa Org deve-se testar o seu funcionamento, um teste tem de cobrir pelo menos 75% do código (figura 39). Para isso é necessário criar testes de unidade que consistem em blocos de código onde, criando dados de teste, se verifique se o desempenho do *trigger* é o desejável. Estes testes correm na *Developer Console*.

Para o "Event Attendee Trigger" foi desenvolvida uma classe de teste onde foram criados registos de teste para os objetos "Event Organizer", "EventMF" e "Attendee" de modo que se pudesse criar um registo no objeto "Event Attendee" e verificar que era enviado um email.

Overall Code Coverage

Class	Percent	Lines
EventAttendeeTrigger	100%	2/2
EventAttendeeTriggerHandler	93%	40/43

Figura 39 - Painel da Developer Console relativo à cobertura do teste ao trigger.

O código deste teste pode ser consultado no Anexo H.

7.4.4 Event Attendee Not Duplicate Trigger

A função deste *trigger* é a de impedir que seja criado um registo no objeto "Event Attendee" em duplicado. No caso de um utilizador tentar introduzir um registo nestas condições é exibida uma mensagem de erro na IU (figura 40).

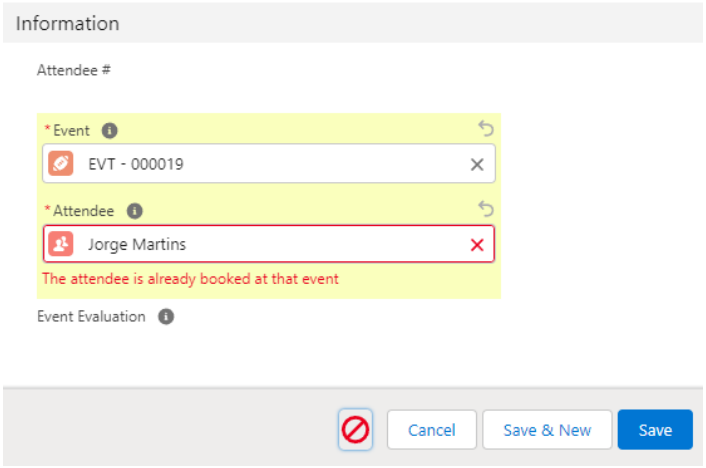


Figura 40 - Resultado da ativação do "Event Attendee Not Duplicate Trigger".

Como se pode verificar nos Anexos I e J, o *trigger* invoca uma classe (*handler*) que contém o método responsável pela sua funcionalidade.

7.4.5 Event Attendee Not Duplicate Trigger Test

Foi também criada uma classe de teste para o *trigger* (Anexo K) onde foram criados dados e dois métodos de teste. O primeiro método faz o teste para um só registo duplicado enquanto o segundo método faz o teste para duzentos registos que é o número de registos que são tratados de cada vez quando são importados dados em grande escala para o Salesforce. O teste teve sucesso (figura 41) e consistiu em verificar que não era possível criar registos duplicados no objeto "Event Attendee".

Overall Code Coverage

Class	Percent	Lines
EventAttendeeNotDuplicateTrigger	100%	1/1
EventAttendeeNotDuplicateTriggerHandler	82%	19/23

Figura 41 - Painel da Developer Console relativo à cobertura do teste do trigger.

8. Apresentação da App (2ª fase)

Para a apresentação da App à entidade acolhedora foram elaborados dois vídeos, o primeiro corresponde à apresentação do que era pretendido implementar e o segundo corresponde à demonstração das configurações feitas e das funcionalidades implementadas na App. Os dois vídeos podem ser acedidos pelo link: https://drive.google.com/drive/folders/1EmTd4VmtEgymsq-rlbIKbCjNG060b9lu.

8.1 Requisitos

Neste vídeo é feita uma exposição dos requisitos da App, são apresentados e discutidos todos os objetos que a constituem, o modelo de dados e os campos, bem como as regras de validação e duplicação dos dados. São também discutidas todas as permissões aos objetos e campos mencionando o caminho para as implementar. Por fim são apresentadas as funcionalidades pretendidas, nomeadamente os *triggers*, o *flow* e o *dashboard*.

8.2 Demonstração

Com este vídeo pretendeu-se mostrar tudo o que foi implementado na App Salesforce. Em primeiro lugar foram testadas as regras de validação e de duplicidade e dados. As figuras 42 e 43 mostram, a título de exemplo, estes testes. Em seguida são abordadas as configurações de perfis, utilizadores e funções (*roles*) em que se baseiam as permissões de acesso aos dados (figura 44).

No que toca a funcionalidades implementadas, a apresentação mostra o funcionamento dos três *triggers*, do *flow* de avaliação de evento e do *dashboard* localizado na *home page* de um utilizador cujo perfil seja "Event Organizer".

A título de exemplo a figura 45 mostra a ativação do "Event Attendee Not Duplicate Trigger", a figura 46 mostra

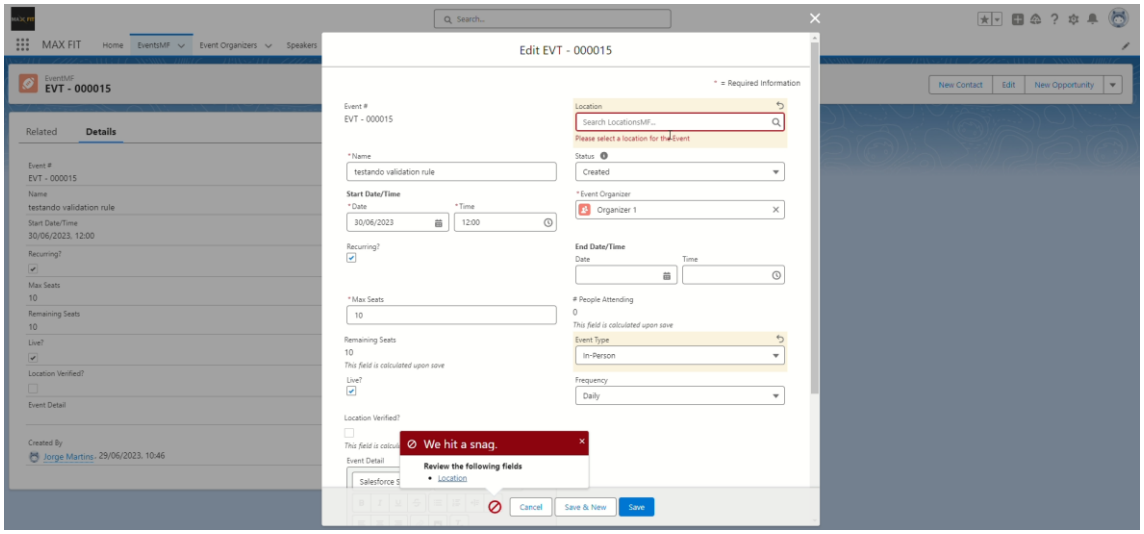


Figura 42 - Teste de regra de validação de dados na interface de utilizador.

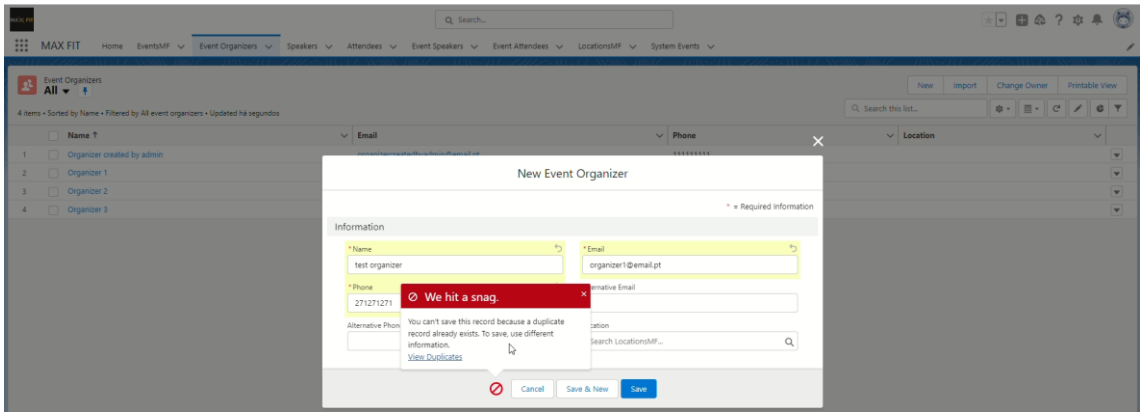


Figura 43 - Teste de regra de duplicidade de dados na interface do utilizador.

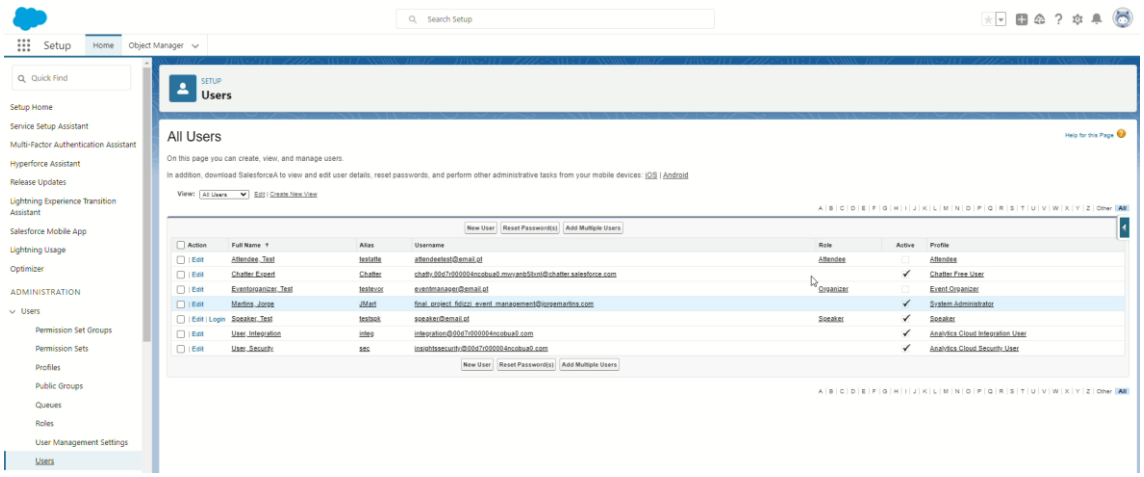


Figura 44 – Configuração de utilizadores tipo mostrando o perfil e função (*role*).

No que toca a funcionalidades implementadas, a apresentação mostra como funcionam os três *triggers* e o *flow* para a avaliação de eventos.

A título de exemplo a figura 45 mostra a ativação do "Event Attendee Not Duplicate Trigger", a figura 46 mostra um registo do objeto "Event Attendee" com o painel de avaliação do evento gerado pelo *flow* implementado.

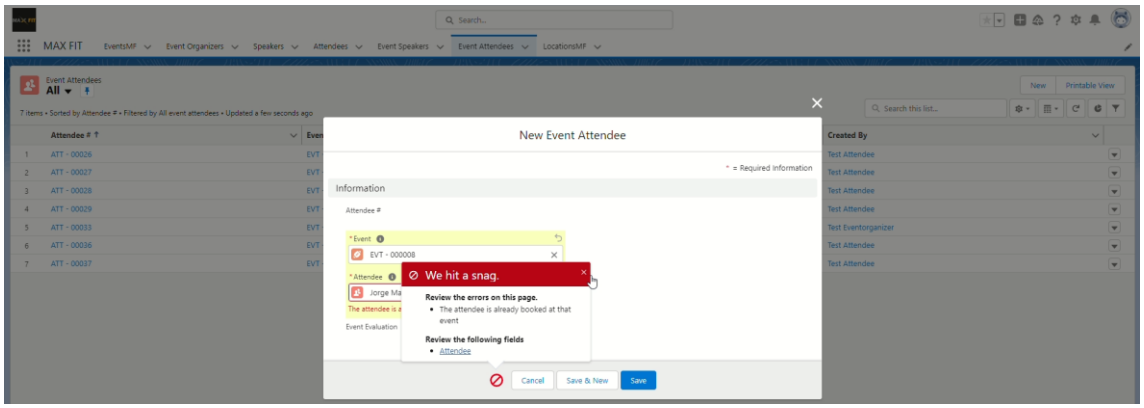


Figura 45 – Testando o "Attendee Not Duplicate Trigger" na interface do utilizador.

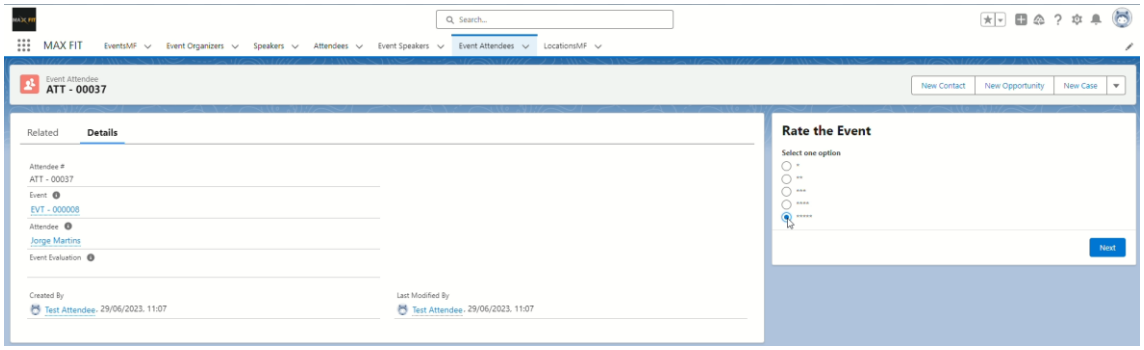


Figura 46 - Registo no objeto "Event Attendee" mostrando o painel gerado pelo "Event Evaluation Flow".

É também apresentado o *dashboard* localizado na *home page* de um utilizador cujo perfil seja "Event Organizer" (figura 47).

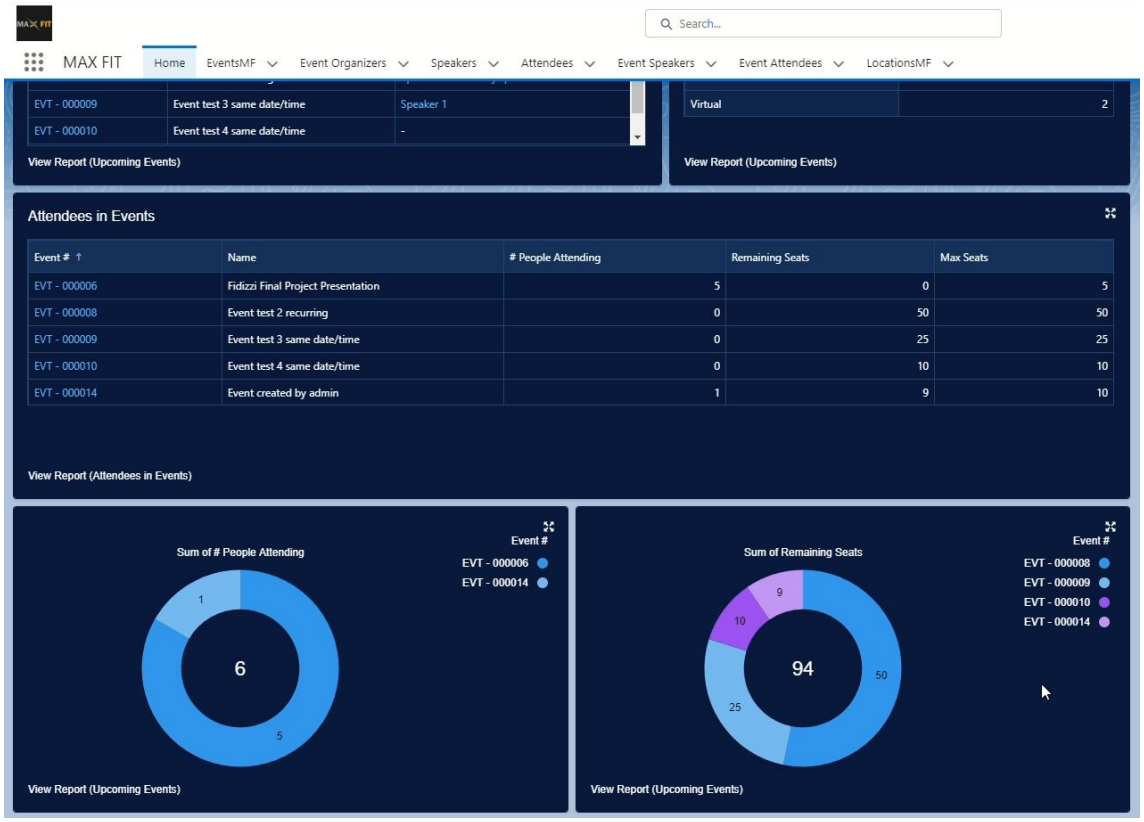


Figura 47 - Dashboard da App MAXFIT.

Para finalizar a apresentação é feita uma breve abordagem aos códigos em Apex usados nas implementações (figura 48).

```
public class EventAttendeeNotDuplicateTriggerHandler {

    private static Set<Id> attendeeIdSet = new Set<Id>();
    private static Set<Id> eventIdSet = new Set<Id>();
    private static Set<ID> eventAttendeesIdToUpdateSet = new Set<Id>();//no caso de update os valores 'velhos' podem ser incluídos na lista registeredEvtAtts por isso é criado
                                                        //este set para ser usado como filtro e não serem incluídos estes valores na lista

    public static void NoDuplicateEventAttendee(List<Event_Attendee__c> newRecords){

        for (Event_Attendee__c evt:newRecords){

            attendeeIdSet.add(evt.attendee__c);

            eventIdSet.add(evt.Event__c);

            //este set é criado só no caso de update, e servirá como filtro na construção da lista registeredEvtAtts para que não sejam incluídos os registos que já constam
            //na BD (Trigger.old) e assim não haver elementos duplicados nessa lista, o filtro é feito pelo Id pois esses registos, como já constam na BD já possuem Id
            //ao contrário dos novos registos(trigger.new) que, como ainda não foram inseridos não possuem Id
            //if(Trigger.isUpdate){
                //eventAttendeesIdToUpdateSet.add(evt.Id);
            //}
        }


        List <Event_Attendee__c> registeredEvtAtts = [SELECT Event__c, Attendee__c FROM Event_Attendee__c
                                                      WHERE Event__c IN :eventIdSet AND Attendee__c IN :attendeeIdSet];
//se o trigger fosse tb de update usava WHERE Event__c IN :eventIdSet AND Attendee__c IN :attendeeIdSet AND Id NOT IN :eventAttendeesIdToUpdateSet];
//este filtro do Id causou problemas no flow quando fiz updates no campo event evaluation__c (o record final tinha parâmetros iguais id attendee e id de event porque quando foi
```

Figura 48 - *Developer console* com os códigos em Apex.

9. Conclusão

A realização deste estágio permitiu adquirir competências importantes para a área das tecnologias da informação, entre elas estão a capacidade de pesquisa, autonomia e comunicação. O contexto de trabalho remoto permitiu experienciar a realidade nas empresas nos dias de hoje.

A fase de aprendizagem foi bastante extensa e intensiva, além dos conteúdos disponibilizados no Trailhead foi necessário fazer muitas pesquisas passando por documentação técnica do próprio Salesforce, *websites* especializados em Salesforce, vídeos tutoriais e cursos *on-line*. O roteiro proposto (Trailmix) pela entidade acolhedora foi cumprido ligeiramente abaixo dos 90% por razões de calendário, no entanto, segundo a opinião do tutor da entidade acolhedora era algo que seria de esperar, uma vez que o roteiro era deveras extenso e os conteúdos finais (*Lightning Web Components*) eram de complexidade mais avançada. Apesar de o Trailmix não ter sido completado na sua totalidade, o *feedback* foi muito positivo.

A fase de desenvolvimento de uma App Salesforce permitiu pôr em prática e consolidar os conhecimentos adquiridos, tendo sido também necessária muita pesquisa adicional. Enquanto os desafios práticos que eram propostos na fase de aprendizagem contavam com exemplos e ajuda passo a passo, a implementação desta App em algumas das funcionalidades, nomeadamente a criação de *triggers* e de unidades de teste, exigiu um estudo mais aprofundado. Esta fase foi também limitada por razões de calendário e isso determinou a quantidade de funcionalidades implementadas, no entanto foi possível desenvolver capacidades ao nível das configurações numa Org Salesforce, criação de uma App Salesforce e implementação de funcionalidades *low-code/no-code* bem como funcionalidades *hard-code*.

Sendo assim, pode concluir-se que os objetivos do estágio foram alcançados, o estagiário conseguiu ter uma visão geral das potencialidades e desenvolver soluções na Plataforma CRM Salesforce.

Visto que se perspetiva uma grande procura por profissionais credenciados em Salesforce, o caminho a seguir deverá passar pela obtenção de certificações.

10. Webgrafia

[1] Temos mais dados do que nunca, Como usá-los a nosso favor?. Exame, 2021. Disponível em: https://exame.com/carreira/dados-uso-favor/. (Acedido em: 13/07/2023)

[2] Customer Relationship Management Software - Worldwide. Statista, 2023. Disponível em: https://www.statista.com/outlook/tmo/software/enterprise-software/customer-relationship-management-software/worldwide#analyst-opinion. (Acedido em 13/07/2023)

[3] Customer Relationship Management (CRM) Software. TrustRadius. Disponível em: https://www.trustradius.com/crm. (Acedido em: 07/07/2023)

[4] LEES, Harry. 50 Crucial CRM Statistics for the 2021 Market. TrustRadius, 2021. Disponível em: www.trustradius.com/vendor-blog/crm-statistics-trends. (Acedido em: 06/07/23)

[5] GANTZ, John; WEBBER, Alan. The Salesforce Economic Impact: 9 Million New Jobs by 2026, $1.6 Trillion of New Revenues for Customers, 2021. Disponível em: https://www.salesforce.com/content/dam/web/en_us/www/documents/reports/idc-salesforce-economy-report.pdf. (Acedido em: 14/07/2023)

[6] Salesforce vs. Microsoft Dynamics CRM: Frente a frente. Liminal, 2020. Disponível em: https://liminal.pt/martech-magazine/salesforce-vs-microsoft-dynamics-365-sales-crm. (Acedido em: 07/07/2023)

[7] Ungureanu, A., & Ungureanu, A. (2014). Methodologies used in project management. Annals of Spiru Haret University, Economic Series, 5(2), 47-53. Disponível em: https://core.ac.uk/download/pdf/267905655.pdf. (Acedido em: 17/07/2023)

[8] Agile Manifesto. Airfocus Glossary. Disponível em: https://airfocus.com/glossary/what-is-agile-manifesto/. (Acedido em: 23/08/2023)

[9] Agile Manifesto. Scrum Portugal. Disponível em: https://www.scrumportugal.pt/agile-manifesto. (Acedido em: 15/07/2023)

[10] Miguel, A. (2019). Gestão Moderna de Projetos - Melhores Técnicas e Práticas, 8ª edição, FCA.

[11] Scrum or Framework Ágil Scrum. Scrum Portugal. Disponível em: https://www.scrumportugal.pt/scrum/. (Acedido em 15/07/2023)

[12] What is Scrum? Scrum.org. Disponível em: https://www.scrum.org/learning-series/what-is-scrum. (Acedido em: 02/08/2023)

[13] Glossary. Salesforce. Disponível em: https://help.salesforce.com/s/articleView?id=sf.glossary.htm&type=5. (Acedido em: 11/07/2023)

[14] MANN, Jason. Salesforce organizations. Gearset. Disponível em: https://docs.gearset.com/en/articles/2285301-salesforce-organizations. (Acedido em: 20/07/2023)

[15] Types of Organizations in Salesforce? Salesforce Tutorial. Disponível em: https://www.salesforcetutorial.com/types-of-organizations-in-salesforce. (Acedido em: 20/07/2023)

[16] Salesforce Object Query Language (SOQL). Salesforce. Disponível em: https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql.htm. (Acedido em 11/07/2023)

[17] What is Apex? Salesforce. Disponível em: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_what_is_apex.htm. (Acedido em: 11/07/2023)

[18] Developer Console. Salesforce. Disponível em: https://help.salesforce.com/s/articleView?id=sf.code_dev_console.htm&type=5. (Acedido em: 10/07/2023)

[19] Install Development Tools. Trailhead. Disponível em: https://trailhead.salesforce.com/content/learn/projects/set-up-your-lightning-web-components-developer-tools/install-development-tools. (Acedido em: 09/07/2023)

[20] ARGAWAL, Chitiz. What is Salesforce CLI? Salesforce DX CLI. Techila, 2021. Disponível em: https://techilaservices.com/blog/salesforce-cli. (Acedido em: 13/07/2023)

[21] Set Up Your Salesforce DX Environment. Trailhead. disponível em: https://trailhead.salesforce.com/content/learn/projects/quick-start-lightning-web-components/set-up-salesforce-dx. (Acedido em: 09/07/2023)

[22] Create a Visualforce Page. Trailhead. Disponível em: https://trailhead.salesforce.com/content/learn/projects/quickstart-visualforce/vf-qs-1. (Acedido em: 11/07/2023)

[23] What is VisualForce? Visualforce Developer Guide. Disponível em: https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_intro_what_is_it.htm?q=what+is+visualforce. (Acedido em: 11/07/2023)

[24] CARVALHINHO, João (2018). Relatório de Estágio. Disponível em: https://bdigital.ipg.pt/dspace/bitstream/10314/4481/1/Jo%c3%a3o%20Carvalhinho_1012366.pdf (Acedido em 05/07/2023)

[25] HTML Introduction. W3 Schools. Disponível em: https://www.w3schools.com/html/html_intro.asp. (Acedido em: 11/07/2023)

ANEXOS

EAnexo A. Regras de validação

Objeto “EventMF”

Validation Rule Detail

Edit | Clone

Rule Name	End_Date_must_be_Greater_than_Start_Date
Error Condition Formula	AND (NOT ISBLANK (End_Date_Time__c), End_Date_Time__c < (Start_DateTime__c + 1))
Error Message	End Date/Time must be at least 1 day ahead of Start Date/Time
Description	Validates if the End Date is at least 1 day ahead of Start Date (in case that there is a value for End Date)

Validation Rule Detail

Rule Name	Select_location_for_InPerson
Error Condition Formula	AND(TEXT (Event_Type__c) = 'In-Person', ISBLANK (Location__c))
Error Message	Please select a location for the Event
Description	If Event type is "In-person" must add Location

Validation Rule Detail

Edit | Clone

Rule Name	If_virtual_location_blank
Error Condition Formula	AND(TEXT (Event_Type__c) = 'Virtual', NOT ISBLANK (Location__c))
Error Message	You can't select a Location if the Event is Virtual
Description	If the Event Type field is "virtual" Location field must be blank

Validation Rule Detail

Edit | Clone

Rule Name	Must_Select_Frequency
Error Condition Formula	AND(Recurring__c , ISBLANK(TEXT(Frequency__c)))
Error Message	You must need to Select Frequency field for Recurring events
Description	if Recuring checkbox is checked must fill Frequency field. If unchecked do not select Frequency field

Validation Rule Detail Edit Clone

Rule Name	You_can_not_Select_Frequency
Error Condition Formula	AND(NOT Recurring__c , NOT ISBLANK(TEXT (Frequency__c)))
Error Message	You can't Select Frequency for Non-Recurring Events
Description	if Recurrent checkbox is not selected you can't select a Frequency

Objeto "Event Speaker"

Validation Rule Detail Edit Clone

Rule Name	Event_is_Not_Live_or_Has_been_Organized
Error Condition Formula	OR (NOT Event__r.Live__c, Event__r.End_Date_Time__c < NOW())
Error Message	if Live checkbox is not checked or the End Date of the event has already passed, the speaker can't be associated to the event
Description	if Live checkbox is not checked or the End Date of the event has already passed, the speaker can't be associated to that event

Objeto "Event Attendee"

Validation Rule Detail Edit Clone

Rule Name	Event_is_Not_Accepting_Registrations
Error Condition Formula	AND(OR (NOT Event__r.Live__c , Event__r.End_Date_Time__c < NOW() , Event__r.Remaining_Seats__c <=0), ISNEW())
Error Message	Event is Not Accepting Registrations. The End date/time of the event has passed or the live checkbox field (event object) is not checked or there are no seats available
Description	tells the user that Event is Not Accepting Registrations. The End date/time of the event has passed or the live checkbox field (event object) is not checked or there are no seats available

Anexo B. Regras de correspondência

Matching Rule
Event Organizer Matching rule

Matching Rule Detail Delete Clone Deactivate

Object	Event Organizer
Rule Name	Event Organizer Matching rule
Unique Name	Event_Organizer_Matching_rule
Description	Matching Rule for the MAX FIT App project
Matching Criteria	(Event Organizer: Email EXACT MatchBlank = FALSE) AND (Event Organizer: Phone EXACT MatchBlank = FALSE)
Status	Active

Matching Rule
Speaker Matching rule

Matching Rule Detail Delete Clone Deactivate

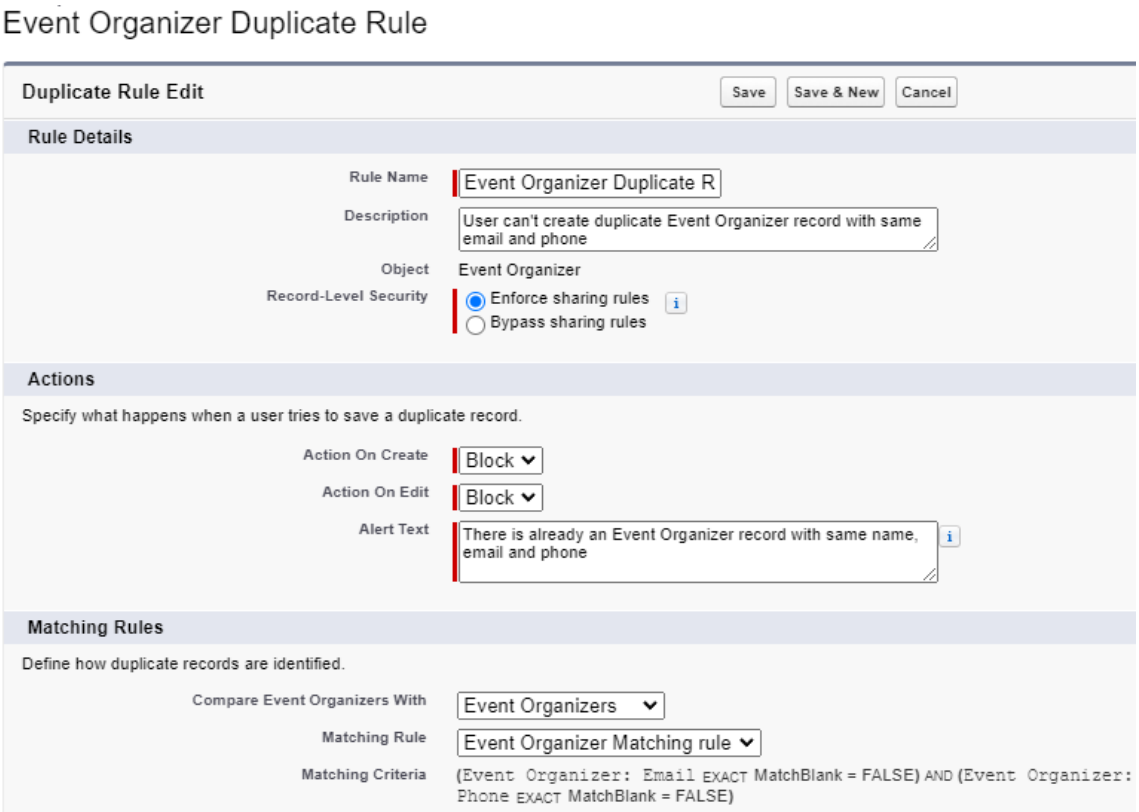
Object	Speaker
Rule Name	Speaker Matching rule
Unique Name	Match_Speaker
Description	Rule for matching speaker email and phone to use in duplicate rules
Matching Criteria	(Speaker: Email EXACT MatchBlank = FALSE) AND (Speaker: Phone EXACT MatchBlank = FALSE)
Status	Active

Matching Rule
Attendee Matching rule

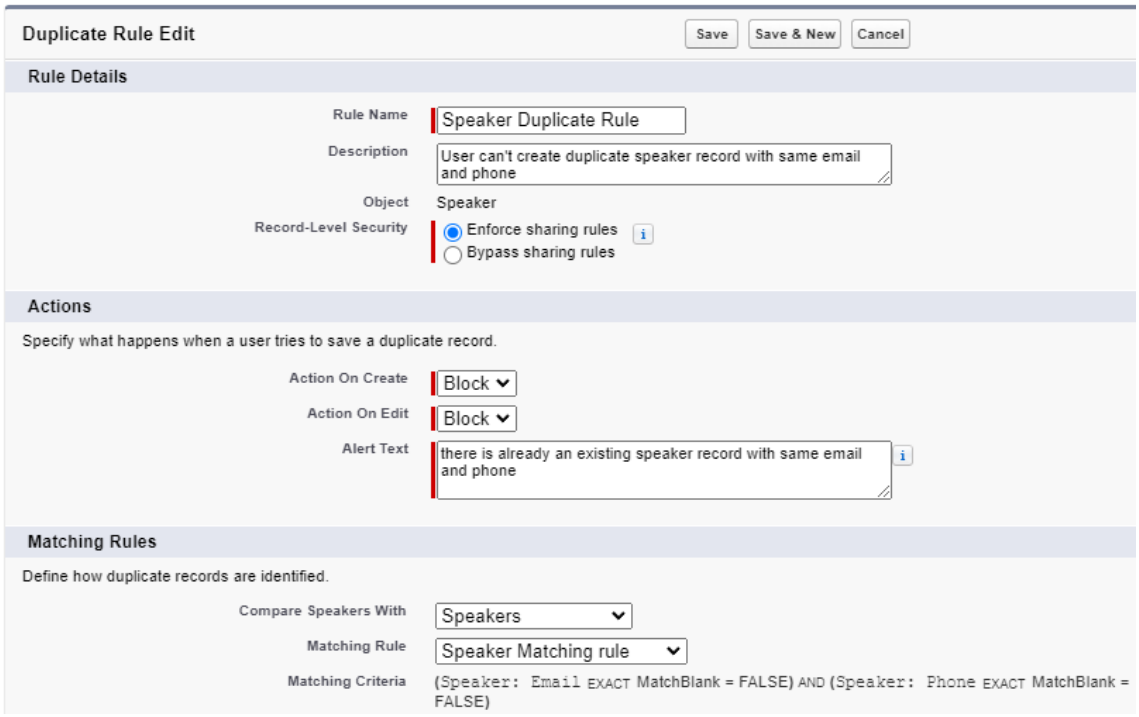
Matching Rule Detail Delete Clone Deactivate

Object	Attendee
Rule Name	Attendee Matching rule
Unique Name	Attendee_Matching_rule
Description	MAtching rule for MAX FIT APP project Attendee object for matching email, name, and phone so i can use it in a duplicate rule
Matching Criteria	(Attendee: Email EXACT MatchBlank = FALSE) AND (Attendee: Name EXACT MatchBlank = FALSE) AND (Attendee: Phone EXACT MatchBlank = FALSE)
Status	Active

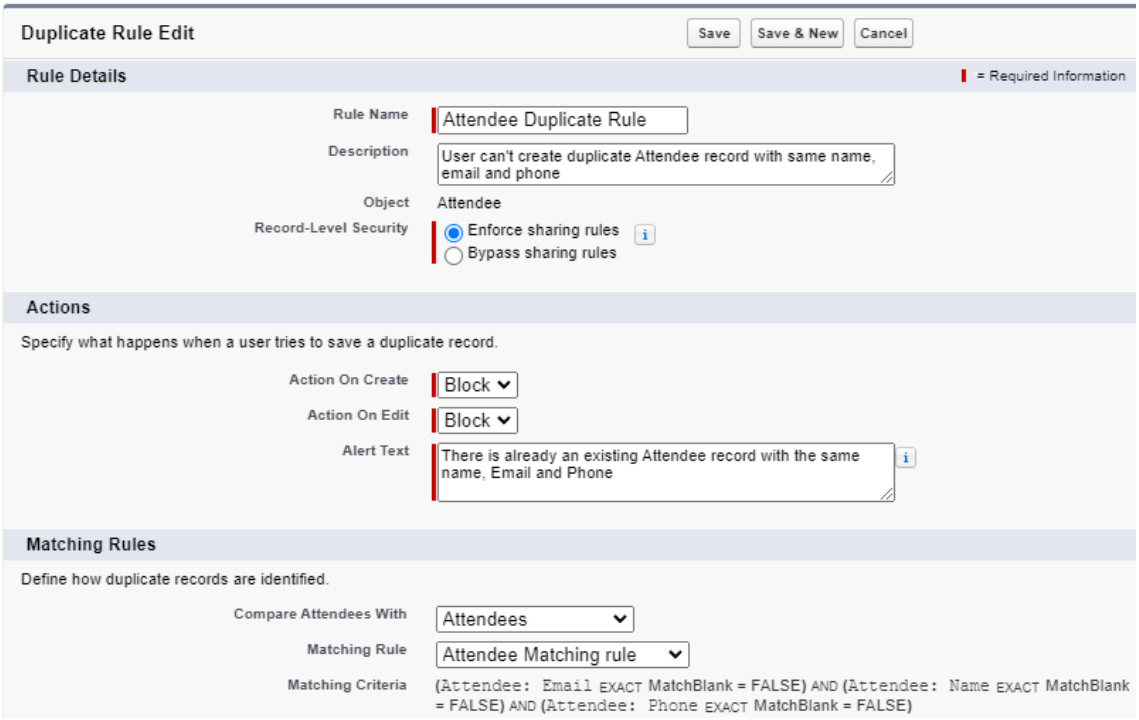
Anexo C. Regras de duplicidade



Speaker Duplicate Rule



Attendee Duplicate Rule



Anexo D. Código do "Transaction Log Handler"

```
public class TransactionLogHandler {


    private static List<Error_Log__c> errorsList = new
List<Error_Log__c>();


    public static void storeLogs(){
        if (errorsList != null && errorsList.size() > 0 ){
            insert errorsList;
        }
    }


    public static void doHandleException(System.Exception ex
, String processName){
          Error_Log__c  transactionLog = new Error_Log__c (
          Log_Details__c  = ex.getStackTraceString() +'<br/>
     <strong> Message: </strong> ' + ex.getMessage()
            +  '<br/>     <strong>  Cause:  </strong>  '+
ex.getCause() +' <br/><strong> Type Name: </strong>  '+
            ex.getTypeName()+' <br/> <strong> Line Number:
</strong> ' +ex.getLineNumber(),
            Lod_DateTime__c  = System.Now(),
            Process_Name__c    = processName
        );
        system.debug (transactionLog);
        errorsList.add(transactionLog);
    }


    public  static  void  doHandleExceptionWithError(String
errors , String processName){
```

```
        Error_Log__c  transactionLog = new Error_Log__c (
            Log_Details__c = errors,
            Lod_DateTime__c = System.Now(),
            Process_Name__c = processName
        );
        errorsList.add(transactionLog);
    }
}
```

Anexo E. Código do "Event Speaker Trigger"

```
trigger EventSpeakerTrigger on EventSpeakers__c (before
insert,before Update) {
    Set<Id> speakerId = new Set<Id>();
    Set<Id> eventIdsSet = new Set<Id>();

    Map<Id, DateTime> requestedEvents = new Map<Id,
DateTime>();
     Map<Id,List<Datetime>> eventDataMap = new
Map<Id,List<Datetime>>();

    for(EventSpeakers__c evt:Trigger.new){
        speakerId.add(evt.Speaker__c);
        eventIdsSet.add(evt.Event__c);
    }

    List<Event__c> relatedEventList = [Select Id,
Start_DateTime__c From Event__c Where Id IN : eventIdsSet];

    for(Event__c evt : relatedEventList ){
        requestedEvents.put(evt.Id, evt.Start_DateTime__c);
    }

    List<EventSpeakers__c> ESrelatedList = [select
id,Event__c,Speaker__c,Event__r.Start_DateTime__c from
EventSpeakers__c where Speaker__c IN: speakerId];

    System.debug(EsrelatedList);

    for(EventSpeakers__c esitr: ESrelatedList){
```

```
        if(eventDataMap.containskey(esitr.Speaker__c)){
          eventDataMap.get(esitr.Speaker__c).add(esitr.Even
     t__r.Start_DateTime__c);
        }
        else{
            eventDataMap.put(esitr.Speaker__c,          new
List<Datetime>{esitr.Event__r.Start_DateTime__c});
        }
    }
    System.debug(eventDataMap);




    for(EventSpeakers__c eventspeaker: Trigger.new){
        DateTime                bookingTime                =
requestedEvents.get(eventspeaker.Event__c);

if(eventDataMap.containsKey(eventspeaker.Speaker__c)     &&
eventDataMap.get(eventspeaker.Speaker__c).contains(bookingT
ime)){
            eventspeaker.Speaker__c.addError('The   Speaker
is already booked at that time');
            eventspeaker.addError('The  speaker  is  already
booked at that time');
        }
    }
}
```

Anexo F. Código do "Event Attendee Trigger"

```
trigger EventAttendeeTrigger on Event_Attendee__c (after
insert)                                                    {
EventAttendeeTriggerHandler.sendConfirmationEmail(Trigger.N
ew);

    }

}
```

Anexo G. Código do "Event Attendee Trigger Handler"

```
public class EventAttendeeTriggerHandler {

    public static void sendConfirmationEmail(List<Event_Attendee__c> newRecordList) {

        Set<Id> attendeesIdsSet = new Set<Id>();
        Set<Id> eventIdsSet = new Set<Id>();

        for(Event_Attendee__c ea : newRecordList){
          attendeesIdsSet.add(ea.Attendee__c);
            eventIdsSet.add(ea.Event__c);
        }

        Map<Id,Attendee__c> attendeeMap = new Map<Id,Attendee__c>(
            [Select Id, Name, Email__c From Attendee__c WHERE Id IN : attendeesIdsSet]
        );

        Map<Id, Event__c > eventMap = new Map<Id, Event__c > (
            [Select Id, Name__c, Start_DateTime__c, Event_Organizer__c , Event_Organizer__r.Name,
             Location__c , Location__r.Name, Location__r.City__c,Location__r.Country__c,Location__r.Postal_Code__c, Location__r.Street__c FROM Event__c WHERE ID IN: eventIdsSet]
        );
```

```
        List<Messaging.SingleEmailMessage> emailList = new
List<Messaging.SingleEmailMessage>();

        for(Event_Attendee__c ea : newRecordList){

            Attendee__c                att                =
attendeeMap.get(ea.Attendee__c);

            Event__c evt = eventMap.get(ea.Event__c);

            Messaging.SingleEmailMessage    mail    =    new
Messaging.SingleEmailMessage();

            mail.setSubject('Pass for the '+evt.Name__c);

            List<String> toAddress = new List<String>();

            toAddress.add(att.Email__c);

            mail.setToAddresses( toAddress );

            mail.setSenderDisplayName
(evt.Event_Organizer__r.Name);

            String                location                =
'https://www.google.com/maps/place/'+evt.Location__r.Street
__c+'                          '+evt.Location__r.City__c+'
'+evt.Location__r.Country__c+'                                '+
evt.Location__r.Postal_Code__c;

            String       hrefForLocation       =       '<a
href="'+location+'"'+'target="_blank">Here</a>';

            String   emailBody   =   'Dear   '+   att.Name   +
',<br/><br/>'+'Thank you for registering for '+evt.Name__c+'
which will be Organized on '+evt.Start_DateTime__c+' & will
be held in '+evt.Location__r.Name+'.<br/>We are excited to
have you,'+ 'see you in the event. <br/>'+'Find the Google
Map         Location         for         the         Event
'+hrefForLocation+'.<br/><br/><br/>'+

'Thanks,<br/>'+evt.Event_Organizer__r.Name;


            mail.setHtmlBody(emailBody);

           emailList.add(mail);

        }

        try{
```

```
            List<Messaging.SendEmailResult>    results    =
Messaging.sendEmail(emailList, false);

            for(Messaging.SendEmailResult email : results){

                System.debug(email.isSuccess());

                if(!email.isSuccess()){

                    List<Messaging.SendEmailError> errors =
email.getErrors();

               TransactionLogHandler.doHandleExceptionWithE
               rror                (JSON.serialize(errors),
               'EventAttendeeTriggerHandler');


                }

            }

            TransactionLogHandler.storeLogs();


        }catch(System.Exception ex){

            TransactionLogHandler.doHandleException(ex,
'EventAttendeeTriggerHandler');

            TransactionLogHandler.storeLogs();

        }

    }

}
```

Anexo H. Código do "Event Attendee Trigger Test"

```
@isTest
public class EventAttendeeTriggerTest {

    @testSetup
    public static void setupData(){

        Event_Organizer__c  org = new Event_Organizer__c (
            Name = 'John Doe',
            Phone__c = '980765432',
            Email__c = 'email@email.pt',
            Alternative_Phone__c  = '960765432',
            Alternative_Email__c ='email@gmail.com'
        );
        insert org;

        Event__c event = new Event__c(
            Name__c = 'MAX FIT Campaign',
            Event_Organizer__c  = org.Id,
            Event_Type__c = 'Virtual',
            Frequency__c  = 'Weekly',
            Max_Seats__c  = 199,
            Recurring__c = true,
            Live__c = true,
            Start_DateTime__c = System.now(),
            End_Date_Time__c  = System.now().addDays(3)
        );
        insert event;
```

```
        Attendee__c att = new Attendee__c(
            Name = 'Jane Doe',
            Email__c = 'jane@email.pt',
            Phone__c = '939495789'
        );
        insert att;
    }

    @isTest
    static void sendEmailTest(){
        Event__c evt = [SELECT Id FROM Event__c];
        Attendee__c att =[SELECT Id FROM Attendee__c];

        Test.startTest();
        Event_Attendee__c evtAtt = new
Event_Attendee__c(Event__c = evt.Id, Attendee__c = att.Id);
        insert evtAtt;

        System.assertEquals(1,
Limits.getEmailInvocations(), 'Email should be sent');

        Test.stopTest();
    }
}
```

Anexo I. Código do "Event Attendee Not Duplicate Trigger"

```
trigger EventAttendeeNotDuplicateTrigger on
Event_Attendee__c (before insert) {

EventAttendeeNotDuplicateTriggerHandler.NoDuplicateEventAtt
endee(Trigger.New);

}
```

Anexo J. Código do "Event Attendee Not Duplicate Trigger Handler"

```
public class EventAttendeeNotDuplicateTriggerHandler {

    private static Set<Id> attendeeIdSet = new Set<Id>();
    private static Set<Id> eventIdSet = new Set<Id>();

    public static void NoDuplicateEventAttendee(List<Event_Attendee__c> newRecords){

        for (Event_Attendee__c evt:newRecords){
            attendeeIdSet.add(evt.attendee__c);
            eventIdSet.add(evt.Event__c);
        }

        List <Event_Attendee__c> registeredEvtAtts = [SELECT Event__c, Attendee__c FROM Event_Attendee__c WHERE Event__c IN : eventIdSet AND Attendee__c IN :attendeeIdSet];

        Map<Id,Map<Id,String>> eventAttendeeMap = new Map<Id,Map<Id,String>>();

        for(Event_Attendee__c evat: registeredEvtAtts){

            if (eventAttendeeMap.containsKey(evat.Event__c)){
eventAttendeeMap.get(evat.Event__c).put(evat.Attendee__c, evat.Attendee__c);
            }

            else{
```

```
                eventAttendeeMap.put(evat.Event__c,     new
Map<Id,String>{evat.Attendee__c => evat.Attendee__c });
            }
        }

        try{
            for (Event_Attendee__c ea : newRecords){
                Map<Id,String>       attendeeInMap       =
eventAttendeeMap.get(ea.Event__c);

                if      (attendeeInMap!=      null      &&
String.isNotBlank(attendeeInMap.get(ea.Attendee__c))){
                    ea.Attendee__c.addError('The   attendee
is already booked at that event');
                }
            }
        } catch(System.Exception ex){
            TransactionLogHandler.doHandleException(ex,
'EventAttendeeTriggerHandler');
            TransactionLogHandler.storeLogs();
        }
    }
}
```

Anexo K. Código do "Event Attendee Not Duplicate Trigger Test"

```
@isTest
public class EventAttendeeNotDuplicateTriggerTest {

    //criar dados de teste
    @testSetup
    public static void setupTestData(){

        //criar dados para testar a inserção de um só registo duplicado

        Event_Organizer__c  orgTest = new Event_Organizer__c (
            Name                =                'Organizer testEvetAttNoDuplicateTrigger',
            Phone__c = '9807654321',
            Email__c                                        = 'organizertestEvetAttNoDuplicateTriggerTest@email.pt'
        );
        insert orgTest;

        Event__c eventTest = new Event__c(
            Name__c                 =                'Event EventAttendeeNotDuplicateTriggerTest ',
            Event_Organizer__c  = orgTest.Id,
            Event_Type__c = 'Virtual',
            Frequency__c  = 'Weekly',
            Max_Seats__c  = 20,
            Recurring__c = true,
            Live__c = true,
```

```
            Start_DateTime__c = System.now(),
            End_Date_Time__c  = System.now().addDays(3)
        );
        insert eventTest;

        Attendee__c attTest = new Attendee__c(
            Name                =                'Attendee
EventAttendeeNotDuplicateTriggerTest',
            Email__c                                        =
'attendeetestEvetAttNoDuplicateTriggerTest@email.pt',
            Phone__c = '9807654321'
        );
        insert attTest;

        //criar um registo Event_Attendee__c que vai servir
de comparação à nova inserção de registo duplicado
        Event_Attendee__c evtatt = new Event_Attendee__c(
            Event__c = eventTest.Id,
            Attendee__c = attTest.Id
        );
        insert evtatt;

        //criar dados para testar a inserção de vários (Bulk)
registos duplicados

        List<Event_Organizer__c>   evtOrgBulkList   =   new
List<Event_Organizer__c>();
        List<Event__c> evtBulkList = new List<Event__c>();
        List<Attendee__c>      attBulkList      =      new
List<Attendee__c>();

        for (integer i=0; i<199;i++){
```

```
            Event_Organizer__c       orgTestBulk    =    new
Event_Organizer__c (
                Name            =            'OrganizerBulk
testEvetAttNoDuplicateTrigger'+i,
                Phone__c = '980765432'+i,
                Email__c                                    =
'organizerBulktestEvetAttNoDuplicateTriggerTest'+i+'@email.
pt'
            );
            evtOrgBulkList.add(orgTestBulk);
        }
        insert evtOrgBulkList;


        for (Event_Organizer__c eo: evtOrgBulkList){
            integer i = 0;
            Event__c eventTestBulk = new Event__c(
                Name__c               =               'Event
EventAttendeeNotDuplicateTriggerTest'+i,
                Event_Organizer__c  = eo.Id,
                Event_Type__c = 'Virtual',
                Frequency__c  = 'Weekly',
                Max_Seats__c  = 2000,
                Recurring__c = true,
                Live__c = true,
                Start_DateTime__c = System.now(),
                End_Date_Time__c  = System.now().addDays(3)
            );
            evtBulkList.add(eventTestBulk);

            Attendee__c attTestBulk = new Attendee__c(
```

```
                Name                =                'Attendee
EventAttendeeNotDuplicateTriggerTest'+i,
                Email__c                                    =
'attendeetestEvetAttNoDuplicateTriggerTest'+i+'@email.pt',
                Phone__c = '9807654321'
            );
            attBulkList.add(attTestBulk);
            i+=1;
        }
        insert evtBulkList;
        insert attBulkList;

        List<Event_Attendee__c>   evtAttBulkList   =   new
List<Event_Attendee__c>();
        for (Integer i = 0; i<199; i++){
            Event_Attendee__c     evtattBulk     =     new
Event_Attendee__c(
                Event__c = evtBulkList[i].Id,
                Attendee__c = attBulkList[i].Id);
            evtAttBulkList.add(evtattBulk);
        }
        insert evtAttBulkList;
    }

//método de teste com um registo
    @isTest
    static void notCreateDuplicateEvetAtt(){

       Event_Attendee__c   eaTest   =   [SELECT   Event__c,
Attendee__c FROM Event_Attendee__c LIMIT 1];
```

```
        Event_Attendee__c ea = new Event_Attendee__c(
        event__c = eaTest.Event__c,
        Attendee__c = eaTest.Attendee__c);

         Test.startTest();
         Database.SaveResult result = Database.insert(ea,false);
         Test.stopTest();

         System.assert(!result.isSuccess());
    }

//Método de teste Bulk
    @isTest
    static void notCreateDuplicateEvetAttBulk(){

        List<Event_Attendee__c> eaBulkTest = [SELECT Event__c, Attendee__c FROM Event_Attendee__c ];
        //system.debug(eaBulkTest.size());

        List<Event_Attendee__c> eaBulkToInsert = new List<Event_Attendee__c>();

        for (Event_Attendee__c eaBulk : eaBulkTest){
            Event_attendee__c evat = new Event_attendee__c(
            Event__c = eaBulk.Event__c,
            Attendee__c = eaBulk.Attendee__c);
            eaBulkToInsert.add(evat);
        }
```

```
        Test.startTest();
        List<Database.SaveResult>         results         =
Database.insert(eaBulkToInsert, false);
        Test.stopTest();


        for (Database.SaveResult res: results ){
            System.Assert(!res.isSuccess());
        }
    }
}
```