IPG Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

RELATÓRIO DE ESTÁGIO

Curso Técnico Superior Profissional
em Desenvolvimento de Aplicações Informáticas

Gabriel Carrilho Pinto

setembro | 2019

Curso Técnico Superior Profissional

Desenvolvimento De Aplicações Informáticas

Gabriel Carrilho Pinto

Setembro 2019

Ficha de identificação

Nome: Gabriel Carrilho Pinto

Nº de aluno: 1700512

Curso: Técnico Superior Profissional em Desenvolvimento de Aplicações Informáticas

E-mail: besarfy2@gmail.com

Ano Letivo: 2018/2019

Escola: Escola Superior de Tecnologia e Gestão - Instituto Politécnico da Guarda

Morada: Av. Dr Dr. Francisco Sá Carneiro, 50 - 6300-559

Telefone: +351 271 222 100

Fax: +351 271 222 690

E-mail: ipg@ipg.pt

coordenador: Professor José Alberto Quitério Figueiredo

Empresa: Loba

Morada: Av. Rainha D. Amélia, 142 Cave Guarda – 6300-749

Telefone: +351 271 224 509

E-mail: geral@loba.pt

Coordenador: Micael Costa

Supervisor: Samuel Alves

estágio realizado entre: 25 de fevereiro a 25 de julho de 2017

Duração: 750 horas

Agradecimentos

Antes de iniciar este relatório, tenho que agradecer a todos aqueles que me acompanharam e puxaram por mim durante todo este percurso. Em primeiro lugar devo agradecer aos meus orientadores na empresa Micael Costa e Samuel Alves por acompanharem e orientarem o processo de desenvolvimento em todo o percurso deste estágio.

Devo também agradecer há Filipa Ferreira, Carlos Pinto, Carlos Chora e de novo Samuel Alves que muitas das vezes foram os meus companheiros nas horas de almoço e que me ajudaram a sentir melhor e aliviar o stress. Também tenho que agradecer ao João Costa por me ter ajudado na adaptação à plataforma e linguagem usada na mesma.

Um especial agradecimento há Filipa Costa que esteve presente ao longo de todo o estágio trazendo sempre a sua boa disposição e risos, não esquecendo nunca o trabalho que foi desenvolvido, especialmente por lembrar a todos os estagiários o relatório.

Aproveito também para agradecer aos meus colegas de projeto, que para além do trabalho realizado não ser possível sem ajuda deles, também proporcionaram momentos de alegria. Em especial há Marisa Santos que me apoiou nos momentos mais complicados e stressantes.

E por último mas não menos importante agradeço à minha família por me ajudar a manter o foco e equilíbrio quando necessário.

Resumo

Durante o período de estágio na Loba houve uma grande evolução no aspecto pessoal e sobretudo profissional ao realizar este projecto num ambiente de trabalho real.

O maior objetivo de conhecer e aprender a usar Salesforce foi concretizado com sucesso mesmo este utilizando uma linguagem por mim desconhecida (apex), tal como o aprofundamento de conhecimentos acerca de android studio e linguagem nele usada(java)

Ao longo deste estágio foi desenvolvido um projeto juntamente com mais três estagiários. Este projeto tinha o propósito de desenvolver uma plataforma(web site e app android) para realizar divulgação de eventos do qual estive encarregue de desenvolver toda a aplicação android.

Apesar de aplicação não estar perfeita e ainda existirem bastantes bugs por resolver (que pretendo procurar uma solução) o seu objetivo principal foi realizado.

Palavras-chave: APP, API, Salesforce

Índice

Lista de siglas e acrónimos **6**

1. INTRODUÇÃO **7**

1.1. Contextualização 7

1.2. Objetivos 7

1.3. Caracterização da entidade acolhedora 8

2. Estado da Arte **8**

2.1 Mercado 9

2.2 Ofertas existentes 9

2.2.1. BlueTicket 10

2.2.2. TicketLine 11

2.2.3. EventBrite 11

2.2.4. TMG 11

2.2.5. ViralAgenda 12

2.2.6. Algarve Eventos 12

2.2.7. Eventos do Facebook 12

2.3. Análise 13

3. Tecnologias utilizadas **14**

3.1 SalesForce 14

3.1.1 Funcionamento 14

3.2.1 Exploração 15

3.2.2 Trailhead 15

3.3. Android Studio 19

3.3.1. Funcionalidades Usadas 19

4. Trabalho desenvolvido **21**

4.1 Pré-Projecto 21

4.1.1 Procura de Bugs 21

4.2. Designação do projecto **24**

4.2.1 Introdução 24

4.2.2 App android 24

4.2.3. Website 24

4.2.4. Segurança 24
4.3 Planeamento do Desenvolvimento 25
4.3.1. Escolha de Nome e elaboração de um logotipo 25
4.3.2. Definição das Funcionalidades a implementar 26
4.3.3. Estruturação de Base De Dados 27
4.3.4. Casos de Uso 29
4.3.4. Dicionário de dados 29
4.4. Desenvolvimento 31
4.4.1 Rest API 31
4.4.2. APP android 33
4.4.2.1. Design da interface 33
4.4.2.2. Visualização de eventos 36
4.4.2.3. Compra de bilhete 43
4.4.3. Seguir Evento 44
4.4.3.1. Eventos Seguidos 44
4.4.3.2. Histórico de eventos 45
4.4.3.3. Pesquisa de eventos 45
4.4.3.4.Segurança 46
4.4.3.5. REST API e org 47
5. CONCLUSÃO 48
6. REFERÊNCIAS 49

Lista de siglas e acrónimos

CRM - Customer Relationship management

GPS - global positioning system

API - application programing interface

GET - Método da API utilizado para receber informação

POST - Método da API utilizada para gravar informação na base de dados

APP - aplicação (móbile)

Bug - Erro ou falha no software

IDE - ambiente de desenvolvimento integrado (Integrated Development Environment)

Org - organization (ambiente de trabalho da plataforma Salesforce)

1. INTRODUÇÃO

1.1. Contextualização

Este Relatório tem como objetivo relatar toda atividade realizada durante o tempo de estágio do aluno, tendo início no dia 25 de fevereiro de 2019 e finalizado no dia 25 de julho de 2019.

Este estágio foi realizado num âmbito curricular com a finalidade de completar o curso profissional de desenvolvimento de aplicações informáticas.

1.2. Objetivos

Ao longo deste relatório será descrita toda a atividade realizada durante o período de estágio, sendo assim debatidas algumas noções acerca da plataforma Salesforce, relatar o desenvolvimento de uma aplicação android e a sua interação com o Salesforce.

Pretende-se também debater um pouco sobre Bug catching e as técnicas utilizadas pelo aluno para captar bugs no design gráfico de uma aplicação móvel e bugs funcionais da mesma.

Para além do referenciado acima neste período pré-profissional também se pretende que o aluno desenvolva as capacidades fundamentais no mundo do trabalho como por exemplo:

- o cumprimento dos prazos estipulados,
- éticas de trabalho.

Para tal o aluno teve a responsabilidade desenvolver uma app android cujo objetivo é divulgar eventos consumindo uma API para receber e gravar dados.

1.3. Caracterização da entidade acolhedora

A LOBA - Customer Experience Design, é a primeira agência da experiência de cliente em Portugal; a experiência de Cliente é a qualidade do conjunto de todos os contactos que o cliente estabelece com a sua marca, ou seja a LOBA foca-se principalmente na experiência que os utilizadores dos seus clientes irão receber.

É uma empresa portuguesa na área da Comunicação e Marketing desenvolvendo aplicações móveis e websites para seus clientes utilizando Salesforce como base. São inovadores e exigentes no desenvolvimento, procurando as melhores ferramentas e metodologias que assegurem a qualidade. Formam boas parcerias com os seus clientes, tendo a satisfação destes prioridade da empresa, bem como o sucesso através dos programas desenvolvidos e nelas aplicados.

No decorrer dos anos trabalhou com grandes parceiros mundiais, como BMW, Thales, BULL, IBM, ATOS, Universidade de Salford. Foi nomeada no ano 2019, como Agência Portuguesa do Ano e Agência Lusófona do Ano na categoria de Digital.

2. Estado da Arte

Antes da realização deste projeto foi efetuada uma análise de aplicações web existentes e suas respetivas app's mobile, quando existentes, a fim de comparar e identificar diferenças e semelhanças nas funcionalidades providas por cada uma delas até à data de finalização deste estudo.

2.1 Mercado

Das plataformas exploradas mesmo que por vezes limitadas, todas contêm algum tipo de filtro e criação de eventos. Para além do seu objetivo principal, a pesquisa de eventos, todas permitem a partilha do mesmo nas redes sociais.

A maioria peca por não fornecer uma maneira de realizar a compra de bilhetes e nos filtros de pesquisa que por vezes não são muito avançadas, o que leva a muitos dos utilizadores a preferirem utilizar as plataformas mais completas como BlueTicket e TicketLine

2.2 Ofertas existentes

No mercado já existem algumas aplicações bastante completas para divulgação de eventos e compra dos respetivos bilhetes, no entanto através deste estudo (imagem abaixo) podemos concluir que ainda não existe nenhuma plataforma que esteja completa faltando sempre alguma funcionalidade que poderia vir a ser benéfica para o utilizador, como por exemplo a filtragem de eventos pelo preço.

	criaçao de evento	filtro de pesquisa	Seguir evento	compra de bilhetes	temporizador de compra	quantidade de bilhetes
eventbrite	X	X	X	X	X	X
blueticket	X	X		X	X	X
ticket line	X	X		X		X
viral agenda	X	X	X			
algarve eventos		X	X			
facebook	X	X	X			

2.2.1. BlueTicket

Esta plataforma disponibiliza tanto na app móvel como no website (a criação de eventos embora que esta seja permita apenas a utilizadores que se tornem promotores (funcionalidade paga da plataforma)) a criação de um perfil de utilizador para que este possa comprar bilhetes, receber notificações dos eventos aos quais conseguiu bilhete, notificações baseadas no historial de bilhetes que adquiriu.

É também disponibilizada a escolha do lugar, quando este existe, juntamente com um limite do número de ingressos no ato da compra. No momento de compra fica reservado o bilhete para o evento em questão para que o utilizador tenha tempo de preencher os dados necessários para aquisição do mesmo. Para evitar que este fique preso indefinidamente é ativado um temporizador e quando chegado ao fim põe termo à reserva feita no início da transação.

Permite uma filtragem bastante completa podendo esta ser feita por:

- nome
- dia
- mês
- localização
- categoria
- relevância

2.2.2. TicketLine

Com uma configuração exclusivamente para website é em tudo semelhante à BlueTicket, no entanto, sem a possibilidade de criar um perfil de utilizador, esta aplicação não disponibiliza nenhum tipo de personalização, ou seja, não lhe é possível oferecer ao seu utilizador recomendações com base nas suas aquisições. Ainda que sem esta funcionalidade (criação de perfil) a compra é na mesma disponibilizada.

2.2.3. EventBrite

Tanto na sua aplicação móvel como web as funcionalidades que fornece são bastantes idênticas à BlueTicket apesar de não existirem recomendações personalizadas, a grande diferença reside no facto de ser fornecida ao utilizador uma experiência mais personalizada, pois esta plataforma traz uma funcionalidade nova; a possibilidade de seguir eventos dando ao utilizador uma maior facilidade em encontrar os eventos onde haja interesse.

Na pesquisa, diferem um pouco, uma vez que nesta é feita por nome, preço, dia, localização, categoria, relevância e data.

2.2.4. TMG

Apenas disponível na web é uma bilheteira destinada a promover apenas eventos que sejam realizados no TMG (Teatro Municipal da Guarda) ou no café concerto (local agregado ao TMG) como tal não existe a necessidade, nem possibilidade de criar um perfil de utilizador.

A pesquisa é feita por dia, nome ou categoria. Para efetuar a compra de bilhetes previamente é requerido que se escolha o lugar, caso exista lugar marcado, conjuntamente com a sua limitação, e da mesma forma que nas app’s anteriormente referidas é feita uma reserva e ativo um temporizador para a concluir a compra.

2.2.5. ViralAgenda

A viralAgenda encontra-se disponibilizada tanto em website como em app android. Também esta contém a possibilidade de perfil de utilizador que pode seguir e criar os seus próprios eventos para além disso é oferecida a partilha de eventos através das redes sociais. Os filtros de pesquisa são simples sendo eles por dia, mês, nome, local e categoria.

2.2.6. Algarve Eventos

Algarve Eventos é semelhante ao TMG no sentido em que esta apenas mostra eventos que estejam localizados na zona geográfica de Portugal que corresponde ao algarve.

Disponível em aplicação móvel e web site é possível a partilha dos eventos pelas redes sociais, inclui filtros básicos de pesquisa no entanto é bastante limitada em termos de eventos uma vez que não foi encontrada nenhuma maneira para criar eventos.

2.2.7. Eventos do Facebook

Esta rede social que todos bem conhecemos permite a qualquer utilizador criar eventos com a particularidade de poderem ser privados. Embora não seja possível a compra de ingressos para os mesmos utilizadores usufruírem do poder de seguir os eventos e partilhar na própria rede ou em outras redes sociais, ainda conta com a capacidade de convidar amigos.

Nela, os filtros de pesquisa também são bastante simples, estes são efetuados por dia, mês, nome, local, categoria e por data.

2.3. Análise

E todas as plataformas estudadas não foi encontrada uma filtragem por hora, sendo que, um utilizador que tenha uma agenda muito ocupada poderia beneficiar dessa funcionalidade.

Outra funcionalidade que apenas 2 plataformas têm implementadas é a uma filtragem por relevância de evento, ou seja, filtrar os eventos mais populares de uma certa categoria, que para um utilizador que queira saber o que anda nas tendências apenas de uma categoria em que esteja interessado poderia vir a ser útil.

A maioria das plataformas não continham uma forma de ver eventos semelhantes aos que o utilizador seguiu ou frequentou; esta é uma funcionalidade que beneficiaria um utilizador ao facilitar a procura de novos eventos semelhantes aos gostos do utilizador.

A maioria das plataformas não contém uma forma de seguir organizadores de eventos, o que poderia ajudar o utilizador a receber notificações de eventos que se costumam repetir (como workshops ou cursos).

Uma das funcionalidades que apenas duas plataformas contêm é a de fazer comentários a um evento. Esta ajudaria tanto os organizadores a saber o que está de bom e mau com o evento através do feedback dos frequentadores no pós evento, assim como os utilizadores que estejam interessados em participar no evento podem ter a sua decisão facilitada pelo mesmo feedback de pré-evento ou pós evento caso este venha a repetir-se.

Uma das plataformas(Ticktline) que possibilitam a compra de bilhetes contêm uma falha que pode tornar-se um grande inconveniente para o utilizador apesar de existir um temporizador na compra do bilhete, ou seja, durante o tempo de compra o bilhete é reservado durante um certo período de tempo (10 a 15 minutos) ou até que o utilizador conclua a compra do bilhete, no entanto este temporizador não é mostrado de forma nenhuma e o utilizador não é notificado do que já não lhe é possível finalizar a compra obrigando o comprador a refazer toda a sua compra.

3. Tecnologias utilizadas

3.1 SalesForce

Salesforce é uma plataforma que visa ajudar os seus clientes a entender melhor quais as necessidades do mercado, uma vez que facilita a gerência e controlo de negócios de pequenas ou grandes dimensões. Está concebida para ajudar na venda, prestação de serviços, anúncios, análise e interligação entre clientes.

3.1.1 Funcionamento

A plataforma funciona como um backoffice, sendo o objetivo principal arquivamento e tratamento de dados

A plataforma guarda os dados em segurança numa cloud organizando-os e os disponibilizando na sua interface de simples uso e entendimento para que o cliente tenha rápido acesso e possa:

- gerir seus contactos
- trabalhar com os seu cliente em potencial
- organizar tarefas
- focar-se nos negócios certos
- colaborar com a equipa
- fechar mais negócios

Salesforce possui vários tipos de cloud cada uma com o seu objetivo diferente

Sales Cloud

É o CRM de vendas da Salesforce. Esta solução possibilita:

- O Acompanhamento de todas as informações de vendas e interações ao cliente.

- A Automatização de tarefas - integra e apoia a gestão das áreas de vendas, marketing e atendimento ao cliente.

Service Cloud

- Solução que permite o atendimento ao cliente vinte e quatro horas por dia, sete dias por semana completamente personalizadas.

Marketing Cloud

- Plataforma dedicada ao marketing personalizado, fácil gestão de anúncios.

Commerce Cloud

- Dedicada ao gerenciamento da comunidade e na conexão entre clientes, parceiros e funcionários.

Community Cloud

- Dedicada à criação de uma comunidade de clientes, parceiros e funcionários.

IoT Cloud

- Permite a conexão entre todos os dispositivos, aplicações móveis e sites.

Org

- É o local onde toda a informação é guardada e gerida, contém uma interface simples e fácil de se acostumar.

3.2.1 Exploração

Inicialmente e com o objetivo de facilitar a familiarização da plataforma Salesforce foi utilizada a ferramenta trailhead criada pela mesma. Alguma da exploração exercida foi através de tentativa-erro recorrendo sempre a trailheads específicos a quando não encontrada uma solução.

3.2.2 Trailhead

Trailhead é uma **plataforma** criada pelo Salesforce concebida para ajudar novos interessados a conhecer e a aprender a usar Salesforce. Para que os utilizadores possam aprender nas melhores condições, esta plataforma presenteia os seus

utilizadores com uma simulação de trabalho para estes praticarem funcionalidades e tutoriais.

Esta **ferramenta** está organizada por trilhas, módulos e unidades. Cada trilha é formada por diversos módulos ou projetos, estes módulos são constituídos por diversas unidades de aprendizagem. Para além das trilhas criadas pela própria plataforma existem também as trailmixes; estas podem ser mais abrangentes que as trilhas normais pois podem ser criadas por qualquer utilizador da comunidade.

Para concluir as unidades existem três formas diferentes, para unidades onde as temáticas não requerem prática, ditas mais teóricas, a conclusão costuma ser através de um questionário no qual todas as perguntas têm que ser respondidas de forma correta (em caso de falha é possível repetir o questionário, representado na **imagem 1**)

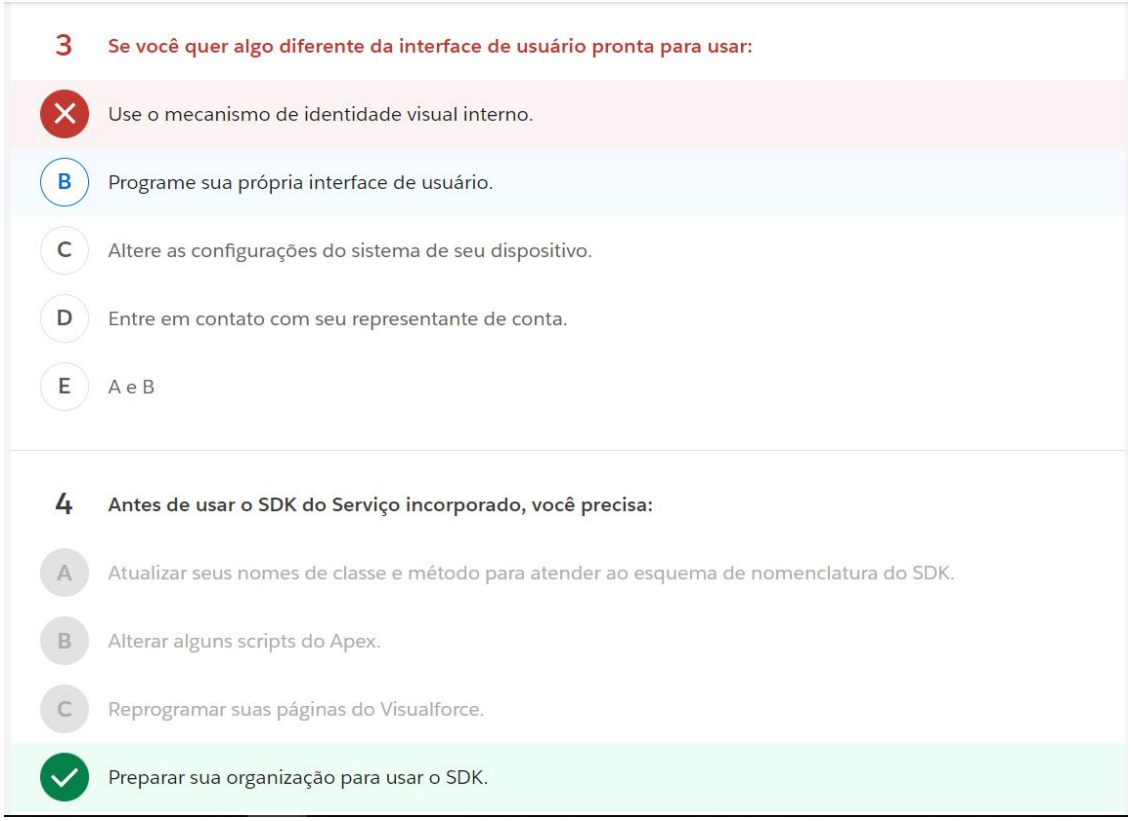

imagem 1 - Repetição de questionário a quando falhada uma pergunta.

fonte: https://trailhead.salesforce.com

em unidades mais práticas é criada uma simulação de trabalho onde o utilizador terá de resolver um desafio e através de uma verificação do mesmo, é concluída a unidade caso a resolução esteja correta com demonstra a **imagem 2** ou sem qualquer tipo de verificação, representado na **imagem 3**, este apenas costuma aparecer em unidades cuja temática tratada nelas se torna impossível para a ferramenta aceder e fazer qualquer tipo de verificação, como por exemplo a instalação de um programa externo no computador do utilizador.

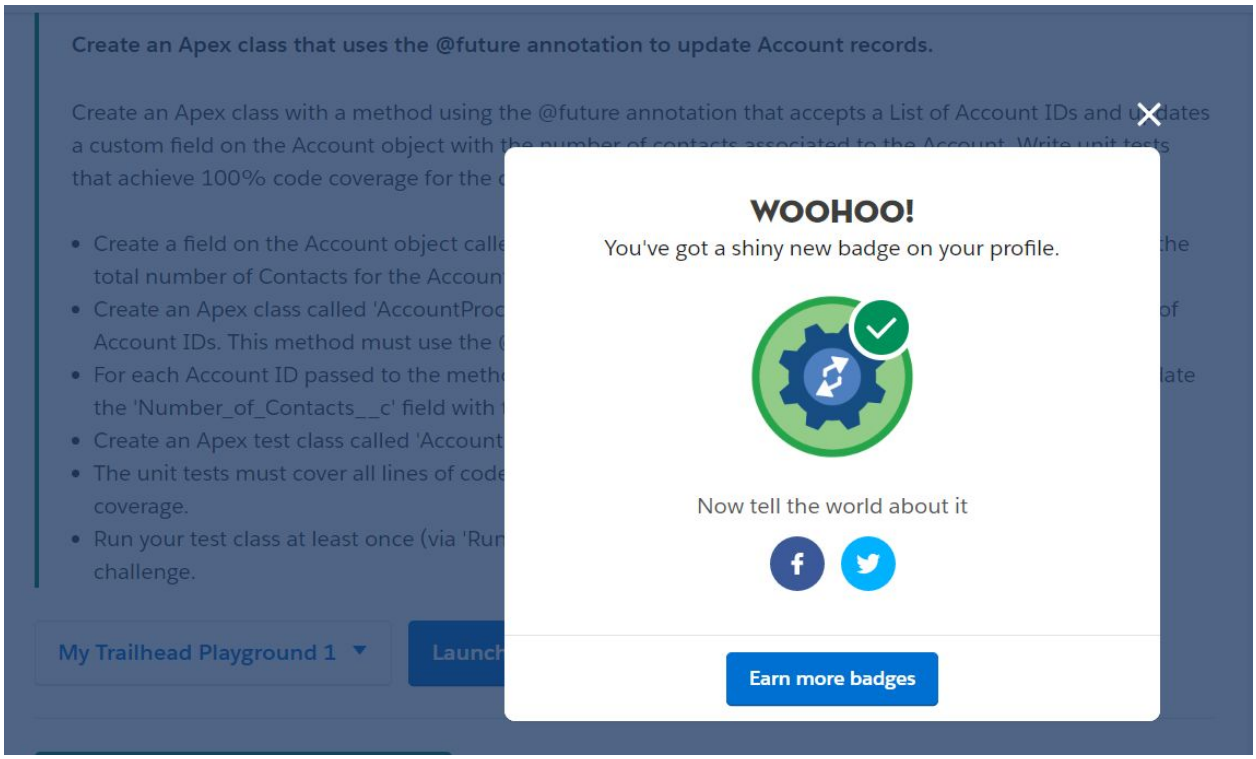

imagem 2 - Unidade concluída depois de realizar a parte prática com sucesso.

fonte: https://trailhead.salesforce.com

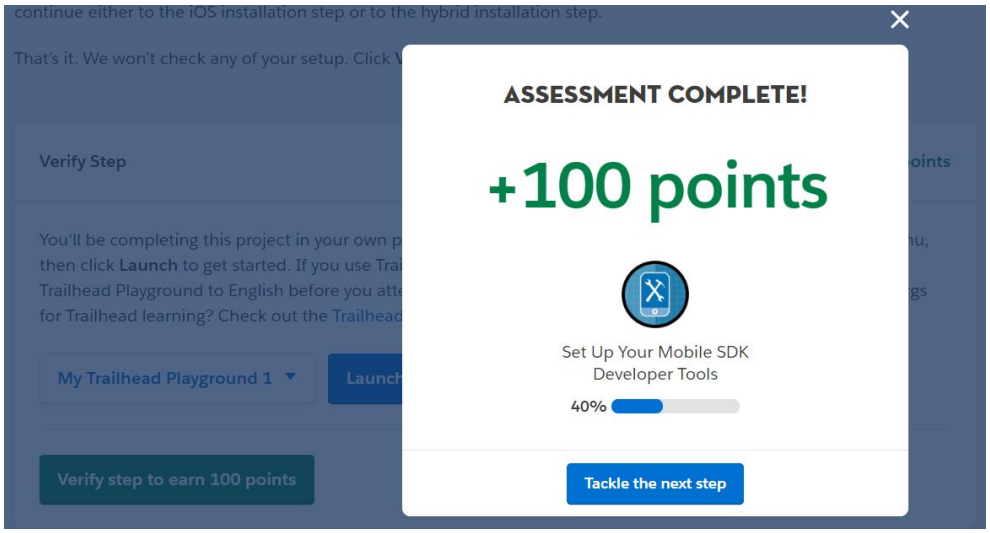

imagem 3 - Unidade concluída sem qualquer tipo de verificação.

fonte: https://trailhead.salesforce.com

Como forma a manter os utilizadores incentivados, ao completar qualquer módulo, projeto, ou trail o utilizador é premiado com uma medalha como é possível ver na **imagem 4**

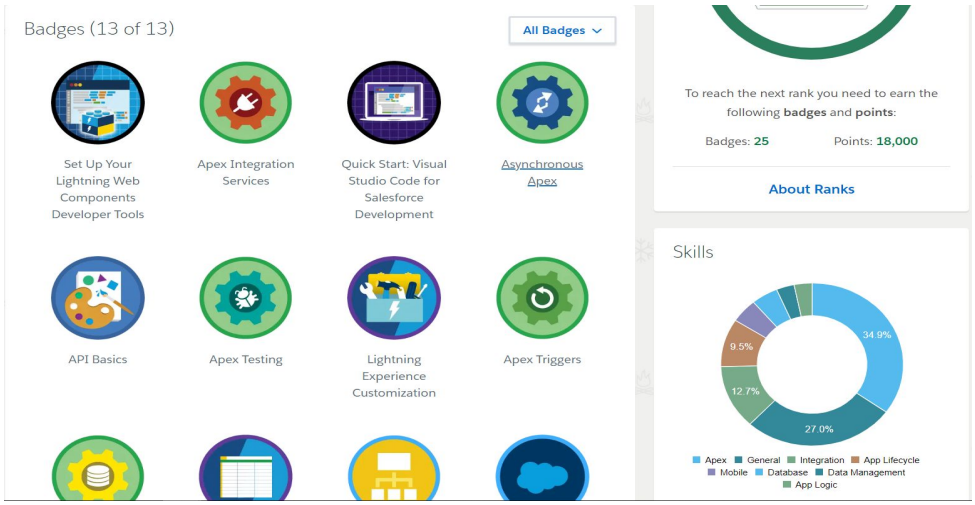

imagem 4 - Medalhas adquiridas pelo utilizador

fonte: https://trailhead.salesforce.com

3.3. Android Studio

Android Studio é um IDE de desenvolvimento inteiramente dedicado ao sistema operativo android , ou seja com este software apenas podemos desenvolver aplicações android. O desenvolvimento neste ambiente é efetuado através de classes usufruindo da linguagem java como base.

3.3.1. Funcionalidades Usadas

Quando são criadas atividades, o software cria uma classe referente à mesma que controla inicialmente duas componentes:

- **activity layout**, estes layouts são gerais e serão utilizados em todas as view referentes a essa atividade.
- **content layouts**, já estes são view que geralmente é trabalhada, pois é esta que o utilizador final irá ver.

Nos layouts construídos foram usados as seguintes componentes visuais:

- **TextView** é uma componente que serve apenas para Ler/mostrar texto.
- **EditTextView** esta componente é semelhante à anterior, no entanto apesar de se poder mostrar texto nela, o seu objetivo principal é funcionar como uma caixa de texto onde o utilizador introduz valores
- **Button** componente é um botão clicável.
- **ImageView** componente cujo objetivo é a introdução de imagens
- **ListView** este item é uma lista que consome outra view pois cada elemento da lista irá tomar a forma da view que estiver a ser consumida
- **ScrollView** este item permite fazer um scroll, normalmente é usado em layouts compridos, ou seja com muitas componentes que necessitem de espaço visual.
- **Radiogroup** Este item serve para controlar radiobuttons
- **RadioButton** Este item é um botão de seleção. Normalmente são adicionados a um radiogroup para que este controle a escolha e não permita múltiplas escolhas embora isso seja possível.
- **Switch** é um interruptor apenas nos dá a possibilidade de ter dois valores ou está ativo ou desativo.
- **Spinner** é uma caixa de seleção.

- **calendarView** é uma componente cujo seu objetivo é mostrar um calendário podendo ser-lhe mudada a data.
- **SeekBar** esta componente é uma barra cujo utilizador vai arrastando consoante a percentagem da barra que necessita para o objetivo em que estiver a ser utilizada.
- **Divider** é apenas uma linha reta normalmente usada para fazer separações.

4. Trabalho desenvolvido

4.1 Adaptação à empresa

Antecedente ao projeto houve um período de tempo de adaptação e aprendizagem no qual o aluno se dedicou a conhecer a plataforma e as normas de trabalho da empresa.

4.1.1 Procura de Bugs

No contexto de uma situação no mundo real a procura de Bugs em aplicações é sempre um passo essencial no desenvolvimento de um projeto para evitar que o cliente sofra danos colaterais, então foi-me dado acesso a duas aplicações no fim do seu desenvolvimento com a finalidade de procurar e reportar qualquer bug existente.

Ambas as aplicações continham bugs embora uma delas (simulador MKT-azeites Montanha) tivesse bugs visuais facilmente corrigíveis. Nesta, foram realizados maioritariamente testes funcionais. A **imagem 5** demonstra uma tabela onde colocados os resultados dos testes de validação

Validar output de **Espanha e Itália**	07/03/2019	Gabriel Pinto	OK	
Validar output de **França**	08/03/2019	Gabriel Pinto	OK	
Validar output de **Reino Unido**	09/03/2019	Gabriel Pinto	OK	
Validar output de **Alemanha**	10/03/2019	Gabriel Pinto	OK	
Validar output de **Brasil**	10/03/2019	Gabriel Pinto	OK	
Validar output de **EUA**	10/03/2019	Gabriel Pinto	OK	
Validar output de **Japão**	11/03/2019	Gabriel Pinto	OK	
Validar output de **China**	12/03/2019	Gabriel Pinto	OK	
Validar output de **Portugal**	10/03/2019	Gabriel Pinto	OK	É necessario definir melhor as respostas corretas na aba "Mercados"

Imagem 5 - tabela de resultados depois de realizado o teste de validação de output

Na segunda aplicação (Expamscience - Academy) foram encontrados vários bugs, alguns de baixo nível de preocupação como uma simples multiplicação de resultados (não prejudiciais) quando se realizava uma tarefa, como detalhado na **imagem 6** retirada do relatório de reporte.

#ID	6	**Data:**	28/02/2019
Estado:	não resolvido	**Tester:**	Gabriel Pinto
Ambiente:	Mobile	**SO/Browser:**	Android
Zona:	curso	**Versão:**	V0.0.8
Descrição:	Multiplicação de utilizadores na zona de pontuação		
Passos do teste:	Ao clicar no botão info os utilizadores que estão no top de pontuação daquele curso são duplicados		
Observações:	Apenas acontece uma vez. Ao sair e voltar a entrar no curso o duplicados sao removidos		

Imagem 6 - tabela de identificação do bug e sua prova de existência.

No entanto, existiam bugs que seriam potencialmente prejudiciais ao cliente uma vez que se tratavam de troca de produtos comerciais por pontos ganhos na aplicação, pontos que na altura da análise não se perdiam quando gastos, como descrito na **imagem 7** retirada do relatório de reporte

#ID	10	**Data:**	28/02/2019
Estado:	não resolvido	**Tester:**	Gabriel Pinto
Ambiente:	Mobile	**SO/Browser:**	Android
Zona:	créditos	**Versão:**	V0.0.8
Descrição:	Recuperação de todos os créditos gastos quando se ganha créditos		
Passos do teste:	Gastar creditos, fazer uma actividade.		
Observações:			

Imagem 7 - tabela de identificação do bug encontrado.

4.2. Designação do projecto

4.2.1 Introdução

O projeto teve o seu início com a entrada de três novos estagiários no dia 20 de março formando uma equipa de quatro elementos para desenvolver o mesmo. Apesar de este ser em comum a maioria do trabalho foi realizado individualmente uma vez que coube a cada elemento desenvolver uma parte distinta.

Utilizando o Salesforce como a base do projeto, este consiste em criar um website e uma app móvel para fazer divulgação de eventos juntamente com as políticas de privacidade e segurança necessárias para ambas as plataformas.

A app móvel ficou toda ela sobre a minha responsabilidade de a desenvolver.

4.2.2 App android

A app android irá principalmente receber dados de uma Rest API criada na plataforma SalesForce. Esta API utilizará um método Get que reunirá os dados de cada evento como o nome, descrição e data do mesmo, para que sejam expostos e utilizados de forma simples e dinâmica. Esta app irá também usar o método post da Rest API como forma de enviar dados à plataforma Salesforce a fim de os registar e recuperar sempre que necessário.

4.2.3. Website

O website foi inteiramente elaborado pelo colega de trabalho Mário Simões. Este website continha o mesmo objetivo da aplicação android.

4.2.4. Segurança

Em termos de segurança tanto o website como a app android foram analisadas múltiplas vezes para encontrar falhas onde hackers pudessem interferir, e para manter os utilizadores informados da forma como são tratados os dados introduzidos por eles foi elaborada uma política de privacidade, pela colega de trabalho Marisa Santos.

4.3 Planeamento do Desenvolvimento

4.3.1. Escolha de Nome e elaboração de um logotipo

Como qualquer aplicação foi debatido em conjunto e escolhido um nome para identificar a plataforma desenvolvida ao longo deste projeto. Inicialmente o nome fora RPortalEventZ mas após a confirmação do fundo vermelho na app mobile e o início da elaboração do logotipo fez-se a redução para o nome final PortalEventoZ.

Foi concebido um teste de visualização na app android o logótipo a preto e branco e sua inversão de cores e a cores, sendo que o logótipo a preto e branco, como se pode ver na **imagem 8** foi o mais votado para permanecer como final.

O Design deste logotipo foi trabalho inteiramente realizado pela Estagiária Marisa Santos com aprovação da restante equipa encarregue deste projeto.

imagem 8 - Logótipo Final preto e branco

fonte : Elaboração Marisa Santos

4.3.2. Definição das Funcionalidades a implementar

Após o estado de arte juntamente com os colegas de projeto foram levantadas as funcionalidades mais comuns, tal como as menos utilizadas e as que estão em falta, e após uma análise às funcionalidades, foi decidido quais as que devem ser implementadas na aplicação:

- Mostrar o caminho para o evento através do sistema gps do dispositivo
- fazer uma filtragem por:
    - preço
    - dia
    - nome
    - localização
    - categoria
- O utilizador deve poder seguir eventos e receber notificações sobre os mesmos quando:
    - dia do evento está aproximar-se
    - houver algum update nos dados do evento
    - caso o evento seja cancelado
- Os utilizadores com uma conta devem poder fazer uma denúncia a qualquer evento ou organizador caso esta ação seja justificada.
- A aplicação deve conseguir recolher um QRCode gerado na plataforma SalesForce para que o utilizador o use como entrada no evento.
- A aplicação irá também exibir o histórico de eventos que o utilizador (com login efetuado) frequentar.

Para o desenvolvimento desta aplicação foi construída uma base de dados na plataforma SalesForce e foram usadas algumas APIs para receber e enviar os dados necessários. A Base de dados foi preenchida com dados acerca de eventos verídicos. Esta app foi completamente desenvolvida em Android Studio.

4.3.3. Estruturação de Base De Dados

Primeiro passo antes de realizar a base de dados utilizada durante todo o projeto, foi elaborada um modelo conceitual para melhor compreender o problema e tentar encontrar a melhor solução para o resolver. Este modelo também permite a análise funcional da base de dados para que não houvessem incoerências na mesma e existisse uma identificação correta entre os atributos de cada entidade e a relação entre elas. No processo de desenvolvimento deste modelo foram estudadas as tabelas existentes por pré-definição em cada org da Salesforce(profile,role, Contact, entre outras) tentando assim utilizá-las para a versão final da base de dados utilizada no projeto. Abaixo está o modelo inicial (**imagem 9**)

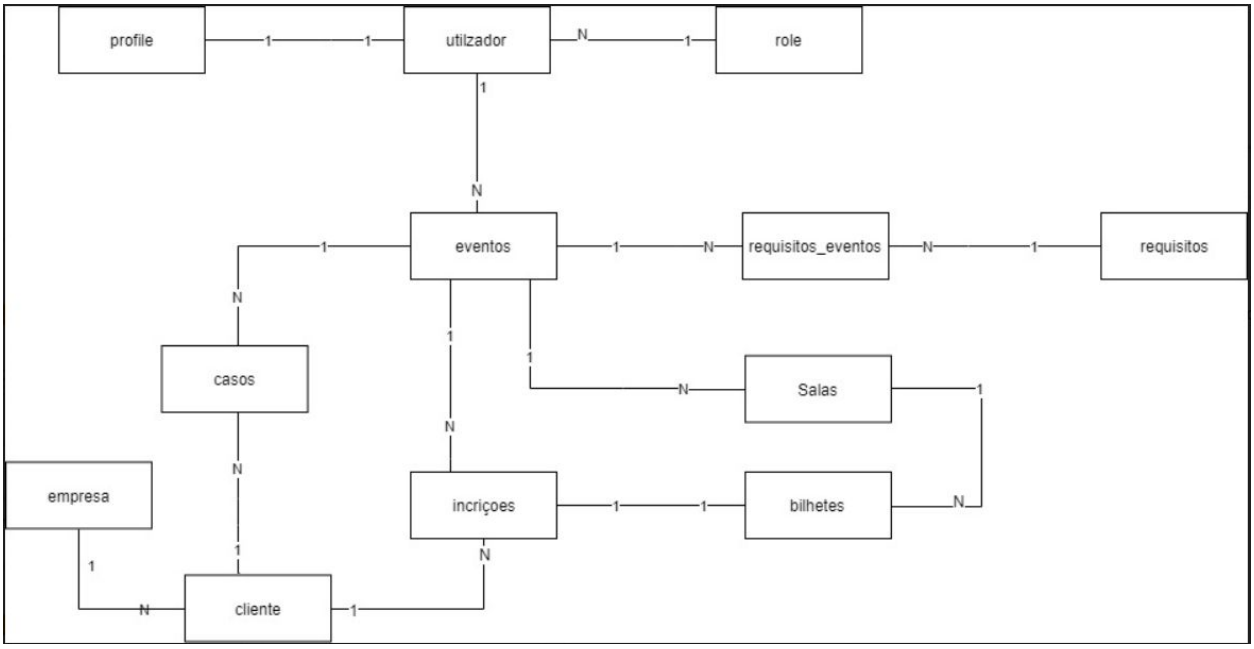

imagem 9 - modelo conceitual inicial.

Fonte: realização do próprio.

Após a materialização do modelo acima referido foram encontradas diversas falhas no seu funcionamento que depois de vários testes e retoques sobre este modelo foi desenhado o modelo ER (modelo entidade relacionamento), representado na **imagem 10**, utilizado durante todo o resto do projeto.

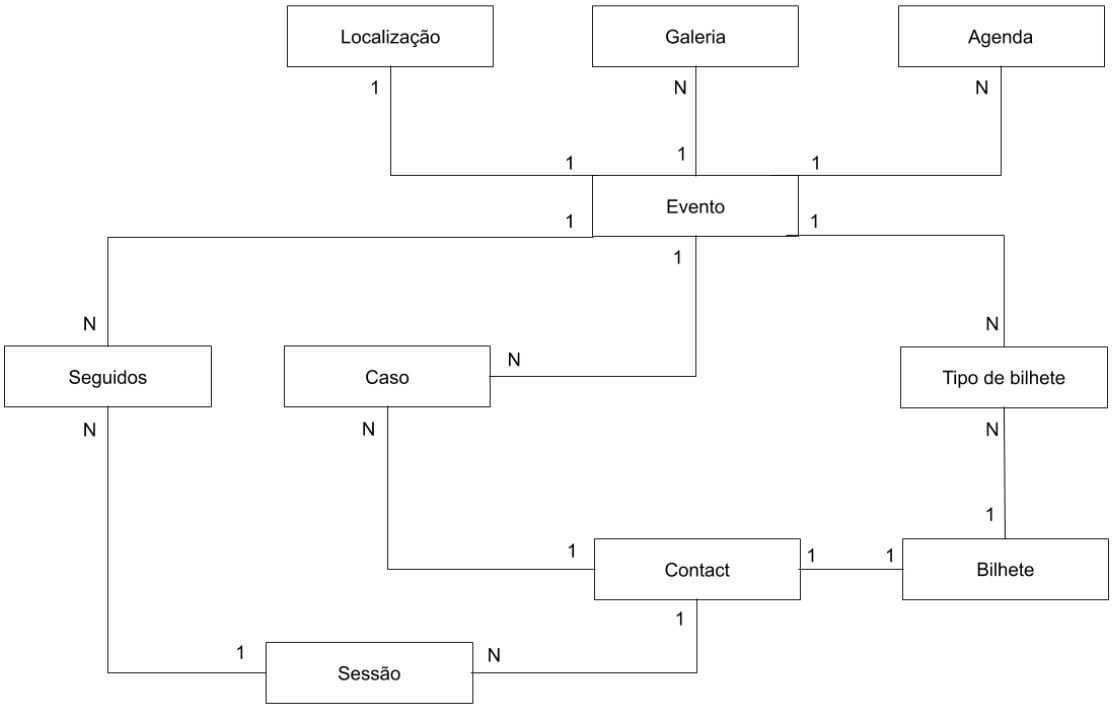

imagem 10 - modelo entidade relacionamento final.

4.3.4. Casos de Uso

Os casos de uso funcionam como uma interligação entre o que o cliente pediu para ser realizado e o programador que irá desenvolver, pois os casos de uso são como um manual de instruções para ambos. O programador através destes consegue identificar que funcionalidade deve implementar e como ela deveria funcionar. É possível ver um destes casos na **imagem 11**.

Nome	Filtrar evento por localização
Descrição	O sistema filtra os eventos ao utilizador de acordo com a localização do país ou cidade selecionada.
Pré-condições	Selecionar a área de pesquisa na aplicação móvel ou página web;
Caminho principal	1. O ator seleciona a localização 2. O sistema devolve todos os eventos que se realizem na localização inserida
Caminhos alternativos	Não existe
Suplementos	Fazer um teste onde se verifique: - Se o sistema devolve todos os eventos realizados na localização inserida

Imagem 11 - tabela demonstrativa de um caso de uso na filtragem de eventos por local.

4.3.4. Dicionário de dados

O dicionário de dados é um documento onde se encontram todas as tabelas utilizadas no projeto e seus atributos juntamente com o tipo de dados a que estes estão reservados. Este documento é útil no:

- **início do projeto -** ao estudá-lo podem se fazer correções na tabela caso os dados da mesma possam vir a dar problemas no futuro ou estejam errados.

- **durante o desenvolvimento -** embora menos utilizado, caso o programador que estiver a desenvolver não saiba/lembre que tipo de dados está a trabalhar este documento fornece-lhe uma maneira fácil e rápida de adquirir essa informação.

4.4. Desenvolvimento

Neste projeto foi atribuído ao aluno trabalhar toda a parte da aplicação android. Apesar de ter sido maioritariamente esse o foco também houve espaço para desenvolver algumas funcionalidades dentro da plataforma, como criar campos e editá-los, criação de triggers, modificação de layouts na org (inserção e visualização de dados) e desenvolvimento de parte da API utilizada.

4.4.1 Rest API

Uma API (Application Programming Interface) é um conjunto de mensagens de requisição e resposta HTTP, geralmente expresso nos formatos XML ou JSON.

No contexto deste projeto a API foi utilizada para fazer a ponte entre a app e o servidor através de dois métodos distintos:

- Método GET no qual a app faz uma chamada HTTP com ou sem parâmetros para no final receber os dados hospedados no servidor em formato JSON.

- Método POST no qual a app faz uma chamada HTTP com pelo menos um parâmetro para que o servidor fizesse uma inserção de dados na base de dados devolvendo no final uma resposta simples em formato string para informar o utilizador da app se os seus dados foram inseridos corretamente ou se ocorreu algum tipo de falha.

Nas **imagens 12 e 13** é possível a visualização do código de um método GET e parte de um método POST respetivamente.

```
@RestResource(urlMapping='/Eventos')
global class WSEventos {

    @httpget
    global static List<Evento__c> getEventos(){
        RestRequest request = RestContext.request;
        // grab the caseId from the end of the URL

        List<Evento__C> result =  [SELECT ID,Name,URLImagem__C,Descricao__c,breve_descricao__c,Data__c,preco__c
                                   FROM Evento__C];
        return result;

    }

}
```

Imagem 12 - Código de um método GET da API elaborada em Salesforce.

```
@HttpPost
global static sessao__c Login(String email, String secret) {
    //lista para todas as sessoes do utilizador
    List<sessao__c> todasSessoes = new List<sessao__c>();

    List<Contact> Contacts = [Select id,email,password__C from contact];
    List<Id> ContactsId = new List<Id>();

    for(Contact w : Contacts){
        ContactsId.add(w.Id);
    }
```

Imagem 13 - parte do código de um método POST da API elabora em Salesforce.

4.4.2. APP android

Embora tenha sido sugerida e estuda a possibilidade de desenvolver uma aplicação híbrida em **cordova** devido ao estado de desenvolvimento a quando desse estudo a aplicação manteve-se Nativa.

Esta aplicação foi inteiramente desenvolvida em **Android Studio** maioritariamente na **Versão 3.4.2**

4.4.2.1. Design da interface

A primeira tarefa efetuada nesta aplicação foi design. Por questões de facilidade em fazer alterações foi inicialmente esboçado em papel. Assim que o esboço foi finalizado e a ideia do design final foi concretizada, uma vez que envolvia a visualização de imagens; fez-se a sua implementação para que fosse possível testá-lo. Parte deste primeiro design implementado é possível ser visto na **imagem 14**

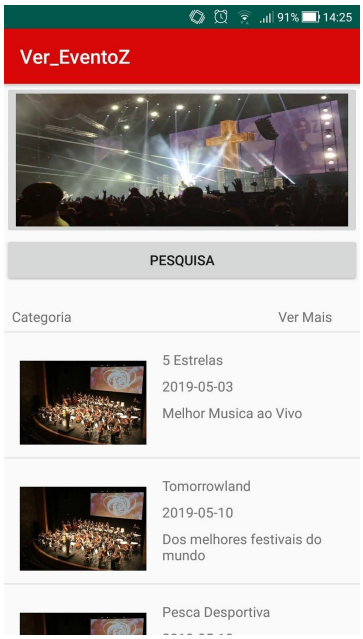

Imagem 14 - desgin da visualização de todos os eventos existentes na aplicação

Fonte: Elaborado pelo próprio

Mesmo existindo a possibilidade de bloquear a rotação de ecrã, essa função não foi utilizada, pelo que foi criado um design para a visão de landscape (ecrã rotacionado). Este design foi inteiramente baseado no já referido acima, como tal o esboço foi realizado direto no android studio e retocado. É possível ver um exemplo deste design na **imagem 15**

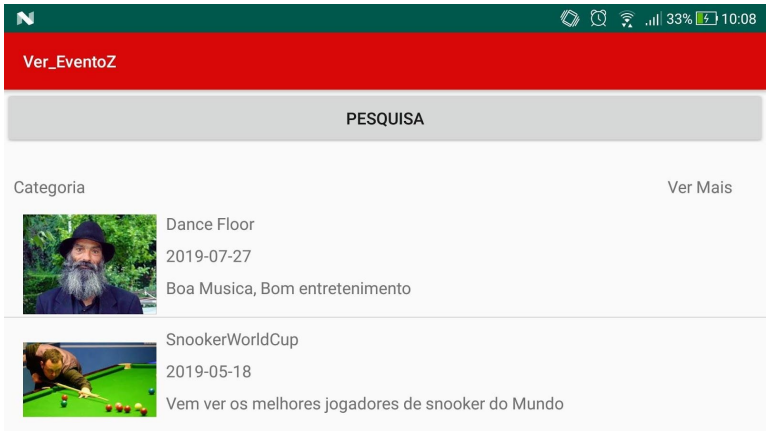

Imagem 15 - Desgin da Visualização de todos os eventos em landscape.

Fonte: Elaborado pelo próprio.

Para que não houvesse uma discrepância entre o website e a aplicação móvel debateu-se com a restante equipa as cores a utilizar tendo sido o vermelho a cor mais votada.

À medida que a aplicação ia ganhando forma e o projeto se ia desenvolvendo surgiu a necessidade de fazer retoques no design adicionando novas views retirando outras, modificando campos de local acabando culminado na versão final visível nas **imagem 16**

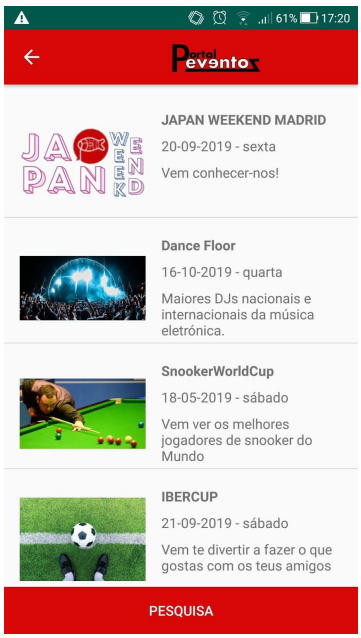

Imagem 16 - Desgin final da aba de visualização de todos os eventos da app android.

fonte: Elaborado pelo próprio.

4.4.2.2. Visualização de eventos

A visualização dos eventos está dividida em duas partes. **A primeira** foi utilizada umas lista para colocar todos os eventos existentes na plataforma onde o utilizador vê apenas nome, imagem e uma breve descrição do evento.

Para preencher esta lista recorreu-se à chamada da API através do código visível na **imagem 17** para receber os dados de todos os eventos no servidor, estes foram colocados separadamente em **StringList** para facilitar a transposição entre classes.

```
private static StringBuilder sb = new StringBuilder();
    private ArrayList<String> EventosNome = new ArrayList<String>();
    private ArrayList<String> EventosDescricao = new ArrayList<String>();
    private ArrayList<String> EventosData = new ArrayList<String>();
    private ArrayList<String> EventosImagen = new ArrayList<String>();
    private ArrayList<String> EventoID= new ArrayList<String>();
    private int PrecoMax=0;
    //make connection with API
    @Override
    protected Void doInBackground(Void... voids) {
        BufferedReader br=null;
        try {
            String myUrl="https://apitest-developer-edition.eu19.force.com/services/apexrest/Eventos";
            URL url = new URL(myUrl);

            br = new BufferedReader(new InputStreamReader(url.openStream()));
            String temp="";

            //get API Data
            while ((temp = br.readLine()) != null){
                sb.append(temp+"\n");
            }
        }catch (IOException e) {
            Log.e("log_tag", "Error in http connection "+e.toString());
        }finally {
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return null;
    }
```

Imagem 17 - código de chamada da api em android studio.

Fonte: Elaborado pelo próprio.

Após essa transposição foi usada uma classe com extensão de uma classe base do android studio (BaseAdapter), esta repetir-se-á tantas vezes quantas o tamanho da lista dos eventos e realizará o preenchimento de dados nos respectivos campos de cada item da lista. a **imagem 18** mostra parte dessa classe.

```
class CostumView extends BaseAdapter {
        @Override
        public Object getItem(int id) {
            return null;
        }
        @Override
        public long getItemId(int id) {
            return 0;
        }
        @Override
        public int getCount() {
            return EventosNome.size();
        }

        @Override
        public View getView(int posicao, View app, ViewGroup viewgroup) {
            if (orientacao == 1) {
                app = getLayoutInflater().inflate(R.layout.ver_eventos_normal_lista, null);
                TextView nomeEvento = (TextView) app.findViewById(R.id.textView_Nome_ver_evento);
                TextView dataEvento = (TextView) app.findViewById(R.id.textView_data_ver_evento);
                TextView descricaoEvento = (TextView) app.findViewById(R.id.textView_descricao_ver_evento);
                ImageView imagenEvento = (ImageView) app.findViewById(R.id.image_eventos_ver_eventos);
                if (ImagenURL.get(posicao) != null) {
                    imageConverter silence = new imageConverter();
                    silence.myImageView(imagenEvento, ImagenURL.get(posicao));
                    silence.execute();
                    try {
                        Thread.sleep(500);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                try {
                    nomeEvento.setText(EventosNome.get(posicao));
```

Imagem 18 - Completar lista com os valores recebidos da API.

Fonte: elaborado pelo próprio.

Os dados referentes às imagens vem em formato url o que se torna num problema a ser resolvido para que uma aplicação android as pudesse visualizar sem abrir uma página à parte com a imagem para isto foi usado uma classe com extensão à classe "AsyncTask" do próprio android studio, esta classe permite transformar a imagem contida no url em bitmap e assim colocá-la numa imageview (tipo de campo usado na visualização) como se pode ver na **imagem 19.**

```
private ImageView whatup;
    private String Url;

    public void myImageView (ImageView as,String url){
        this.whatup = as;
        this.Url = url;
    }

    @Override
    protected Bitmap doInBackground(String... strings) {
        Bitmap convertImage = null;
        try {
            String imagenGetter = Url;
            InputStream in = new java.net.URL(imagenGetter).openStream();
            convertImage = BitmapFactory.decodeStream(in);

        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return convertImage;
    }
```

Imagem 19 - código para converter a imagem em bitmap.

Fonte: Elaboração do próprio.

Também possível ver na **imagem 20,** encontra-se o método usado para transformar o formato de “date” vindo da plataforma SalesForce para um formato “String” para tal utilizou-se uma função interna do Android Studio chamada SimpleDateFormat que nos permite fazer essa transição para que pudesse ser colocada na TextView (tipo de campo usado na visualização) a que lhe é destinada.

```
//2019-09-20
String senddata = ArrayData.get(posicao);

SimpleDateFormat newformat = new SimpleDateFormat( pattern: "dd-MM-yyyy");
String dia = newformat.format(Date.valueOf(senddata));

data.setText(dia);
```

Imagem 20 - transposição do formato de “Date” para formato “String” e uso da mesma

fonte: elaborado pelo próprio.

A segunda parte ocorre quando um evento é selecionado, nesse momento é aberta uma nova atividade e é efetuada uma nova chamada a Rest API que devolverá todos os dados relacionados com o id do evento selecionado estes estão divididos consoante a sua importância separando-se assim em três abas

1ª aba (imagem 21) - Esta aba mostra a informação geral e mais importante do evento como o nome do evento, dia, tipo e preços dos ingressos, caso o utilizador tenha feito login as opções de seguir e compra de bilhete serão disponibilizadas.

Imagem 21 - Informação mais essencial do evento.

fonte: Elaborado pelo próprio

2ª aba (imagem 22) - Esta aba mostra informação mais detalhada do evento como a descrição a sua localização, organizador e coorganizador.

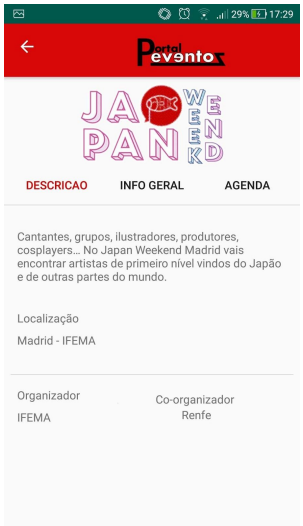

Imagem 22 - Informação mais detalhada do evento

3ª aba (imagem 23) - Esta aba tem como finalidade pormenorizar as atividades do evento, ou seja fazer uma lista de tudo o que irá acontecer no evento. Nesta aba

também existe uma chamada à API para preencher toda a lista que está na agenda do evento.

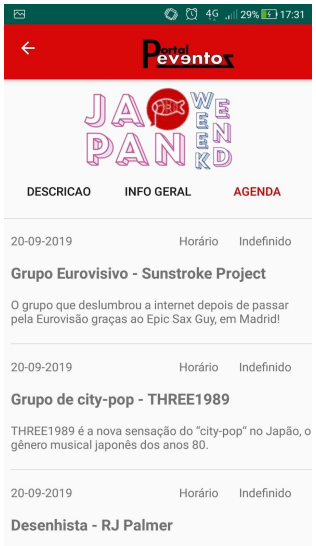

Imagem 23 - Visualização da agenda do Evento.

Apesar de serem diferentes Views, em todas elas está visível a imagem do evento uma vez que esta está guardada na atividade.

4.4.2.3. Compra de bilhete

A compra do bilhete é apenas permitida quando o utilizador tem login feito de outra maneira a aplicação envia uma mensagem de erro e mantém-se na mesma página.

Quando existe login e é acedida à compra a aplicação abre uma nova actividade fazendo uma chamada à API para que esta devolva todos os tipos de bilhetes existentes e os preços respectivos a cada um deles. Para evitar que existam utilizadores a re-vender bilhetes extra foi adicionada um sistema onde cada compra apenas pode ter no máximo um total de dez bilhetes no entanto esta ainda é uma vulnerabilidade possível de atacar uma vez que na plataforma esse limite não chegou a ser implementado.

Para evitar que existam utilizadores a re-vender bilhetes extra, foi adicionado um sistema onde cada compra apenas pode ter no máximo um total de dez bilhetes, no entanto, esta ainda é uma vulnerabilidade possível de atacar uma vez Da mesma maneira é que a compra esta opção de seguir os eventos apenas é para utilizadores que tenham efetuado login de outra maneira esta opção é bloqueada mandando uma mensagem de erro.

Esta opção serve também para deixar de seguir os eventos; Funcionando toda ela com a mesma chamada da API (método POST com dois parâmetros) que verifica se o utilizador está a clicar no botão e se este já está a seguir ao não o evento em questão e realiza um insert ou delete dependendo da situação a que estiver a ser submetida.

O utilizador pode adicionar à lista usada na compra todos os tipos de bilhetes e quantidade que quiser em cada um deles sendo esta aumentada caso volte a adicionar o mesmo tipo de bilhete e assim que adiciona o bilhete a quantidade de ingressos que pode adicionar diminui pela quantidade adicionada. Esta lista é organizada consoante o último bilhete adicionado.

Ao finalizar a compra aparece uma caixa de diálogo ao utilizador nesta a lista dos bilhetes está organizada do bilhete mais caro para o mais barato (como é possível ver na **imagem 24**) se o utilizador concorda em finalizar a compra é chamada a API de novo desta vez é utilizado o método POST que usa quatro parâmetros

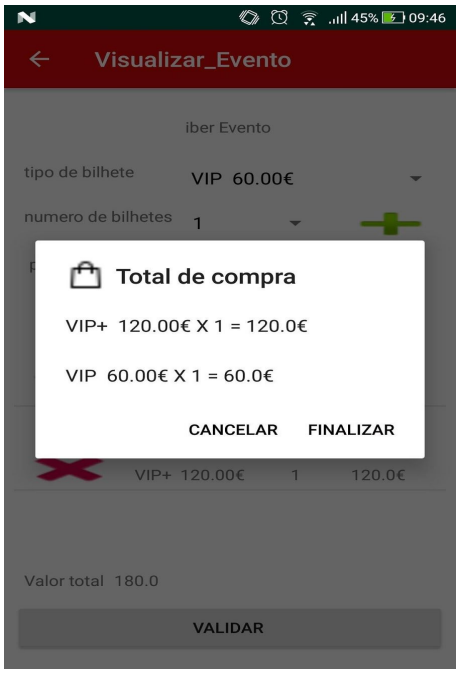

Imagem 24: caixa de diálogo no momento pré-finalização de compra de bilhete.

fonte: elaborado pelo próprio

4.4.3. Seguir Evento

4.4.3.1. Eventos Seguidos

Existe na aplicação uma opção também ela bloqueada caso não exista login efetuado, uma atividade na qual faz o display de todos os eventos já seguidos pelo utilizador apresentando-os numa lista depois de fazer uma chamada da API que nos devolve esse resultado. A mesma é capaz de enviar o utilizador para a atividade que detalha todo o evento.

4.4.3.2. Histórico de eventos

O histórico de eventos é uma atividade em tudo semelhante á dos eventos seguidos uma vez que também faz uma chamada GET da API no entanto esta não mostra os eventos seguidos mas sim todos os eventos aos que o utilizador em questão tiver adquirido um bilhete.

O histórico de eventos está bloqueado para utilizadores que não tenham feito um login válido.

4.4.3.3. Pesquisa de eventos

O método de pesquisa na aplicação funciona através de alguns filtros, o utilizador para além da pesquisa mais básica e essencial, como fazer uma pesquisa pelo nome ou data do evento o utilizador tem a sua disposição a escolha do país e cidade podendo também fazer uma pesquisa por meio do preço, por exemplo pesquisar todos os eventos que variam entre 0€ e 20€. Depois de escolhidos os diversos filtros é de novo consumida a API(método GET) que vai levar todos esses parâmetros sendo que aqueles que o utilizador não usar contêm um valor base.

Para a pesquisa feita pelo nome foi apenas usada uma caixa de texto (EditText) que caso não preenchida pesquisará todos os eventos.

Os filtros feitos por meio de país e cidade foram feitos de igual modo mas uma vez que um depende do outro os valores que aparecem nas cidades variam dependendo do país escolhido, estes valores estão contidos num spinner, ou seja, uma lista de escolha com valores pré definidos.

Já para o filtro por preço é uma barra (seekBar) que automaticamente toma o valor 0€ que se não for mexida irá pesquisar todos os eventos independentemente do preço no entanto e podendo ela variar até ao preço mais caro que existe na plataforma basta o utilizador arrastar a barra até ao valor que pretende para a filtro ser aplicado.

A pesquisa feita por data é executada por um calendário que está interligado com um interruptor (switch) este último serve para controlar o filtro, inicialmente está desligado mas caso seja ligado a pesquisa irá ser realizada pela data que estiver selecionada no

calendário (podendo este ser clicável mesmo com o filtro desligado).

É capaz ver-se uma demonstração da pesquisa na **imagem 25**

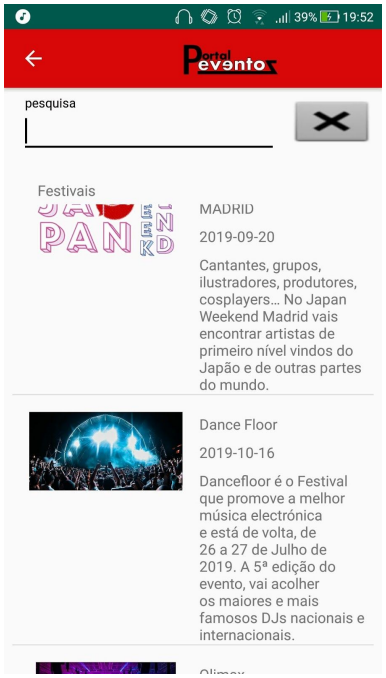

Imagem 25 - pesquisa de todos os festivais

fonte: elaborado pelo próprio

4.4.3.4.Segurança

A aplicação android foi analisada três vezes, esta análise teve o propósito de encontrar as falhas de segurança que poderiam ser exploradas por hackers tendo sido apenas encontradas quatro falhas, duas delas com elevado risco (tentativa de spoofing e permissão de backups) uma de baixo risco (falta de segurança nas configurações de rede) e a quarta de médio risco (código da app não encriptado) no entanto apenas a falha de backups foi corrigida para tal bastou codificar a seguinte linha de código no manifest:

- **android:allowBackup = false**

4.4.3.5. REST API e org

O trabalho realizado na org embora mais curto não menos trabalhoso com o progresso da aplicação foi necessário editar algumas classes da Rest API desenvolvidas por um colega de trabalho e criar outras de raiz.

Os objetivos das seguintes classes criadas de raiz eram os seguintes:

- **schedule_bilhetes** o objetivo desta classe é diariamente chamar a classe **futureBilhetes** enviando-lhe o dia em que essa chamada foi efetuada. Parte do código encontra-se em **anexo**.
- **futureBilhetes** o objetivo desta classe é analisar todos os bilhetes comprados comparando a data atual com a data do evento e caso o estado do bilhete não seja usado a classe muda esse estado para expirado. Parte do código encontra-se em **anexo**
- **WSRegisto** esta classe tem como propósito fazer o registo de um utilizador. Parte do em **anexo.**
- **WsDestaques** De momento a app ainda não usufrui desta classe mas o seu propósito é devolver os três eventos com maior taxa de adesão.

5. CONCLUSÃO

Classificar o próprio trabalho é sempre uma tarefa complicada de realizar mas no geral pode dizer-se que o estágio foi bastante produtivo. Houve uma evolução clara no trabalho em equipa que teve de ser elaborado ao longo do estágio, apesar de ser complicado é sempre necessário chegar a um consenso comum caso contrário, num contexto de projeto como este, o resultado final pode ser muito fraco comparado com o esperado.

Inicialmente definiu-se que a plataforma, tanto a aplicação android como o website desenvolvidos deveriam ser capazes de criar eventos, suas restrições e requisitos como também uma forma de visualizar o caminho do utilizador para o evento utilizando as tecnologias de GPS no entanto estas, entre outras, funcionalidades foram abandonadas pela falta de tempo contudo aplicação android contém o principal definido no início, sendo esse filtros de eventos, localização, conversor de url em imagem, chamadas POST e GET da API e um sistema de login (login e registo de utilizadores) com o historial e eventos seguidos pelo utilizador.

No contexto de evolução na programação android também este foi produtivo sendo possível aprender bastante sobre como funciona uma app android, não esquecendo o trabalho desafiante onde foi perdida a maior parte do tempo para que o resultado final fosse perfeito, o que é bastante difícil pois os bugs aparecem sempre e como tal a aplicação ainda tem alguns como o loading ser demorado.

6. REFERÊNCIAS

1. https://www.salesforce.com/br/products/what-is-salesforce/
2. https://trailhead.salesforce.com/pt-BR/home
3. https://developer.android.com/
4. https://www.loba.pt/pt/

Anexo

schedule_bilhete

```
global class schedule_bilhetes implements Schedulable {
    global void execute(SchedulableContext ctx) {
        futureBilhetes reminder = new futureBilhetes();
        Date data = system.today();
        reminder.Bilhetestado(data);
    }
}
```

future_bilhete

```
List<Bilhete__c> BilhetesEstado = [select id, Estado__c,Evento__c from Bilhete__c
                                   where Evento__c =: [select id from Evento__c where id =: EventosID]];
List<Evento__c> EventosComBilhete = [select Data__c,id from Evento__c where id =: EventosID];
List<Bilhete__c> actualizar = new List<Bilhete__c>();

for(Evento__c c : EventosComBilhete){
    for(Bilhete__c b : BilhetesEstado){
        if(b.Evento__c == c.id){
            //+7 para indicar o estado concluido
            if(data > c.Data__c + 7){
                //expirou
                //
                if(b.Estado__c!='Expirado' && b.Estado__c!='Usado'){
                    b.Estado__c='Expirado';
                    actualizar.add(b);
                }
```

WSRegisto

```
@HttpPost
global static sessao__c Registo (String email, String secret, String nome, String data, String telemovel) {
    //lista para todas as sessoes do utilizador

    List<contact> allEmail = [select email from contact];
    List<contact> userRegistado = new List<contact>();
    boolean emailExiste = false;

    for(contact c : allEmail){
        if(c.Email==email){
            emailExiste = true;
            return null;

        }
    }
```

```
if(!emailExiste){
    contact a = new Contact();
    a.Email=email;
    a.password__c = secret;
    a.Birthdate = dataNasc;
    a.LastName = nome;
    a.Phone = telemovel;
    insert a;
```