Relatório de Projeto

Tiago Alexandre Pereira dos Santos

Engenharia Informática

fev | 2023

GUARDA
POLI
TÉCNICO

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO EM CONTEXTO DE ESTÁGIO

TIAGO ALEXANDRE PEREIRA DOS SANTOS

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMMÁTICA

Fevereiro/2023

Agradecimentos

Queria começar por agradecer aos meus pais, e à minha restante família por me terem proporcionado esta oportunidade e por toda a ajuda que me deram ao longo da minha vida e principalmente ao longo do meu percurso académico.

Um agradecimento ao meu orientador, Doutor José Carlos Martins Fonseca, por toda a disponibilidade e paciência ao longo deste trabalho e por todas as sugestões que tornaram este trabalho o melhor possível.

Um obrigado especial à entidade de acolhimento, Armis Group, por me ter dado a oportunidade de trabalhar como estagiário e mais tarde como membro da empresa, numa área completamente nova, pelo conhecimento que me transmitiram e por toda a ajuda sempre que necessitei.

Agradeço também aos meus amigos, que desde sempre acreditaram e estiveram presentes quando eu precisei, para me dar força, ânimo e por todos os momentos que me proporcionaram ao longo destes últimos anos.

Por fim, agradeço á minha namorada, que esteve sempre presente para me dar força, ânimo, motivação, por nunca me deixar desistir, transmitindo-me sempre todo o apoio, paciência e amor.

A todos o meu sincero obrigado!

Ficha de identificação | Elementos Identificativos

Aluno:

Nome: Tiago Alexandre Pereira Santos

Número: 1012045

Curso: Engenharia Informática

Estabelecimento de Ensino:

Escola Superior de Tecnologia e Gestão – Instituto Politécnico da Guarda

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

Telefone: 271220120 **Fax:** 271220150

Entidade de Acolhimento:

Armis Group

Morada: Rua do Freixo 725, 4300-217, Porto

Telefone: 226002295

Duração do Projeto em Contexto de Estágio:

Inicio: 2 de agosto de 2021

Fim: 2 de dezembro de 2021

Orientador do Projeto (Estabelecimento de Ensino):

Nome: José Carlos Fonseca

Grau Académico: Doutor

Orientador do Projeto (Entidade de Acolhimento):

Nome: Hélio Fidalgo

Grau Académico: Mestre

Resumo

O presente documento tem como objetivo relatar o trabalho realizado no projeto em contexto de estágio realizado na empresa Armis Group, na área de sistemas de informação.

Este consistiu no desenvolvimento de uma plataforma web, para dar suporte à criação e gestão de campanhas de fidelização de clientes e substituir um produto já existente que é suportado por terceiros.

Esta nova plataforma web possibilita a gestão de novas campanhas, de forma integrada com as aplicações já existentes do cliente.

Esta nova plataforma permite a criação de novas campanhas, da forma mais dinâmica possível e também possibilitar a gestão de campanhas já existentes. É capaz também de dar apoio ao serviço a clientes, com interfaces que permitem consultar todo o estado do cliente dentro das campanhas.

O objetivo final deste projeto em contexto de estágio foi o desenvolvimento do *front-end* que permite a criação e gestão de campanhas de fidelização, e a realização de toda a lógica necessária para o funcionamento otimizado dessa gestão.

Devido ao curto espaço de tempo do estágio, o objetivo deste projeto não era a implementação completa desta nova plataforma, mas sim uma versão funcional com dados de teste, prevendo-se assim que a mesma só fique completamente desenvolvida em setembro do ano subsequente, em 2023.

Os objetivos deste estágio foram concluídos com sucesso, visto que no final obteve-se uma versão funcional da plataforma no que diz respeito à gestão de campanhas.

Palavras-chave: front-end, OutSystems, ASP.NET, Azure *DevOps*, Rest API

Abstract

This document aims to report the work carried out in the project in the context of an internship carried out at the company Armis Group, in the area of information systems.

This consisted of the development of a web platform, to support the creation and management of loyalty campaigns and will also replace an existing product that is supported by third parties.

This new web platform will enable the management of new campaigns, in an integrated way with the existing applications of the client.

This new platform will allow the creation of new campaigns in the most dynamic way possible and enable the management of existing campaigns. It should also be able to support customer service, with interfaces that allow to consult the entire state of the client within the campaigns.

The final objective of this project in the context of internship was the development of the front end and allow the creation and management of loyalty campaigns, and to realize all the necessary logic for the optimized operation of this management, thus missing for the next stages of the project the requirements that were not addressed at this stage.

Due to the short time of the internship, the objective of this project was not the complete implementation of this new platform, but only a functional version with test data, and it is expected to be fully developed in September of the following year.

The objectives of this internship were successfully completed, since in the end a functional version of the platform was obtained throughout the campaign management.

Keywords: front-end, OutSystems, ASP.NET, Azure *DevOps*, Rest API

Índice Geral

Índice de figuras | Índice de ilustrações

Índice de tabelas

Lista de siglas e acrónimos

API – *Application Programming Interface*

ATM – *Automated Teller Machine*

BL – *Business Logic*

DS – *Digital Sports*

ESTG – Escola Superior de Tecnologia e Gestão

FID – Código de projeto.

FT – *Financial Technology*

GET – Método HTTP

HTML - *HyperText Markup Language*

HTTP - *Hypertext Transfer Protocol*

IDE – *Integrated Development environment*

IoT – *Internet of Things*

IPG – Instituto Politécnico da Guarda

IS – *Integration Services*

IT – *Information Technology*

ITS – *Inteligent Transport Systems*

LIB – *Library*

PACTH – Método HTTP

POST – Método HTTP

PUT – Método HTTP

RESS – *Responsive through Server-Side*

REST – *Representational State Transfer*

SAP - *Systems, Applications, and Products in Data Processing*

SOAP - *Simple Object Access Protocol*

TFVC – *Team Foundation Version Control*

TH – *Theme*

TI – Tecnologias da Informação

UC – Unidade Curricular

UML – *Unified Modeling Language*

URL – *Uniform Resource Locator*

XML - *Extensible Markup Language*

XP – *Extreme Programming*

1 Introdução

O presente documento, descreve, no âmbito da Unidade Curricular (UC) Projeto de Informática do 3º ano da licenciatura de Engenharia Informática, lecionada na Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico da Guarda (IPG), a realização do projeto em contexto de estágio decorrido na empresa Armis FT (*Financial Tech*), localizada em Lisboa.

Este projeto teve uma duração de 4 meses, e nele é desenvolvida uma nova plataforma de gestão de campanhas de fidelização, para um cliente da Armis na área da banca, mais concretamente uma instituição de crédito.

Esta plataforma tem como objetivo a capacidade de criar e gerir campanhas de fidelização de clientes já existentes.

Este projeto é motivado pela necessidade do cliente em ter uma nova plataforma de gestão de campanhas com mais recursos do que os que a plataforma atual possui.

Vai também dar a possibilidade ao cliente de economizar na manutenção visto que a plataforma atual é gerida e mantida por terceiros, o que fica mais dispendioso.

1.1 Caraterização sumária da instituição

De acordo com a Armis (ARMIS, s.d.), esta foi fundada em 2005, com sede na cidade do Porto, é hoje uma entidade na vanguarda na área de TI (Tecnologias de Informação).

Com uma forte presença no mercado nacional, com escritórios no Porto e Lisboa, a Armis abraça também diversos projetos internacionais em países com o Egito, Rússia, Espanha, Angola e Irlanda.

O crescimento significativo e exponencial da Armis e participação em novos projetos internacionais, revelou a necessidade em continuar com o processo de internacionalização e dar vida à Armis Benelux, que tem sede na cidade de Utrecht, nos Países Baixos. (ARMIS, s.d.)

Paralelamente, a expansão *overseas* para São Paulo, a sede da Armis Brasil, consolidou a presença do grupo em toda a América Latina com novos projetos e clientes.

A Armis tem como principal objetivo oferecer as melhores soluções tecnológicas aos clientes. Para que tal seja possível e de modo a responder às especificações de cada projeto e de cada cliente ou negócio, a Armis é composta por 5 grupos: (ARMIS, s.d.)

- **Armis IT –** Na Armis IT, a empresa dedica-se ao desenvolvimento de soluções tecnológicas ambiciosas na área das tecnologias da informação.
- **Armis ITS –** A Armis ITS está empenhada na digitalização e descarbonização do transporte e da mobilidade.
- **Armis DS –** Na Armis DS, pretende-se definir o futuro da tecnologia no setor do desporto, criando soluções digitais especializadas em tecnologia desportiva.
- **Armis FT –** A Armis FT desenvolve soluções de Banca Digital que permitem às instituições gerir os desafios de segurança e a necessidade de inovação contínua.
- **Ozono –** Na Ozono desenvolve-se competências para fornecer um serviço completo aos clientes através do seu centro de operações de IT com elevada disponibilidade, custos reduzidos e manutenção constantes em cada projeto.

Na figura 1, podemos ver o logótipo da Armis.

Figura 1 - Logótipo ARMIS. (Fonte: Armis Group)

1.2 Motivação | Enquadramento

A principal motivação na realização deste projeto e da entrada nesta empresa, foi a oportunidade de frequentar um estágio num ambiente empresarial, ainda que com a pandemia de Covid-19, tenha sido em regime remoto. Mas também que me permitisse desenvolver capacidades tanto a nível técnico como também a nível interpessoal, e de comunicação com colegas e clientes.

Um dos outros fatores de grande importância, foi também trabalhar numa área muito cativante que é a área financeira, que foi interessante e desafiante ao mesmo tempo porque comecei a trabalhar numa tecnologia que desconhecia como é o caso do OutSystems.

1.3 Descrição do problema

A participação no estágio inclui o desenvolvimento de uma aplicação de gestão de campanhas e consulta de clientes para um cliente na área da banca.

O problema está relacionado com a empresa ter uma plataforma de BackOffice desatualizada, com interfaces antigas, que não são *user-friendly* para uso diário. Prende-se também com um motor, ou seja, uma ferramenta que é capaz de receber e analisar os dados diários de cada cliente (por exemplo, transações ou resgates de prémios ou vouchers), que já não tem capacidade de gerir o volume de dados com que a empresa lida no momento.

Está relacionado também com a difícil manutenção, dada a constante necessidade de ser adaptado a novas realidades e formatos de dados da empresa, para além de necessidade de correção de alguns erros de cálculo encontrados. E como é realizada por uma empresa externa para garantir o seu funcionamento torna a sua manutenção muito dispendiosa, tanto em termos monetários como de burocracia.

E essa necessidade de um novo *backoffice*, vem com o facto da empresa necessitar de uma plataforma gerida e mantida internamente que seja mais económica, sem recurso a empresas externas com um *software* mais caro e difícil de manter.

Portanto, o objetivo mantém-se com o desenvolvimento completo de um *backoffice* capaz de servir o mesmo propósito da plataforma já existente, mas com mais funcionalidade e desenvolvida com tecnologias mais recentes.

Mais concretamente, no caso da equipa onde estou inserido é o desenvolvimento de uma plataforma de gestão de campanhas e de associação de clientes às campanhas.

1.4 Objetivos

O objetivo principal é o desenvolvimento de uma plataforma web de Gestão e Criação de Campanhas de fidelização para uma instituição de crédito na área da banca, permitindo assim que seja feita a criação e parametrização dos mais diversos tipos de campanhas e a sua gestão e possível edição. E prende-se também com a respetiva gestão de clientes e possibilidade de os poder adicionar a uma campanha a qualquer altura.

Após a parametrização do programa, a plataforma web deverá ser capaz de aceder a um *dashboard* para ver como o motor do programa está a executar e identificar os erros existentes no motor e possibilidade de mandar reprocessar alguns processos necessários para o bom funcionamento e cálculo da aplicação.

Esta solução, será também capaz de permitir uma consulta aos clientes que pertencem a cada campanha, as suas respetivas conta-correntes e também os benefícios atribuídos manualmente ou por cálculo de programa. Isto servirá assim para ajudar o serviço de apoio a cliente para lidar com qualquer dúvida e reclamação proveniente dos clientes.

Especificamente, os objetivos desta plataforma web para o presente trabalho de estágio são:

- **Gestão de programas** – Permitir a criação, edição, consulta e remoção de programas de fidelização a partir de parametrizações pré-estabelecidas.
- **Regras de cálculo** – Permitir a criação, edição, consulta e remoção de um método de cálculo associado a uma campanha de fidelização.
- **Transações para exclusão de cálculo** – Permitir a parametrização de transações que serão contempladas ou não, no cálculo de um determinado programa de fidelização, e estão associadas a uma regra de cálculo.

1.5 Estrutura do documento

Este documento é constituído por oito capítulos, que a seguir se descrevem.

No capítulo 1, **Introdução**, encontra-se uma breve descrição da instituição de acolhimento, a motivação que levou à criação da plataforma realizada, uma descrição do problema inicial, um resumo dos objetivos definidos inicialmente e o plano de estágio definido pela organização.

No capítulo 2, **Estado de Arte**, está uma análise de campanhas de fidelização e de fidelização na área financeira, neste também são apresentadas algumas das soluções existentes na área de gestão de campanhas de fidelização.

No capítulo 3, **Metodologia**, é definida a metodologia utilizada no desenvolvimento do projeto.

Em seguida no capítulo 4, **Análise de Requisitos**, é descrita a análise de requisitos, incluindo o diagrama de contexto, o diagrama de casos de uso, entre outros.

Mais à frente do capítulo 5, **Tecnologias,** são apresentadas as tecnologias utilizadas no desenvolvimento do projeto.

O capítulo 6, **Implementação**, é composto pela descrição da implementação feita durante os meses de desenvolvimento da aplicação, contendo algumas imagens na interface da mesma e da sua lógica.

No capítulo 7, **Verificação e Validação**, pode ser analisado o processo de testes e verificações realizados de forma a garantir o bom funcionamento de cada passo da aplicação.

Finalmente no capítulo 8, **Conclusões**, são apresentadas as conclusões do trabalho realizado.

2 Estado da Arte

Pretende-se, com este capítulo, fazer uma abordagem ao estado da arte atual sobre um ponto de vista tecnológico, analisando alguns dos conceitos utilizados e algumas das aplicações relevantes, no âmbito da fidelização do cliente.

2.1 Fidelização de clientes

Atualmente vivemos num contexto no qual temos à nossa disposição uma quantidade quase ilimitada de opções quando pretendemos adquirir um produto ou serviço.

Paralelamente, a generalidade das empresas constata que é muito mais difícil e mais caro adquirir um novo cliente do que manter um cliente já existente (Pis Marketing, s.d.).

De acordo com a (InboundCycle, 2023), a fidelização de clientes é um conceito de marketing que pretende, por meio de diferentes estratégias e técnicas de marketing e vendas, que os consumidores que obtiveram anteriormente qualquer um dos produtos ou serviços da empresa, o continuem a fazer e se tornem clientes regulares.

A fidelização de clientes tem impacto diretos nos resultados financeiros, bem como na imagem e no prestígio da marca. A influência de um cliente satisfeito pode ser mais decisiva do que qualquer estratégia de marketing.

Programas de fidelização na área financeira

Uma estratégia que muitas empresas têm adotado com o objetivo de fomentarem a lealdade dos seus clientes e de aumentarem a sua frequência de compra está relacionada com os chamados programas de fidelização.

Um programa de fidelização tem o propósito de incentivar o consumidor a comprar várias vezes em troca de um determinado benefício oferecido pela empresa (Pis Marketing, s.d.).

Os programas de fidelização são ações estratégicas impostas com o propósito de persuadir os clientes de modo a aumentar a sua rentabilidade a longo prazo (Outvio, s.d.).

De acordo com a (Outvio, s.d.), as vantagens dos programas de fidelização a clientes são:

- Permitem incrementar as receitas;
- Melhoram a reputação da marca;
- Aumentam a conversão e a taxa de recompra:
- Fornecem informação fiável sobre o comportamento do cliente;
- Aumentam os níveis de satisfação do cliente.

Na área financeira, programa de fidelização é uma estratégia de marketing destinada a ajudar os bancos e outras instituições financeiras a reter a sua base de clientes oferecendo-lhes recompensas interessantes.

Alguns destes programas, oferecem juros mais altos nas contas-poupança dos clientes, descontos nos empréstimos, taxas mais baixas no uso de ATM's, entre outros (AnnexCloud, s.d.).

2.2 Trabalhos relacionados

Nesta secção são analisadas algumas das soluções existentes no âmbito de gestão de campanhas de fidelização, que foram escolhidas porque uma delas é das maiores plataformas de fidelização a clientes do mundo (*Annex Cloud*) uma, subsidiária de uma multinacional, que está sediada em Portugal (*Sogec*) e uma empresa portuguesa (Loyty).

Todas estas empresas produzem software que não especificamente direcionado para gerir plataformas de fidelização a clientes no âmbito financeiro. Visto que para este propósito é necessário construir um software à medida de cada instituição financeira.

Loyty

Loyty é um software de fidelização que vai muito além do cartão cliente. Não só assegura a angariação de dados do cliente, como também a sua gestão e comunicação.

Através da análise do perfil de compra poderá segmentar e comunicar de forma adaptada e automatizada. (Loyty, s.d.)

As funcionalidades principais do Loyty são:

- Segmentar e construir o perfil de cliente.
- Utilizar as ferramentas de marketing mais adequadas a cada cliente.
- *Dashboards* e ferramentas de análise.
- Campanha de atribuição de benefícios.
- Integração com plataformas de diferentes fabricantes.

Annex Cloud

Fundada em 2010, a *Annex Cloud* evoluiu de uma ferramenta de comércio social para uma plataforma respeitada de gestão de programas de fidelização.

Hoje ajudam os seus clientes a familiarizarem-se com os dados dos seus clientes para assim aumentarem o lucro a partir da fidelização. (AnnexCloud, s.d.)

Este serviço é extremamente flexível e possui vários componentes, como por exemplo:

- Programa de recompensas de fidelização
- Sistema por hierarquias
- Assinatura paga
- Soluções para negócios complexos
- Promoções e incentivos

Sogec

A Sogec é uma subsidiária da Sogec Marketing França, líder europeu de Marketing Promocional e que integra o grupo La Poste (Sogec, s.d.).

Com 47 anos de experiência e presença em 7 países europeus, é especialista na implementação e gestão de estratégias promocionais (Sogec, s.d.).

A *Sogec* fornece uma solução em que se pode desenvolver campanhas para marcas e controlar o processo na sua totalidade.

A sua ferramenta de gestão de campanhas fornece as seguintes funcionalidades:

- Promoção de vendas.
- Reduz trabalhos administrativos.
- Dá visibilidade dos resultados das ações em tempo real.
- Fornece uma análise de resultados e visibilidade das campanhas em tempo real.

Todas estas aplicações acima referidas, não reúnem os requisitos necessários para suprimir todas as necessidades de uma instituição financeira, visto que estas aplicações são mais viradas para a parte comercial e de venda de produtos em contexto de *e-commerce*.

3 Metodologia

Para o desenvolvimento de *software*, nomeadamente de uma plataforma *web*, devem-se seguir certas normas e critérios de forma a agilizar o processo. Para que isto seja possível, podem ser utilizadas diversas metodologias de desenvolvimento.

Neste projeto, optou-se por utilizar uma metodologia ágil, mais especificamente a metodologia Scrum. Esta metodologia foi escolhida pelo cliente, por ser a metodologia utilizada nos restantes projetos realizados pelos mesmos.

A metodologia Scrum é a *framework agile* de desenvolvimento mais popular do mundo, existindo, no entanto, outras metodologias ágeis, como o Kanban, o XP, Crystal e o Scrumban. Todas estas permitem desenvolver produtos e serviços com maior agilidade e por isso estão a ganhar cada vez mais adeptos pelo mundo (ScrumPortugal, s.d.).

3.1 Metodologia Scrum

A metodologia *scrum* é uma forma de desenvolver produtos e serviços e também uma *framework* ágil para o planeamento e gestão de projetos de *software*.

O *Scrum* recomenda que os projetos sejam divididos em ciclos, normalmente chamados de *sprints*, com uma duração, geralmente, de aproximadamente 4 semanas. Ao final de cada ciclo, o objetivo será que a equipa tenha algo concreto para entregar ao cliente, ou seja, uma demonstração de evolução no projeto ao longo daquele ciclo que acabou de ser concluído. Em função disso, o *Scrum* permite que sejam atingidos resultados menores e de uma forma contínua ao longo do projeto. Com toda esta ligação entre membros e com a participação ativa dos clientes, obtém-se melhores resultados no desenvolvimento do projeto. Na figura seguinte, é ilustrado o processo da metodologia *Scrum*.

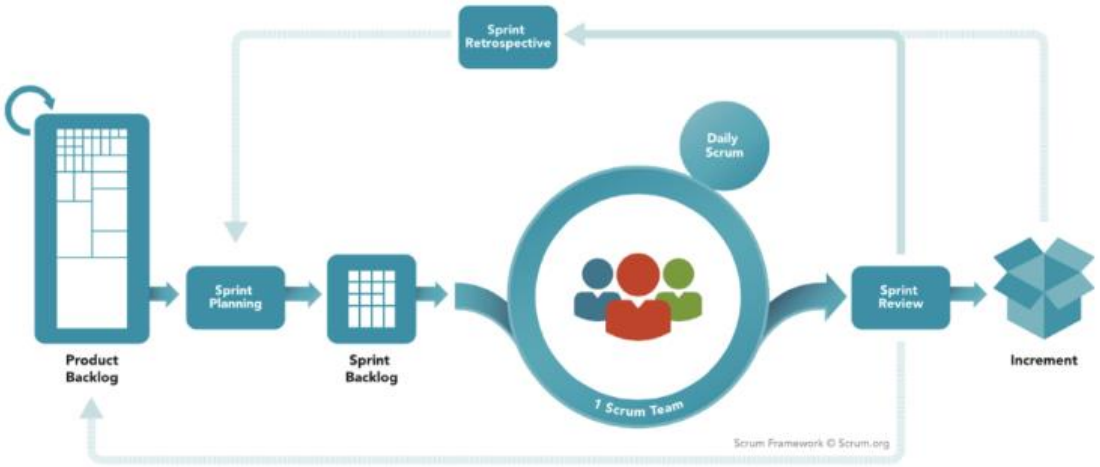


Figura 2- Processo Scrum (Fonte: Scrumportugal,2022)

Em seguida são explicados os processos utilizados por esta metodologia (Scrum Portugal, s.d.):

- ***Product Backlog:*** é a lista principal que contém todas as funcionalidades desejadas para um produto, normalmente definida pelo *Product Owner.*

- ***Sprint Planning Meeting:*** o trabalho a ser desenvolvido durante o *sprint* é habitualmente planeado durante esta reunião, onde estarão presentes o *Product Owner, Scrum Master* e a equipa de desenvolvimento. Serão descritas todas as tarefas a ser feitas ou todos os objetivos a ser desenvolvidos, dando assim como resultado o *Sprint Backlog.*

- ***Sprint Backlog:*** é uma lista de tarefas ou objetivos que a equipa de desenvolvimento se compromete a realizar durante o *sprint.* Estas tarefas são extraídas do *Product Backlog* com base nas prioridades que o *Product Owner* definiu.

- ***Daily Scrum:*** todos os dias é feita uma reunião pela equipa. Esta tem como objetivo disseminar conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho a ser realizado no dia seguinte.

- ***Sprint Review Meeting:*** no final de cada *sprint* é realizado um *Sprint Review Meeting* onde se demonstra tudo aquilo que foi conseguido durante o *sprint*, habitualmente em forma de demo contendo todas as novas funcionalidades. Depois do *feedback*, o *Product Owner* decide se deve ou não realizar o incremento com as novas funcionalidades.

- ***Sprint Retrospective:*** a retrospetiva é onde a equipa se reúne par documentar e discutir o que funcionou, mas mais o que não funcionou durante o *sprint.* Em seguida, é discutido aquilo que pode ser ou não melhorado e que intervenções serão tomadas para que tal aconteça.

- ***Incremente:*** é o produto final após cada *sprint.*

A figura 3, demonstra a organização de um sprint, o *Sprint Backlog*.

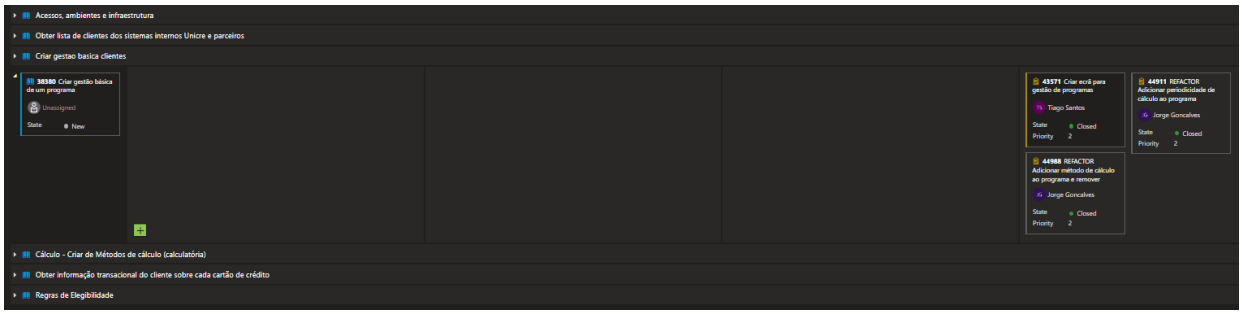


Figura 3 - Organização de um sprint

(Fonte: Elaboração própria)

E a figura 4 ilustra a descrição de uma *User Story*.

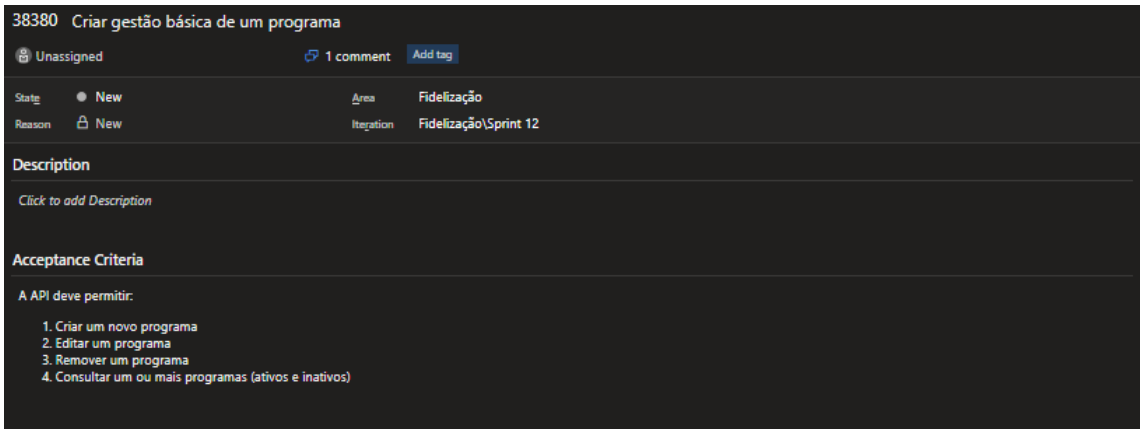


Figura 4 - Descrição de uma User Story

(Fonte: Elaboração própria)

A metodologia scrum define os três papéis chave seguintes:

- **Product Owner** - É a pessoa responsável pelo product backlog utilizada para expressar quais as funcionalidades com maior prioridade e comunicar as tarefas aos restantes elementos da equipa. É igualmente responsável por definir os sprints e realizar demonstrações das diversas fases do produto até ao desenvolvimento do produto final.
- **Scrum Master** - É o gestor da equipa que garante que as tarefas sejam realizadas pelos diversos membros sem qualquer tipo de inconveniente, que é responsável por que a equipa siga as boas práticas da metodologia scrum. É o representante da equipa perante o product owner e protege a equipa de interferências externas.
- **Tech Lead –** É um programador sénior que ajuda a equipa em toda a fase de desenvolvimento, mas que também possui capacidades de liderança e que também ajuda na gestão da equipa.
- **Team Member** - É o elemento da equipa de desenvolvimento que tem como principais objectivos o desenvolvimento do produto e o de tornar as tarefas do sprint backlog em funcionalidades potencialmente concluídas no fim de cada sprint.

De acordo com a descrição de papeis referidos acima, a criação da plataforma web ficou a cargo de uma equipa de desenvolvimento da área financeira. Esta equipa é constituída por 4 elementos, sendo eles os seguintes:

- Fábio Maiorano: Gestor de projeto
- Vasco Ferreira: *Tech Lead*
- Jorge Gonçalves: Programador Sénior (Responsável pelo *back-end* da aplicação)
- Tiago Santos: Estagiário do Instituto Politécnico da Guarda / Programador Júnior (Responsável pelo *front-end* da aplicação)

3.2 Plano de estágio | Etapas do Estágio

Em baixo encontram-se as fases do planeamento relativas ao projeto.

02/08 a 21/08 – Primeiro contacto com a empresa e realização de testes a fim de apurar as capacidades do estagiário.

30/08 a 24/09 – Primeiro contacto com a plataforma *Outsystems* e realização de miniprojectos para familiarização com a mesma.

27/09 a 08/10 – Estudo da documentação do projeto, análise do design fornecido pela equipa de design e realização dos primeiros esboços da interface em *Outsystems*.

11/10 a 29/10 – Sprint 1: Implementação da camada de serviços.

01/11 a 19/11 – Sprint 2: Implementação da camada intermédia e início da adaptação da interface aos serviços fornecidos.

22/11 a 02/12 – Sprint 3: Continuação de implementação de serviços e interface.

3.3 Detalhes da metodologia no desenvolvimento do Hub Fidelização

Numa fase inicial do projeto foram realizadas reuniões entre todos os membros da equipa (incluindo o gestor de projeto) e o *Product Owner*, de forma a levantar todas as funcionalidades que iriam ser desenvolvidas nos meses seguintes. Após a conclusão da fase de planeamento, deu-se início à fase de implementação, ou seja, os *sprints* onde foram desenvolvidas as funcionalidades anteriormente definidas.

Durante a fase de implementação foram realizadas reuniões diárias (*Daily Scrum*), geralmente da parte da manhã com a equipa de desenvolvimento, para analisar as tarefas feitas no dia anterior. Esta reunião servia também para analisar algum bloqueio que poderia ser discutido em equipa, e planear as tarefas que iriam ser implementadas no próprio dia. Durante a mesma fase, durante uma ou duas vezes por semana, o *Product Owner,* também comparecia nas reuniões para resolver algumas dúvidas ou bloqueios da equipa.

No final de cada *sprint*, eram realizadas reuniões periódicas entre a equipa, o gestor de projeto e o *Product Owner,* para assim ser apresentado um feedback do trabalho realizado ao longo do *sprint* (*Sprint Review*), e identificar melhorias em linha com o *Product Owner,* para assim cada vez se aproximar mais do resultado esperado pelo mesmo. No final, eram indicadas as melhorias a ser apresentadas e tudo o que correu bem ou menos bem no *sprint* transato (*Sprint Retrospective*).

Após isto, era realizada uma nova reunião para assim pedir permissão para se abrir um novo sprint e alinhar com a equipa as tarefas do *sprint* seguinte.

Durante o tempo de estágio apresentado neste relatório, foram realizados por volta de 3 *sprints*, sendo este de entre 2 e 3 semanas, dependendo das tarefas a realizar. Após este estágio, o projeto continuou sendo assim feitos mais *sprints.*

4 Análise de requisitos

A análise de requisitos de um sistema é um processo muito importante na elaboração do projeto. Com ela é possível analisar, documentar e verificar os requisitos do sistema de modo que o *software* se torne mais fácil de desenvolver e definir assim mais claramente tudo aquilo que o *software* terá de realizar para assim satisfazer todas as necessidades da organização e dos seus clientes.

Durante a fase inicial deste projeto, a equipa que foi encarregue de desenvolver este projeto, em conjunto com *Product Owner*, conseguiu apurar vários requisitos que vão ajudar a desenvolver a solução proposta.

O requisito principal da interface da plataforma ser desenvolvida em *Outsystems*, prende-se com facto de o cliente ter uma equipa apenas focada a contruir ferramentas para dar suporte às aplicações desenvolvidas em *Outsystems*.

Na implementação, no contexto deste relatório de estágio, foram abordadas as seguintes funcionalidades necessárias para a gestão de campanhas:

- Criação de um programa – Permite ao utilizador ser capaz de criar um programa de fidelização e as suas respetivas regras de cálculo.
- Consultar uma lista de programas –Permite ao utilizador ser capaz de apresentar uma lista de programas de fidelização guardados na base de dados.
- Consultar um detalhe de um programa – Permite ao utilizador a capacidade de consultar um detalhe de um programa de fidelização e a sua respetiva regra de cálculo.
- Editar um programa – Permite ao utilizador ser capaz de editar o detalhe de um programa de fidelização e a sua respetiva regra de cálculo.
- Inativar um programa – Permite ao utilizador a capacidade de inativar um programa de fidelização, já em execução.
- Eliminar um programa – Permite ao utilizador a capacidade de eliminar um programa de fidelização e tudo a que lhe está associado.

Neste capítulo são descritos o diagrama de contexto e os seguintes itens da linguagem UML (Unified Modeling Language):

- Diagrama de casos de uso e algumas descrições dos respetivos casos de uso.
- Diagrama de classes.
- Dicionário de dados.

4.1 Diagrama de Contexto

O diagrama de contexto mostra as relações estabelecidas entre o sistema, ou aplicação, e o meio ambiente, apresentado o sistema com um único processo.

Na figura 5, é apresentado o diagrama de contexto que traduz a solução para o problema apresentado.

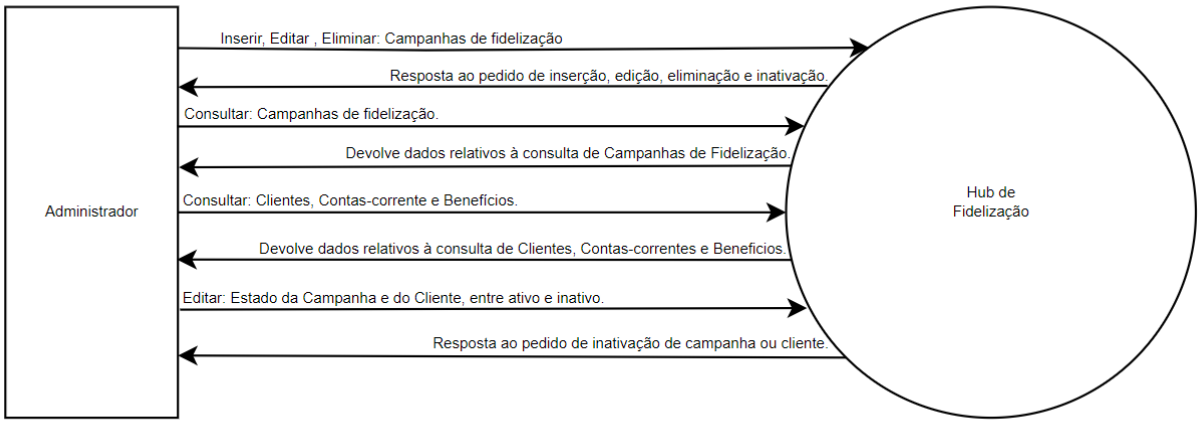


Figura 5- Diagrama de Contexto

(Fonte: Elaboração Própria)

4.2 Diagrama de Casos de Uso

O diagrama de casos de uso permite ver, de forma prática e concisa, todas as funcionalidades existentes na aplicação, assim como todas as interações que cada ator tem com ela.

Na figura 6, podemos ver o resultado inicial dos casos de uso apenas porque no contexto deste relatório de estágio, a aplicação não estará finalizada. A implementação debruçar-se-á sobre os casos de uso que estão dentro do quadrado assinalado na figura.

Inicialmente apenas foi definido um tipo de utilizador, visto que para efetuar toda a gestão de programas de fidelização, só os utilizadores com o perfil de administrador podem efetuar essa gestão.

Os tipos de utilizador vão ser alargados nas próximas etapas do projeto, após o presente estágio, consoante a necessidade de desenvolvimento futuro de novas funcionalidades.

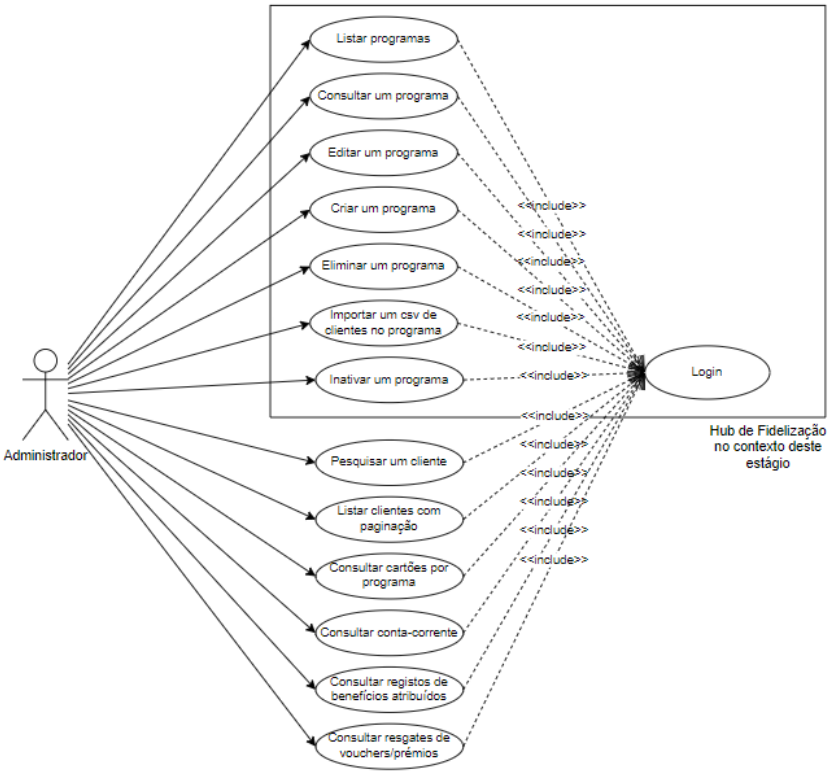


Figura 6- Diagrama de casos de uso

(Fonte: Elaboração Própria)

4.3 Atores e respetivos Casos de Uso

Na tabela 1 podemos ver os atores (neste caso apenas o administrador, pela mesma razão que foi referida no subcapítulo anterior), os casos de uso em que este participa e os seus objetivos.

Tabela 1 - Atores e respetivos casos de uso

Ator	Casos de uso	Objetivo
Administrador	Listar programas com paginação	Listar e consultar campanhas de fidelização já criadas com os seus respetivos estados de ativo ou inativo.
	Editar um programa	Editar uma campanha de fidelização, incluído os dados do programa e as respetivas regras de cálculo.
	Criar um programa	Criar uma campanha de fidelização, incluído os dados do programa e as respetivas regras de cálculo.
	Eliminar um programa	Eliminar uma campanha de fidelização, incluído os dados do programa e as respetivas regras de cálculo.
	Inativar um programa	Inativar uma campanha de fidelização já decorrer e poder inativá-la de acordo com um conjunto de regras específicas.
	Consultar um programa	Consultar a parametrização de uma campanha já existente.

4.4 Descrição dos Casos de Uso

Nesta secção podemos ver, em forma de tabela, a descrição dos casos de uso apresentados na secção anterior.

Esta tabela é composta pelos seguintes campos:

- **Nome:** indica o nome do caso de uso.
- **Descrição:** descreve o objetivo em que o caso de uso consiste.

- **Pré-Condição:** indica todas as condições necessária para que o caso de uso possa ser iniciado.
- **Caminho Principal:** descreve todas as etapas do caso de uso entre o ator e o sistema.
- **Caminhos Alternativos:** descreve validações de campos e operações anormais ao caminho principal.
- **Suplementos ou Adornos:** indica os casos de teste concretos ao caso de uso.
- **Pós Condições:** caso existam, descrevem operações necessárias após a finalização do caso de uso.

Descrição do Caso de Uso "Listar Programas com Paginação"

O objetivo deste caso de uso é permitir ao administrador navegar para o ecrã de "Lista de Programas" e assim aparecerem as campanhas paginadas (Tabela 2).

Tabela 2 - Descrição do caso de uso "Listar Programas com paginação"

Nome	Listar Programas com paginação.
Descrição	O objetivo é o ator "Administrador" poder consultar uma lista de campanhas já existentes.
Pré-Condição	O ator tem de efetuar um login válido.
Caminho Principal	1. O ator encontra-se na página inicial. 2. O ator seleciona o botão "Lista de Programas". 3. O sistema faz uma passagem entre ecrãs, e apresenta assim o ecrã com a lista de programas devidamente paginados, e com a possibilidade de navegar nessas mesmas páginas, e ainda com o filtro de ativos, inativos e todos.
Caminhos Alternativos	3. O sistema apresenta uma mensagem de informação, caso não existam programas criados.
Suplementos ou adornos	N/A

Pós-Condição	N/A

Descrição do Caso de Uso "Criar um programa"

O objetivo deste caso de uso é permitir ao utilizador parametrizar a criação de uma nova campanha de fidelização.

Tabela 3 - Descrição do Caso de Uso "Criar um programa"

Nome	Criar um programa.
Descrição	O objetivo é o ator "Administrador" ter a possibilidade de parametrizar uma nova campanha de fidelização.
Pré-Condição	O ator tem de ter um login válido.
Caminho Principal	1. O ator encontra-se no ecrã onde está apresentada, ou não, uma lista de programas paginados. 2. O ator seleciona no canto superior direito, o botão "Adicionar Programa". 3. O sistema faz uma passagem entre ecrãs, e apresenta assim o ecrã com um formulário com campos de preenchimento, obrigatórios ou não, como por exemplo, nome, descrição, data de início, data de fim, *blockcodes*, produtos, etc. 4. O ator preenche o formulário conforme desejado. 5. O ator seleciona o botão "Regras de Cálculo". 6. O sistema faz uma passagem entre ecrãs e apresenta assim o ecrã com um formulário relativo às regras de cálculo da campanha que está a ser parametrizada. 7. O ator preenche o formulário conforme desejado. 8. O ator seleciona o botão "Guardar". 9. O sistema apresenta uma janela *pop-up* com uma mensagem de confirmação, se o ator deseja mesmo guardar as regras de cálculo parametrizadas.

	10. O ator voltar a selecionar o botão “Guardar”. 11. O sistema faz uma passagem entre ecrãs e volta ao ecrã anterior com o formulário já preenchido previamente. 12. O ator seleciona o botão “Guardar”. 13. O sistema apresenta uma janela *pop-up*, com uma mensagem de confirmação se o ator deseja mesmo guardar a campanha parametrizada. 14. O ator volta a selecionar o botão “Guardar”. 15. O sistema imprime uma mensagem de sucesso, com a mensagem de que a campanha está a ser criada com sucesso, fecha o *pop-up* e volta ao ecrã anterior, que será o ecrã “Lista de Programas”.
Caminhos Alternativos	14. a) O sistema devolve um alerta dos campos que se encontram vazios ou que estão com formato incorreto. 14. b) O sistema devolve um alerta caso já exista uma campanha com o mesmo código. 14. c) O ator cancela a operação de “Guardar” e fecha a *pop-up* de confirmação.
Suplementos ou adornos	• Verificar se os campos obrigatórios estão todos preenchidos. • Verificar se todos os campos estão inseridos no formato correto. • Verificar se existem campanhas existentes com o mesmo código.
Pós-Condição	O sistema envia o pedido para o *back-end*, onde é guardada na base de dados a campanha.

Descrição do Caso de Uso “Consultar um programa”

O objetivo deste caso de uso é permitir ao administrador consultar uma campanha de fidelização.

Tabela 4 - Descrição do Caso de Uso "Consultar um Programa"

Nome	Consultar um programa.
Descrição	O objetivo deste caso de uso é permitir ao administrador consultar uma campanha de fidelização.
Pré-Condição	1. O ator tem de ter um login válido. 2. Já tem de existir pelo menos uma campanha já parametrizada, guardada na base de dados.
Caminho Principal	1. O ator encontra-se no ecrã onde está apresentada uma lista de programas paginados. 2. O ator, na tabela de programas paginados, seleciona o ícone “Lupa”, na coluna dos Detalhes. 3. O sistema faz uma passagem entre ecrãs e apresenta assim o ecrã “Detalhes do programa”, com um formulário já preenchido de acordo com a parametrização na base de dados.
Caminhos Alternativos	2. Não existe nenhuma campanha parametrizada.
Suplementos ou adornos	- Verificar se existem campanhas de fidelização já criadas.
Pós-Condição	N/A

Descrição do Caso de Uso “Editar Programa”

Tabela 5 - Descrição do Caso de Uso "Editar Programa"

Nome	Editar Programa.
Descrição	O objetivo deste caso de uso é permitir ao ator editar os dados pertencentes a uma campanha de fidelização já existente.
Pré-Condição	1. O ator tem de ter um login válido. 2. Já tem de haver pelo menos uma campanha parametrizada, guardada na base de dados.
Caminho Principal	1. O ator tem de seguir todos os passos descritos no caso de uso “Consultar Programa”. 2. O ator seleciona o botão, no canto superior direito, “Editar”. 3. O sistema disponibiliza a edição do formulário que até agora estavam indisponíveis para edição. 4. O ator altera os campos no formulário que lhe são possíveis editar. 5. O ator seleciona o botão “Regras de Cálculo”. 6. O sistema apresenta um novo ecrã com as regras de cálculo, já preenchidas de acordo com a parametrização da campanha escolhida. 7. O ator, altera ou não, os campos que desejar e que lhe são possíveis editar. 8. O ator seleciona o botão “Guardar”, para guardar as regras de cálculo. 9. O sistema apresenta uma janela *pop-up,* para o ator confirmar as regras de cálculo alteradas. 10. O ator volta a selecionar o botão “Guardar”. 11. O sistema volta a apresentar o ecrã com a parametrização já editada. 12. O ator seleciona o botão “Guardar”.

	13. O sistema volta a apresentar uma janela *pop-up,* para confirmar se o utilizador deseja mesmo guardar o programa alterado, ou não. 14. O ator volta a selecionar o botão "Guardar". 15. O sistema apresenta uma mensagem de sucesso, e volta a pôr os campos todos do programa não editáveis.
Caminhos Alternativos	1. O ator cancela qualquer uma das janelas *pop-ups* apresentadas. 2. O sistema devolve um alerta dos campos que se encontram vazios ou que estão com formato incorreto.
Suplementos ou adornos	• Verificar se os campos obrigatórios estão todos preenchidos. • Verificar se todos os campos estão inseridos no formato correto.
Pós-Condição	O sistema guarda na base de dados os novos dados da campanha de fidelização.

5 Tecnologias

Neste capítulo, são apresentadas as várias tecnologias e ferramentas utilizadas no desenvolvimento deste projeto.

5.1 Plataformas *Low-Code*

As plataformas de *low-code* são um tipo de *software* que permite um desenvolvimento rápido e eficiente em pequena ou larga escala (HupData Data Analysis Solutions, s.d.).

A necessidade deste tipo de plataformas apareceu quando começou a haver uma imensa necessidade por aplicações às quais os modelos de trabalho tradicionais não tinham a capacidade de responder, tanto em termos de tempo de desenvolvimento como em termos de satisfação por parte dos clientes.

Um dos principais benefícios no uso deste tipo de tecnologia é o baixo custo e a facilidade de formação, o que permite a alguém sem conhecimento da tecnologia começar a produzir aplicações e software num curto espaço de tempo.

As opções disponíveis são imensas, desde *User Interfaces*, *Data Modelling e Management* ou aplicações *Mobile*. Neste tipo de tecnologias, a componente de programação é feita quase exclusivamente pela plataforma, sem que o programador se aperceba (Silva, 2022).

Um estudo feito pela Forrester Wave (Forrester, s.d.), chegou às seguintes conclusões acerca dos principais benefícios do uso deste tipo de tecnologia:

- Oferece uma variedade vasta de ferramentas que permitem definir fluxos, lógica, dados, diferentes tipos de aplicações e dispositivos suportados.
- Visto ser uma tecnologia de rápido desenvolvimento, facilmente se adapta às necessidades e alterações que os clientes possam necessitar.

- Apesar de muitas plataformas *low-code* necessitarem de licenças para o seu potencial, grande parte disponibiliza um conjunto considerável de ferramentas na sua versão grátis que permite o desenvolvimento de aplicações com alguma complexidade.
- Aparecem como PaaS (*Platform as a Service*), disponibilizando assim os recursos como servidores e bases de dados remotamente, sem que os clientes se tenham de preocupar com um reinvestimento nas suas infraestruturas.

Alguns dos pontos menos positivos destas tecnologias passam pela falta de certificados de segurança, ou a adaptação a temas do presente como Inteligência Artificial ou *Internet of Things* (IoT) (Forrester, s.d.).

Alguns dos exemplos destas plataformas são o Outsystems, Mendix, Microsoft Power Apps, entre outras.

5.2 *OutSystems*

A tecnologia *OutSystems* envolve diversos componentes, sendo que cada um tem um papel importante quando se fala de *OutSystems* como uma aplicação *low-code* e de desenvolvimento acelerado (Outsystems, 2023). Na figura 7, é ilustrada a organização da arquitetura Outsystems.

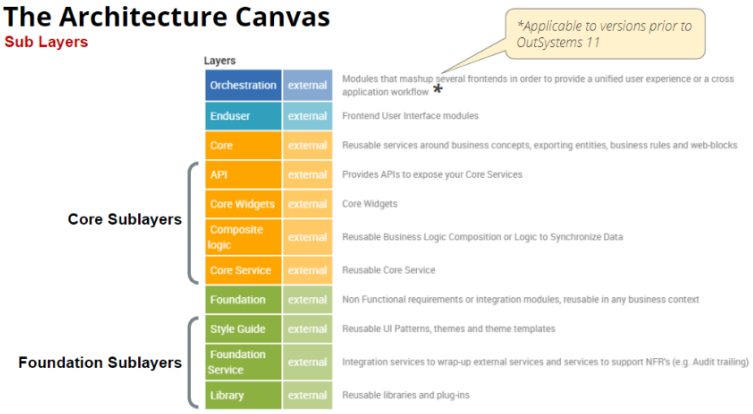


Figura 7 - Arquitetura OutSystems

(Fonte: Outsystems)

Esta tecnologia e os seus componentes foram utilizados em todo o desenvolvimento desta plataforma, desde a integração de serviços até à implementação das interfaces.

Service Studio

O *Service Studio* é o Ambiente de Desenvolvimento Integrado (IDE) onde quem produz *OutSystems* passa a maior parte do seu tempo.

Após fazer a ligação ao servidor de *OutSystems*, pode-se desde logo começar a desenvolver uma aplicação, através da criação de módulos.

Estes módulos podem ser responsivos (inclui os temas ou layouts, que a própria *OutSystems* disponibiliza através do *Forge*, e que carrega logo algumas funções *RESS – Responsive through Server-Side componentes*) ou em branco, onde o programador parte do zero.

Usando estes temas, é gerado todo o código com a capacidade de ser responsivo, de modo à aplicação ficar funcional para tablets, desktops e mobile.

Após a criação dos módulos, é possível começar a definir os modelos de dados, a lógica envolvida ou *workflows* de negócio.

Os módulos podem ser mais tarde referenciados por outros módulos, de modo a criar diversas camadas e transparência dentro da aplicação, ficando tudo o mais isolado possível e com menos dependências. (Outsystems, 2023)

Service Center

No *Service Center* é possível ver um sistema de *logs* de todas as aplicações no servidor *OutSystems*, bem como análise e manutenção.

A plataforma disponibiliza uma pilha de erros individual para cada aplicação, onde é possível ver com algum detalhe o que provocou a falha.

Para cada aplicação, é ainda possível ver definições de segurança, o estado das integrações de API's externas ou *Site Properties* (no fundo são como variáveis globais em Java, mas onde é possível mudar os valores e propagar as alterações no momento).

É possível ainda refrescar dependências desatualizadas ou fazer *downgrades* de versões dos módulos/aplicações. (Outsystems, 2023)

A plataforma disponibiliza também uma *grid* onde é possível analisar tempos e acessos aos diferentes espaces (módulos). No fundo, é uma vista global de todas as aplicações e o seu estado.

Lifetime

O *Lifetime* é similar ao *Service Center*, e é uma parte da plataforma *OutSystems* onde fica registada a performance das aplicações, datas de publicações e respetivo autor, ou o tempo que os pedidos demoram a ser feitos, entre cliente e servidor (Outsystems, 2023).

5.3 Web Services

Nos dias de hoje, quando se fala na *Web*, ouvimos muitas vezes o termo *Web Service*, mas que pode ter significados muito diferentes dependendo do contexto onde é inserido.

Os *Web Services* podem assumir dois tipos de arquiteturas diferentes, *Internal Architecture* (recebem pedidos e reencaminham para outra camada, no fundo são mais uma camada de abstração) e *External Architecture* (que é uma infraestrutura que integra vários *Web Services*).

Independentemente da arquitetura, existem três papéis dentro dos Web Services:

- *Provider* – é quem cria o *Web Service* e o torna disponível para ser usado pelas aplicações consumidoras.
- *Requestor* – quem faz a chamada ao *Web Service.*
- *Broker* – a camada intermédia que facilita a utilização do *Requestor*, abstraindo assim processos complicados ao utilizador e que garante acesso ao *Web Service.*

Na figura 8, é apresentada a organização dos componentes de um *Web Service* baseado em REST.

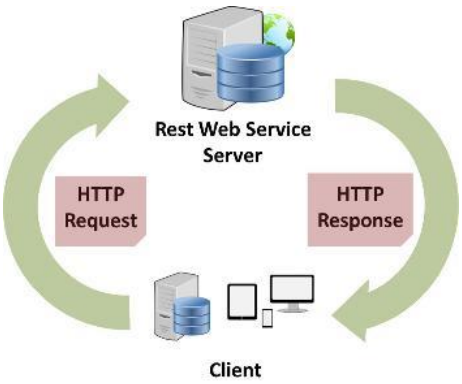


Figura 8 - Web Service baseado em REST

(Fonte: stSofware)

Além disso, os *Web Services* providenciam uma implementação de arquiteturas orientada a serviços, em que o desacoplamento que disponibilizam permite a escalabilidade das aplicações em toda a Internet. (w3, 2022)

Ao nos referirmos a *Web Services*, dois nomes que frequentemente associados a este termo são REST (*REpresentation State Transfer*) e SOAP (*Simple Object Acess Protocol*).

REST

Nesta aplicação foi utilizada a arquitetura REST, por ser mais rápida, permitir uma maior variedade de formatos de dados e fornecer também um desempenho superior. (RESTFulApi, s.d.)

Mais recente do que o SOAP, o REST é uma arquitetura que usa o protocolo HTTP para a sua comunicação. É usado para aceder a recursos espalhados por toda a *Web*. É constituído pelos seguintes elementos:

- *Resources* – É o primeiro elemento, que indica a que tipo de recurso vamos aceder através do protocolo.

- *Request Verbs* – São o que descrevem as ações no pedido. Funcionando à base de HTTP, temos a possibilidade de usar métodos GET, POST ou PUT.
- *Request Headers* – São elementos que atuam como informações adicionais sobre as operações a realizar juntamente com o pedido, podendo ainda definir, por exemplo, permissões de acesso.
- *Request Body* – Consoante os tipos de pedido, pode ser necessário enviar dados ao Web Service: usando o método POST, é necessário indicar os dados que são obrigatórios atualizar ou adicionar.
- *Response Body* – É o elemento que contem a resposta vinda do Web Service.
- *Response Status Codes* – São os códigos pré-definidos pelo HTTP, que vêm juntamente com a resposta do *Web Service*. Alguns exemplos são o Código 200 para uma resposta OK, ou 404 para *Not Found*.

A arquitetura REST tem algumas características especificas:

- Serve como uma aplicação cliente-servidor.
- É *Stateless*, o que significa que é o cliente que efetua o pedido ao servidor onde está o *Web Service* o responsável por enviar a informação necessária para que a ação seja efetuada com sucesso. Além disso, o servidor não tem contexto de diferentes pedidos, mesmo que sejam feitos pela mesma entidade, garantindo assim o estado *Stateless* pelo qual a arquitetura é reconhecida.
- Suporta *cache* do lado do cliente, o que ajuda a controlar o tráfego de pedidos quando a quantidade de acessos ao servidor é elevada. Sendo uma arquitetura que se diz *Stateless*, havendo um cliente a fazer dois pedidos iguais cuja respostas serão também elas iguais, faz todo o sentido a existência de cache. Havendo este mecanismo, a arquitetura garante ainda uma maior escalabilidade.
- É uma arquitetura em camadas, permitindo ainda a inserção de camadas adicionais entre o cliente e o servidor. Por exemplo, uma camada adicional de *middleware* que trata regras de negócio.
- Ao contrário do protocolo SOAP que só funciona com mensagens do tipo XML, é possível haver diferentes tipos, tais como JSON, HTML ou também XML.

Estas caraterísticas possibilitam com que a arquitetura REST possa ser utilizada em diferentes ambientes, bem como em diferentes dispositivos, sem o utilizador se ter de preocupar em construir interfaces (RESTFulApi, s.d.).

A utilização de *Web Services* facilita a comunicação entre aplicações, independentemente da tecnologia ou linguagem por elas utilizadas, garantindo assim inúmeras vantagens tais como o seu funcionamento sobre o protocolo HTTP, abrangendo a maior parte da *Web*.

Em suma, de forma a ter uma arquitetura estável, eficiente e capaz de desempenhar as funções para a qual foi desenhada, é necessário incluir diferentes tipos de tecnologias para as diferentes funcionalidades.

5.4 *Microsoft Azure*

Key Vault

O *Azure Key Vault* é um serviço *cloud* para armazenar e aceder a segredos de forma segura (Microsoft, s.d.).

Um segredo é tudo aquilo a que pretende controlar rigorosamente o acesso, como chaves de API, palavras-passe, certificados ou chaves criptográficas.

Esta solução ajuda a resolver os seguintes problemas:

- **Gestão de segredos** – Armazenar de forma segura e controlar totalmente o acesso aos *tokens,* palavras-passe, certificados, chaves de API e outros segredos.
- **Gestão de chaves** – Torna mais fácil criar e controlar as chaves de encriptação utilizadas para encriptar os seus dados.
- **Gestão de Certificados** – Permite facilmente fornecer, gerir e implementar certificados de segurança para utilização com o *Azure* e os seus recursos internos.

Esta solução pretende também centralizar os segredos das aplicações, guardar os segredos e chaves em segurança, monitorizar o acesso e utilização, administração dos segredos simplificada e também integrar com outros serviços do *Azure* (Microsoft, s.d.).

Azure DevOps

O Azure DevOps é uma plataforma que apoia uma cultura colaborativa e um conjunto de processos que reúnem programadores, gestores de projeto e colaboradores para desenvolver software (Microsoft, s.d.).

O Azure DevOps fornece funcionalidades integradas que podem ser acedidas através do browser ou IDE.

Os serviços e ferramentas que o Azure DevOps fornece são (Microsoft, s.d.):

- Boards - Fornece um conjunto de ferramentas Ágeis para apoiar o planeamento e o rastreio de trabalhos, defeitos de código e problemas usando métodos Kanban e Scrum.
- Repositórios - Fornece repositórios Git ou Team Foundation Version Control (TFVC) para controlo de origem do seu código.
- Pipelines - Fornece serviços de construção e lançamento para apoiar a integração contínua e a entrega de aplicações.
- Planos de Teste - Fornece várias ferramentas para testar aplicações, incluindo testes manuais e testes contínuos.

6 Implementação

Neste capítulo é descrito o processo de desenvolvimento do projeto ao longo do período de estágio. O projeto, no Outsystems, encontra-se dividido em 3 camadas, como ilustrado na Figura 9:

- Camada de serviços (Fidelização *Lib*) – Está dividida em dois módulos, um que funciona como uma biblioteca onde estão ações que chamam serviços do cliente para obtenção de várias variáveis necessárias ao projeto (*FID_Utils_Lib*) e outro que é onde são realizadas as integrações dos *endpoints* provenientes do *back-end* da aplicação (*FID_IS*).
- Camada intermédia (Fidelização *Core*) – Camada de lógica de negócio, onde vão ser manipulados objetos de pedido e resposta para mais fácil integração com o *back-end (FID_IS).*
- Camada *web* (Fidelização *Web*) – Camada onde estão as interfaces da aplicação e onde o utilizador vai realizar todas as ações com a aplicação (*FID_Programas*).

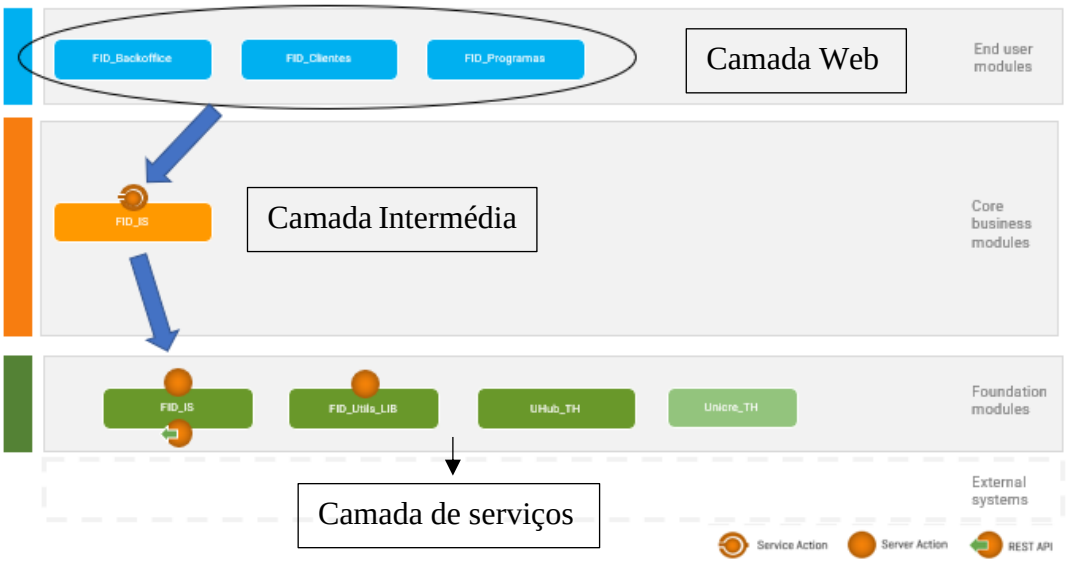


Figura 9 - Diagrama de arquitetura da aplicação em Outsystems.

(Fonte: Elaboração própria)

O desenvolvimento realizado em cada uma destas camadas é detalhado nas secções seguintes.

6.1 Camada de serviços

Nesta secção é explicada a fase inicial do desenvolvimento deste projeto, que será a implementação dos serviços e de tudo o que vai ser preciso para o *Outsystems* consumir corretamente e de uma maneira dinâmica os serviços desenvolvidos pelo *Back-End.*

Numa fase inicial do desenvolvimento deste projeto e de acordo com uma arquitetura definida pelo cliente, foi preciso dividir e camada de serviços em dois módulos.

Um deles é uma biblioteca onde estão alojadas ações que chamam serviços do cliente para obtenção de variáveis (*FID_Utils_LIB*).

E o outro onde são realizadas as integrações com os *endpoints* expostos pelo *back-end (FID_IS).*

É dentro destes módulos que serão implementadas todas as ferramentas necessárias de acordo com a arquitetura interna do cliente, para a integração com os serviços providos pelo *Back-End* da aplicação.

Módulo *FID_Utils_LIB*

Este módulo da aplicação é uma biblioteca onde estão alojadas três *Server Actions* que irão buscar variáveis, perfis de acesso e o URL da API.

As *Server Actions* são essencialmente ações que correm lógica do lado do servidor, que vão ser utilizadas mais tarde nos restantes módulos.

Este módulo funciona essencialmente como uma biblioteca de dados que vêm de sistemas internos do cliente, como chaves, perfis de acesso e URL's. Na Figura 10, é ilustrada a organização das *Server Actions* dentro do módulo.

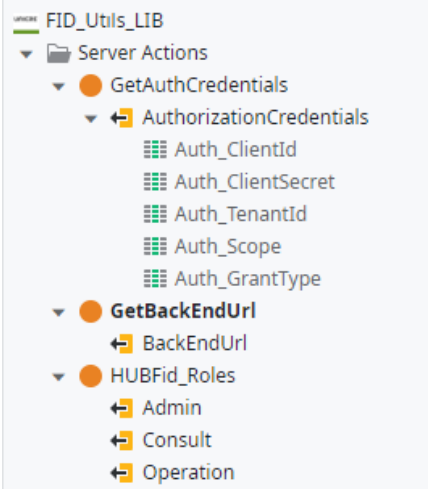


Figura 10 – 3 Server Actions no módulo FID_Utils_LIB

(Fonte: Elaboração própria)

A *Server Action, "GetAuthCredentials",* é onde a aplicação vai buscar as credenciais necessárias ao *Key Vault*, através de ferramentas provenientes da fábrica Outsystems do cliente, que são o identificador da aplicação (*Client_Id*), e um segredo que é conhecido apenas pela aplicação e pelo servidor de autorização, que funciona como uma palavra-passe quando vai buscar o *token* de acesso ao *back-end.*

E irá também concatenar as variáveis *Tenant_Id,* que é um identificador da empresa cliente dentro do *Azure, e o Grant_Type* que é um identificador do processo que vai ser realizado quando se obtém o *token* de acesso.

Esta variáveis referidas no parágrafo anterior estão armazenadas em *Site Properties* porque irão ser alteradas de forma dinâmica entre ambientes, e irão ser concatenadas juntamente com as credenciais referidas anteriormente num objeto de saída da ação que irá ser consumido noutro módulo para fazer a autenticação nas chamadas do *Back-End.* Todo o código da *Server Action* pode ser consultado nos anexos e o seu objeto de saída na Figura 11.

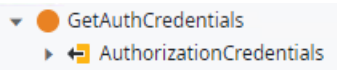


Figura 11 - Server Action "GetAuthCredentials" com o seu objeto de saída.

(Fonte: Elaboração própria)

A seguinte *Server Action*, "*GetBackEndUrl*", é onde vai ser devolvido o objeto com o URL que está também guardado em *Site Properties*, para poder ser alterado de forma dinâmica sempre que a aplicação for passada entre ambientes. Todo o código da *Server Action* pode ser consultado nos anexos e o seu objeto de saída na Figura 12.

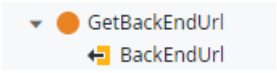


Figura 12 -Server Action "GetBackEndUrl" com o seu objeto de saída.

(Fonte: elaboração própria)

E por fim, a *Server Action "HUBFid_Roles",* é onde a aplicação vai executar uma série de ações para determinar se o *User* que está a utilizar a aplicação pertence a um determinado *Role*, devolvendo assim uma série de booleanos de acordo com o *Role* do utilizador. Todo o código da *Server Action* pode ser consultado nos anexos e os seus objetos de saída na Figura 13.

Figura 13 - Server Action "HUBFid_Roles" com os seus objetos de saída.

(Fonte: elaboração própria)

No subcapítulo seguinte será explicado como todas estas ações serão implementadas na camada de integração de serviços exteriores à plataforma *Outsystems.*

Módulo *FID_IS*

Este módulo, como foi explicado no final do subcapítulo anterior, é onde estarão alojadas todas as integrações com serviços exteriores ao *Outsystems*, sendo estes essencialmente a API e os métodos provenientes pelo *Back-End* da aplicação.

Esta parte de integração, está dividida essencialmente em três partes, sendo estas "HubFID_Programas", "HubFID_Clientes" e "HubFID_Dashboard", sendo que a única que foi desenvolvida dentro do contexto deste relatório de estágio, foi a parte de programas. Pode ser consultada essa divisão, na Figura 14.

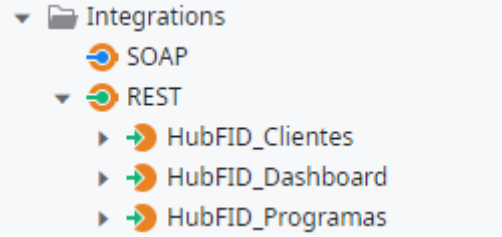


Figura 14 - As 3 áreas de integração com sistemas externos

(Fonte: Elaboração própria)

Como do lado do *Back-End* da aplicação, a API foi desenvolvida em arquitetura REST tivemos também de integrar os métodos na mesma arquitetura apesar do *Outsystems* disponibilizar também integrações com SOAP, SAP e outras arquiteturas.

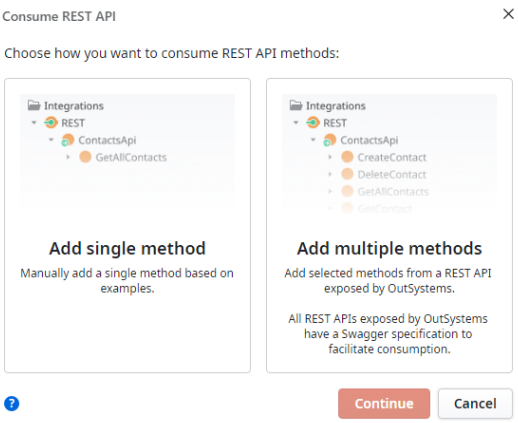


Figura 15 - Menu de Integração de API's com arquitetura REST.

(Fonte: Elaboração própria)

Primeiramente, para fazer uma integração com uma API com arquitetura REST o *Outsystems* disponibiliza duas opções, como demonstrado na Figura 15:

- Adicionar os métodos individualmente, que foi feito nesta aplicação porque o desenvolvimento e as integrações com o *Outsystems* foram feitos em paralelo com o *Back-End*,
- Possibilidade carregar um *Swagger* e escolher um ou todos os métodos disponíveis no mesmo para integrar de uma vez no IDE.

Para integrar um método individual no *Outsystems*, tem de ser feita uma série de passos, que começa por olhar para a documentação da API e perceber qual é o URL base, os requisitos de segurança e autenticação, e a definição dos métodos como o método HTTP, *path* do URL, formato de pedido e de resposta, como ilustrado na Figura 16.

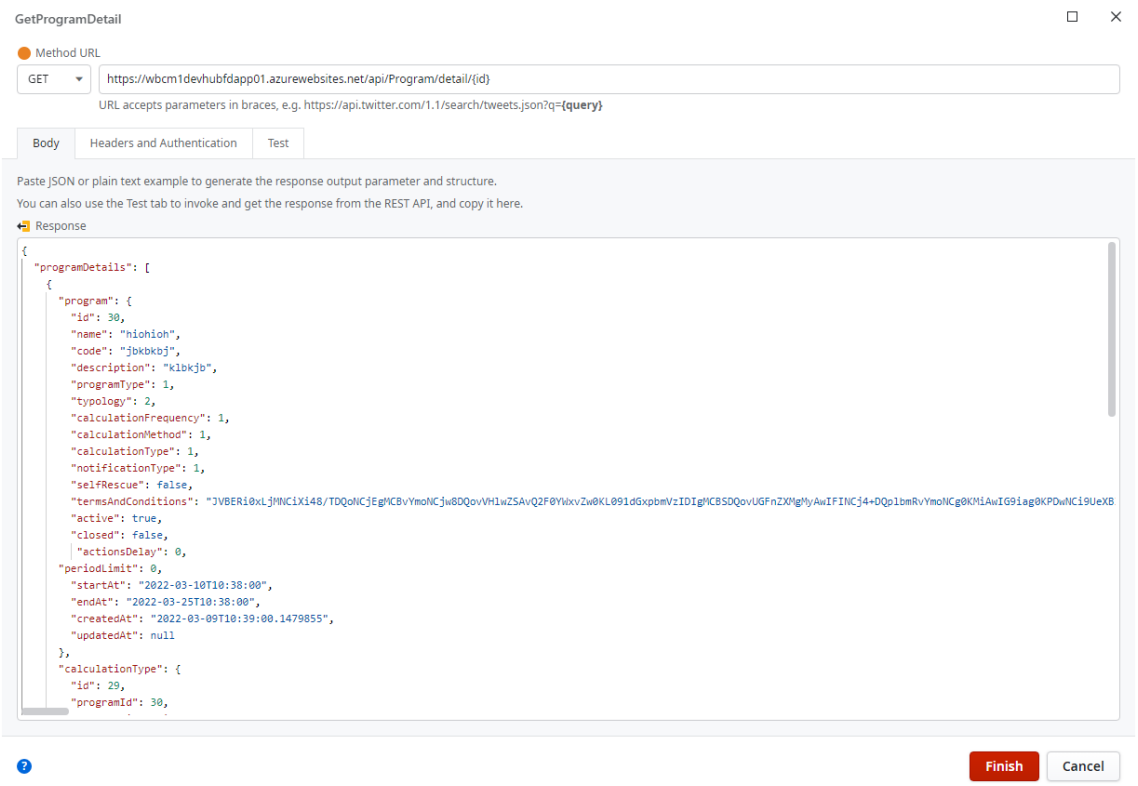


Figura 16 - Janela de integração de uma Web API.

(Fonte: Elaboração própria)

Depois de feita essa análise, abre-se então a janela da Figura 12, seleciona-se o método do pedido, no canto superior esquerdo, que pode ser GET, POST, PUT, DELETE e PATCH.

Em seguida, coloca-se o URL do método da chamada que se quer fazer, à direita. E no caso de ser uma chamada a um método GET, simplesmente tem de se colocar o exemplo da resposta em formato *Json.*

Caso seja uma chamada de qualquer um dos outros tipos de métodos HTTP, idealmente terá de se colocar um exemplo da resposta e um exemplo do pedido, também em formato *Json.*

Depois da chamada estar criada é necessário criar uma ação chamada "*OnBeforeRequest*", que vai ficar dentro de cada uma das 3 principais áreas referidas anteriormente, para ser chamado antes de cada pedido à aplicação, como se ilustra na Figura 17.~

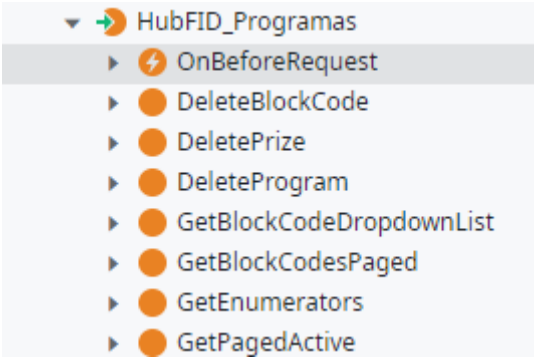


Figura 17 – Chamada REST "HubFID_Programas" com OnBeforeRequest e respetivos endpoints

(Fonte: Elaboração própria)

Dentro deste é onde vai estar presente toda a lógica relativamente ao que o pedido precisa, assim como o URL base e os *Headers* a enviar no pedido, que neste caso serão:

- ***Token* que vai autenticar o pedido no Back-End:** Para obter este *token* é preciso utilizar outra ferramenta, proveniente da fábrica de *Outsystems* do cliente, que com as informações que foram exportadas do módulo "*FID_Utils_LIB*" pela função "*GetAuthCredentials()*", que vai fazer um pedido ao *Microsoft Azure*, no domínio da aplicação, e assim gerar o *token* que será enviado na chamada, como ilustrado na Figura 18.

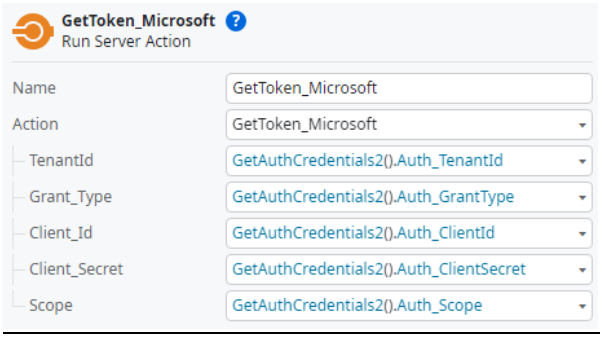


Figura 18 - Ação "GetToken_Microsoft" que gera o token.

(Fonte: Elaboração própria)

- ***UserId:*** Para obter esta informação, preciso recorrer a duas funções em que uma é a função "*GetUserId()*", que vai buscar o *UserId* do utilizador que está de momento a utilizar a aplicação e outra "*GetUser(UserId).User.Username*" que vai buscar à tabela nativa do *Outsystems* de *Users*, o nome do utilizador que fez o pedido, como se vê na Figura 19. E assim, a aplicação ter um controlo maior de quem consultou o quê dentro da mesma.

Figura 19 - Função concatenada que gera o nome de utilizador.

(Fonte: Elaboração própria)

Para se obter o URL base da chamada, é utilizada a função descrita anteriormente "GetBackEndUrl", que vai às *Site Properties* do módulo "*FID_Utils_LIB*" obter o mesmo.

O código do "*OnBeforeRequest*", irá ser executado antes de cada chamada, para assim todas as chamadas serem iguais e todas preencherem os mesmos requisitos para se puder fazer pedidos ao *Back-End* sem problemas. Todo o código do "*OnBeforeRequest"* explicado anteriormente pode ser consultado em anexo.

Após a importação dos serviços pela plataforma do *OutSystems*, esta gera automaticamente todas as estruturas definidas pelo *Back-End*.

É possível então usar estas estruturas para mapear os inputs necessários que vêm das camadas superiores e após a execução do serviço fazer o mesmo para o output.

6.2 Camada intermédia

Depois de implementar a camada de serviços, é preciso montar uma camada intermédia também dentro do módulo (FID_IS).

Esta vai permitir implementar uma série de lógica de negócio e permitir também manipular as respostas e os pedidos das chamadas que fazemos à API de uma forma mais simples e estruturada. Para toda essa lógica não estar implementada na camada de interface.

Módulo FID_IS (continuação)

Como foi explicado anteriormente, a cama intermédia vai ficar dentro de este módulo e vai permitir através de *Service Actions*, criar toda a lógica envolvida para se exportar os pedidos corretamente para os módulos de Web.

Uma *Service Action* é uma chamada remota baseada em REST, mas tem um uso muito parecido às *Server Action*. Quando expomos uma *Service Action* gera uma dependência fraca entre o módulo consumidor e o módulo produtor.

Se a aplicação for relativamente complexa, isto é, bastante lógica no tratamento de inputs e outputs, é aconselhável que esse desenvolvimento seja realizado num módulo à parte de modo a deixar os módulos de serviços independentes da lógica.

Nesta aplicação em concreto, visto que são feitos pedidos com alguma duração e alguns um pouco grandes em tamanho, decidiu-se deixar tudo no mesmo módulo para uma forma mais simples de gerir os pedidos que são realizados à API.

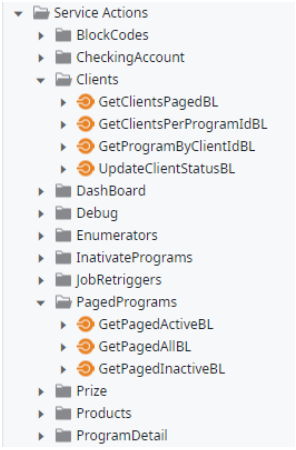


Figura 20 - Exemplo de algumas Service Actions divididas por pastas.

(Fonte: Elaboração própria)

Foram criadas várias *Service Actions*, como é demonstrado na Figura 20, para expor para os módulos de Web para assim serem feitos vários tipos de requisições dos mais diferentes tipos, seja eles pedidos de listagem, criações ou editar dados na base de dados.

Na Figura 21, é apresentado apenas um encapsulamento de um serviço em que a lógica necessitou de funções auxiliares, para lidar com o tratamento de inputs.

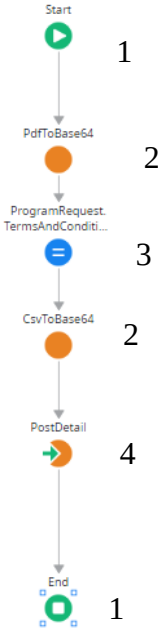


Figura 21 - Implementação de um serviço na cama intermédia na plataforma Outsystems.

(Fonte: Elaboração própria)

Legenda:

1. Nós de entrada e saída, indicam o começo e o fim, respetivamente, de cada ação.
2. Função auxiliar criada, pela fábrica de *Outsystems* do cliente, para converter ficheiros carregados na interface em *binary data* para Base64, ou seja, em formato *string*.
3. Mapeia a *string* criada na primeira função auxiliar numa estrutura de dados.
4. Chamada ao serviço previamente criado e respetivo mapeamento para as variáveis de *input.*

6.3 Camada *Web*

Esta seção foi dividida em 6 partes, uma para cada interface, e correspondem às funcionalidades descritas no capítulo de objetivos. No início de cada secção, é apresentada uma imagem do aspeto final do ecrã de acordo com as funcionalidades referidas na análise de requisitos e seguido das respetivas implementações e decisões.

Todos os dados que aparecem nos ecrãs são de ambientes de desenvolvimento. Para o aspeto visual dos ecrãs foram utilizados módulos Tema do cliente, desenvolvidos anteriormente, para garantir homogeneidade entre as aplicações produzidas.

Lista de Programas

Todos os programas visualizados neste ecrã foram carregados através do *back-end*. Desta forma não é necessário inserir muita lógica do lado do *front-end*, mas sim enviar pedidos GET à API para que seja possível a obtenção dos dados necessários, de seguida estes são carregados em forma de tabela tal como mostra a figura 22.

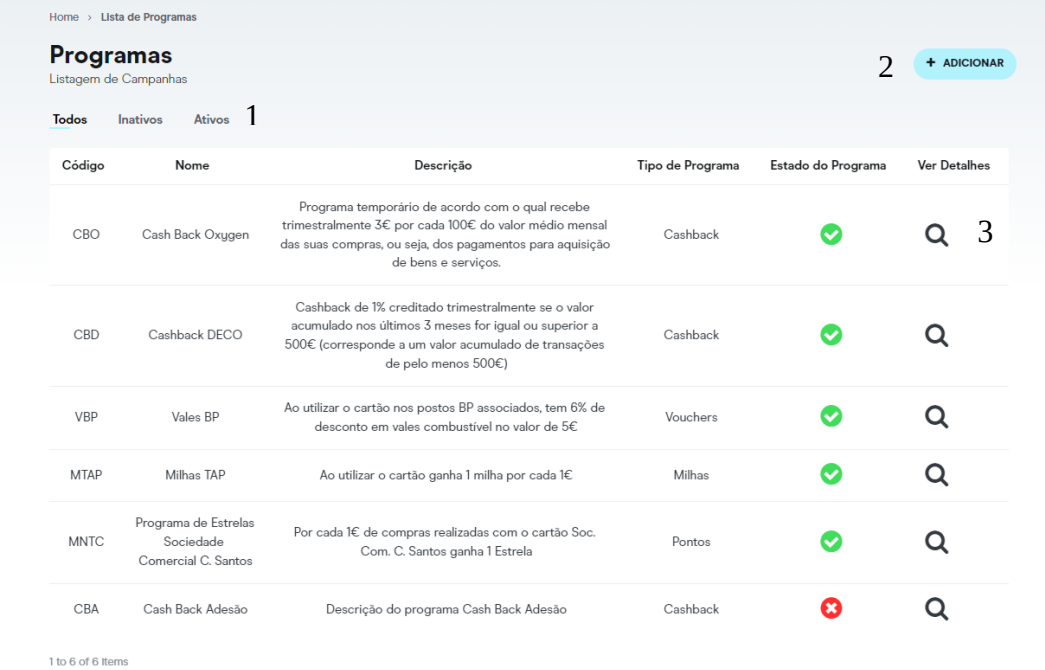


Figura 22 - Interface de Lista de Programas

(Fonte: Elaboração própria)

Nestes pedidos são enviados parâmetros de forma permitir a filtragem, ordenação e paginação dos dados, para que haja um menor consumo de dados permitindo assim uma melhor performance. Os parâmetros são:

- Filtro, ou seja, permite o utilizador filtrar o estado do programa (Todos, Ativos e Inativos), através das abas assinaladas com o número 1.
- Página atual em que o utilizador se encontra na tabela.
- Número de elementos apresentados por página.
- Campo que o utilizador deseja que a tabela seja ordenada, por defeito é o campo de data de criação.
- Tipo de ordenação que o utilizador deseja, por defeito descendente.

O botão assinalado com o número 2, permite ao utilizador navegar para uma página de criação de programa.

E no botão assinalado com o número 3, permite ao utilizador navegar para um ecrã com o detalhe do programa.

Criação de um programa

Um dos fluxos implementados foi a criação de um programa, possibilitando assim ao administrador criar programas conforme seja necessário. Esta funcionalidade é permitida ao utilizador ao pressionar o botão assinalado com o número 2 na Figura 20, e de seguida é apresentado um ecrã com o formulário necessário para a criação de um programa como ilustra a Figura 23.

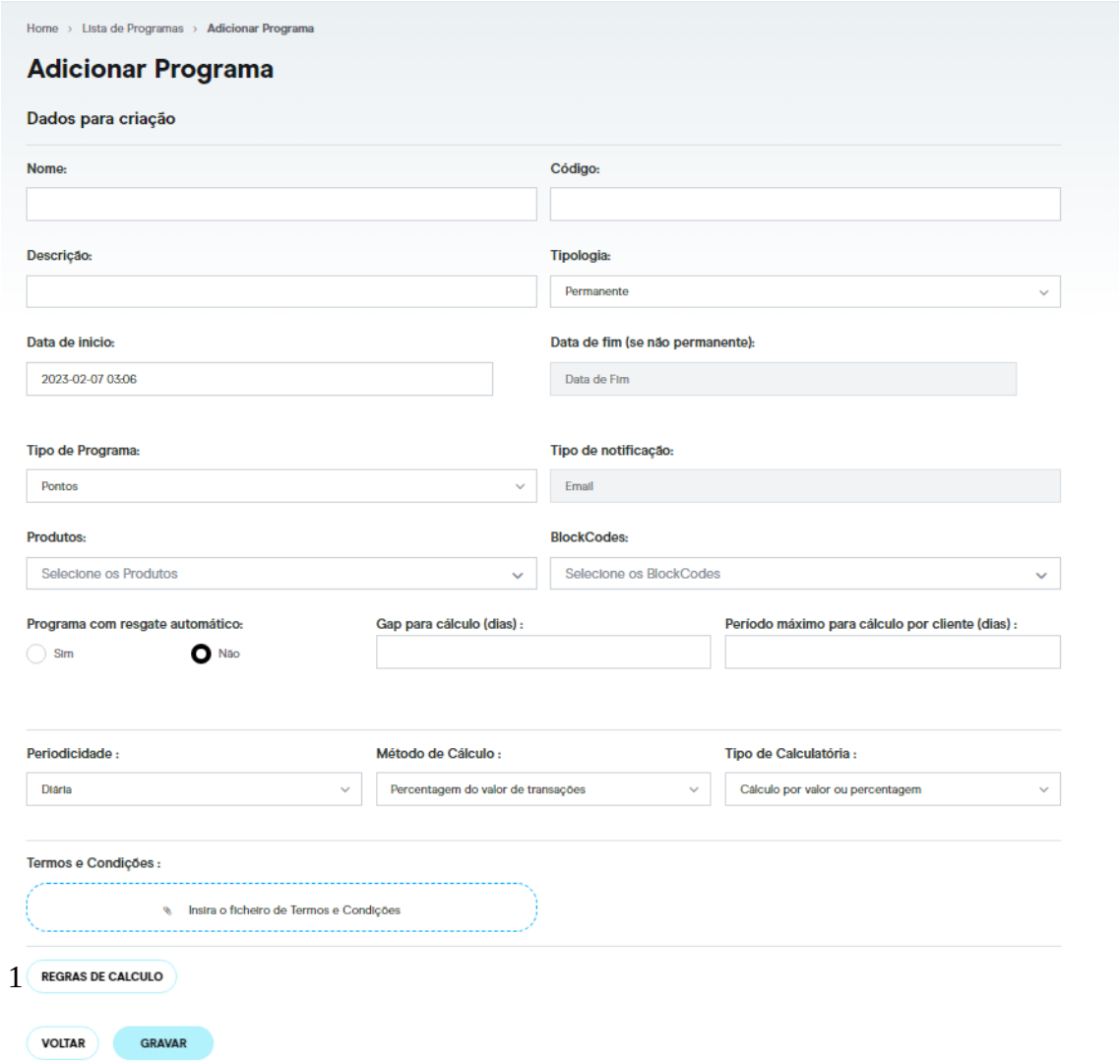


Figura 23 - Interface de Criação de Programa

(Fonte: Elaboração própria)

Após o preenchimento deste formulário o utilizar tem de selecionar o botão “Regras de Cálculo”, para o utilizador navegar para o ecrã das regras de cálculo como mostrado na figura 24.

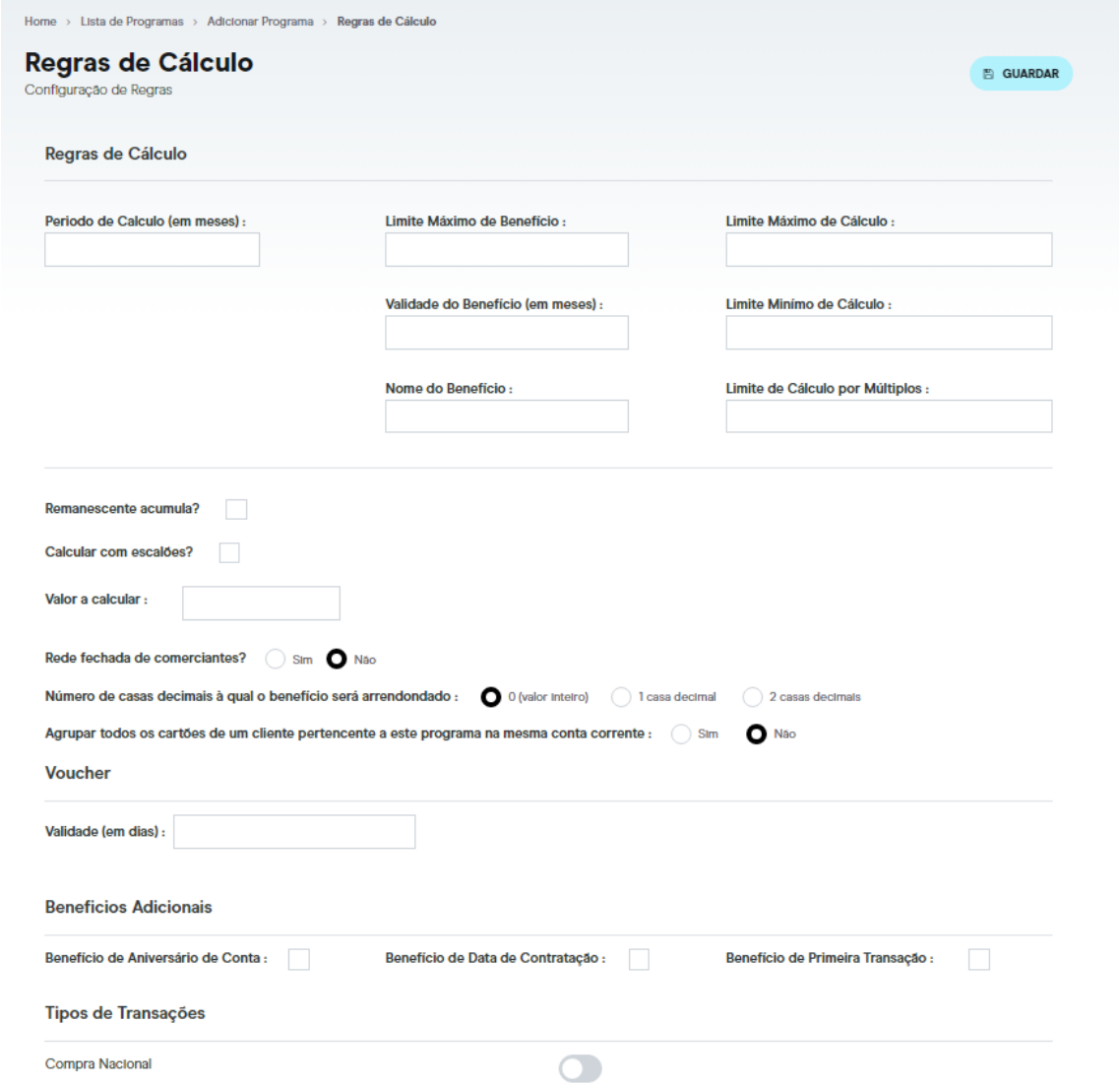


Figura 24 - Interface de Regras de Cálculo

(Fonte: Elaboração própria)

Após o preenchimento deste formulário, o utilizador seleciona o botão “Guardar” e é reencaminhado novamente para o ecrã anterior, onde assim poderá guardar o programa como um todo na base dados.

Consulta/Edição de um programa

Outro dos fluxos implementados foi a consulta do detalhe de um programa e respetiva edição. Como indicado na figura 25, ao carregar na opção de detalhes de um programa (número 3, Figura 22) é aberta uma nova página possibilitando assim a edição e a visualização de um programa já parametrizado.

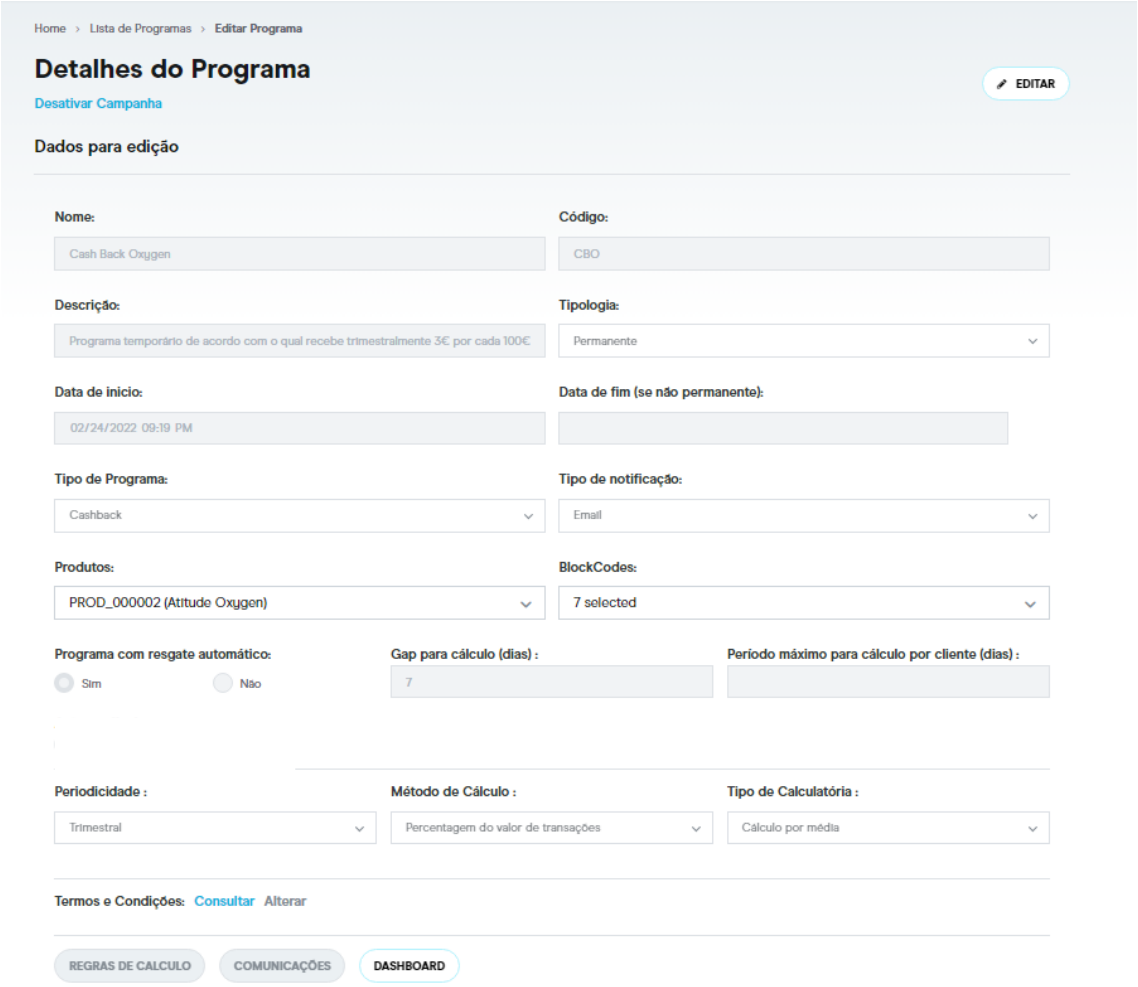


Figura 25 - Interface de consulta de detalhe de um programa

(Fonte: Elaboração própria)

Ao selecionar a opção “Editar”, é disponibilizada a edição dos campos da aplicação e em que o fluxo funciona como a criação de programa, só que os campos já estarão preenchidos.

Eliminar um programa

De modo a completar as operações CRUD, foi implementada a possibilidade de eliminar programas. Para isto ser possível é preciso que o programa em causa esteja criado mas ainda não tenha inicializado a sua execução.

Esta opção é disponibilizada no ecrã de lista de programa, que caso alguns dos programas apresentados ainda não tenha começado a sua execução, aparece um ícone para assim possibilitar a eliminação do mesmo.

Inativar um programa

Para inativar um programa, o mesmo precisa já de se encontrar em execução. Esta funcionalidade é acedida pelo ecrã de consulta de detalhe do programa.

A aplicação caso ainda não seja possível inativar o programa pretendido, imprime uma mensagem com uma data em que o utilizador poderá assim inativar, como ilustrado na Figura 26.

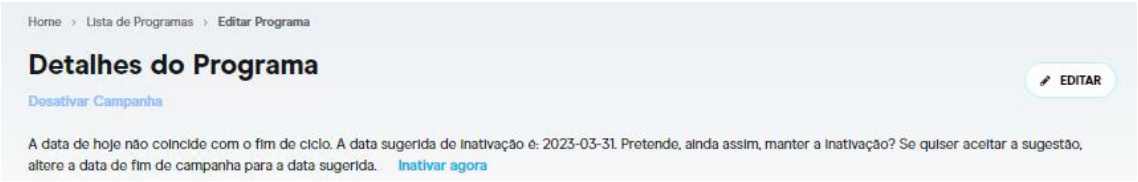


Figura 26 – Mensagem de aviso de inativação do programa.

(Fonte: Elaboração própria)

Ainda é disponibilizado o botão de inativação imediata, com o risco de o utilizador comprometer os dados e a execução do programa a inativar.

7 Verificação e validação

Um processo que deve ser obrigatório no desenvolvimento de *software*, qualquer que tenha sido a tecnologia utilizada, é a execução de testes.

Como neste projeto não estava previsto a realização de testes por uma equipa externa, os testes foram realizados internamente e divididos em duas fases:

- Numa primeira fase as interfaces foram testadas em ambiente de desenvolvimento pela equipa de desenvolvimento.
- Numa segunda fase passava a ambiente de qualidade e assim eram testadas pelo *Product Owner* do projeto.

Sempre que era encontrado um *bug* ou algum mau funcionamento nos testes à aplicação o *feedback* era dado nas reuniões diárias.

Consequentemente era também aberto um *bug*, ou seja, uma tarefa com a descrição desse mau funcionamento numa plataforma interna (*Azure DevOps*), que ficava associado à pessoa que desenvolveu a essa funcionalidade.

Após um elemento da equipa resolver essa anomalia ou mau funcionamento, o *Product Owner* voltava a testar e após um feedback positivo, era validada essa funcionalidade analisada.

Devido à falta de alguns dados em ambiente de qualidade, a aplicação foi também testada num ambiente controlado de produção, pelo que nesse ambiente a aplicação trabalhava com clientes e contas reais, mas sem nunca lhes atribuir ou retirar valores monetários, apenas trabalhar em cima de transações feitas pelos mesmos em ambiente real.

De seguida são explicadas as implementações de algumas das verificações realizadas durante o desenvolvimento do Hub de Fidelização.

Em termos de verificação, nesta aplicação foram feitas várias verificações, essencialmente, na criação e edição de programas.

Estas verificações consistiam em verificar se os formatos dos campos tinham de ser os expectáveis pelo serviço.

Também não será possível criar programas com uma data passada nem com a data de início maior que a data de fim.

Também se teve de verificar se ao criar um programa de fidelização, a data de início de execução da campanha de fidelização não era passada, e se a data de fim de programa não era inferior à data de início, devolvendo assim a mensagem ilustrada na Figura 27.

Figura 27 - Mensagem de erro

(Fonte: Elaboração própria)

Verificou-se também se todos os campos de parametrização obrigatórios de um programa se encontram preenchidos, sendo que a aplicação tem de validar isso e não deixar seguir em frente com os fluxos se os campos estiverem vazios, como se mostra na Figura 28.

Figura 28 - Erro devido a campos vazios.

(Fonte: Elaboração própria)

Ao eliminar programas, e de modo a evitar que o utilizador eliminasse dados indesejado, foi feita a verificação se o mesmo já estava em execução ou não.

Se o programa já estivesse ativo, apenas era dada a possibilidade de inativar o mesmo e a aplicação por defeito não apresenta o ícone de eliminar programa.

Para inativar um programa é preciso que a data de inativação do mesmo esteja em conformidade com o final de um mês (caso o cálculo do programa seja do tipo mensal) ou no final de um trimestre (caso seja trimestral), como pode ser visto na Figura 24.

8 Conclusões

Apesar deste estágio ter sido realizado numa época de pandemia e em teletrabalho, um dos aspetos mais positivos foi ter sido a possibilidade de aprender a utilizar uma tecnologia nova e que está em constante ascensão, mais concretamente o OutSystems. E outro foi a possibilidade de identificar um problema e criar uma solução para o mesmo.

E tudo isto num ambiente empresarial e com um cliente externo, o que me possibilitou ganhar bastantes capacidades, tanto a nível de tecnologias, mas principalmente a nível de comunicação.

Quanto ao projeto desenvolvido, inicialmente foram encontradas algumas dificuldades devido à minha inexperiência a nível empresarial e de comunicação, mas também em perceber o que era realmente pedido pelo cliente. No entanto, após alguma habituação e estudo de documentos como análise funcional, foi possível uma maior compreensão sobre o objetivo do projeto.

A metodologia utilizada adaptou-se perfeitamente à realidade do projeto e à equipa de trabalho. As reuniões diárias revelaram-se importantes pois foram uma forma de controlar o trabalho realizado.

Em relação às ferramentas e tecnologias utilizadas no desenvolvimento da aplicação, encontrei algumas dificuldades porque ainda não me encontrava familiarizado com o *Outsystems*, no entanto foi realizada uma pesquisa da mesma e um estudo aprofundado e assim permitiu também com o tempo melhor as minhas capacidades a nível de desenvolvimento, garantindo assim uma melhor performance e usabilidade do *Hub*.

Devido à metodologia ágil utilizada, foi implementada a abordagem de programar e testar logo em seguida, dando assim a capacidade de encontrar e corrigir erros de uma forma rápida e eficaz.

O problema prendia-se com a necessidade da instituição de crédito, cliente da Armis, precisar de uma plataforma de gestão de campanhas de fidelização.

Essa necessidade advém de que a atual plataforma existente estava desatualizada e era gerida e mantida por terceiros. Por ser gerida e mantida por terceiros, a plataforma tornou-se muito dispendiosa e de difícil manutenção.

Foi então desenvolvida uma plataforma web utilizando a tecnologia acima referida, capaz de gerir plataformas de fidelização a clientes, e realizar as funcionalidades referidas no capítulo Análise de Requisitos.

A plataforma web desenvolvida cumpre com todos os objetivos iniciais estabelecidos, e como tal encontra-se preparada para o utilizador criar, editar, inativar e eliminar um programa de fidelização de clientes, ou seja, efetuar a gestão à volta do mesmo.

Todas as funcionalidades foram testadas e validadas pelo *Product Owner*, visto que se pode deduzir que foram realizadas com sucesso.

Todos os testes foram feitos de uma forma incremental, sendo estes realizados ao longo do desenvolvimento de cada página, garantindo sempre que a plataforma estava de acordo com os objetivos.

Os objetivos deste estágio foram cumpridos com sucesso, e foi entregue uma plataforma capaz de realizar a gestão relativa a programas de fidelização de acordo com todas as funcionalidades inicialmente levantadas, validadas então pelo *Product Owner*.

8.1 Trabalho Futuro

Dado que o estágio teve apenas a duração de 4 meses, não tornou possível o desenvolvimento de uma aplicação completa devido à complexidade da mesma.

Ficam ainda funcionalidades que foram desenvolvidas no futuro, sendo algumas destas a criação de um *dashboard* para uma visualização geral dos *logs* do motor da aplicação, o mapeamento de novos *roles* em todas as interfaces da plataforma, a criação de um catálogo de prémios utilizados em algumas das campanhas previamente parametrizadas e também interfaces de consulta a clientes, entre outras.

No final deste estágio, a aplicação continuou a ser desenvolvida pela equipa, onde me inseri em contexto laboral.

Bibliografia

AnnexCloud. (n.d.). *A Comprehensive Guide To Bank Loyalty Programs : Best Features & Examples Included*. Retrieved Janeiro 21, 2023, from AnnexCloud: https://www.annexcloud.com/blog/comprehensive-guide-bank-loyalty-programs-best-features-examples-included/

ARMIS. (n.d.). *NÓS SOMOS ARMIS*. Retrieved Janeiro 3, 2022, from ARMIS: https://www.armis.pt/

Corda, A. F. (2019). *Desenvolvimento de Solução Outsystems para Instituição* . Lisboa.

Forrester. (n.d.). *New Forrester Wave Recognizes OutSystems as the Developers' Choice*. Retrieved Dezembro 15, 2022, from Outsystems: https://www.outsystems.com/1/low-code-development-platforms-wave-/

HupData Data Analysis Solutions. (n.d.). *Plataforma Low Code – O que é? Quais as vantagens?* Retrieved Fevereiro 05, 2023, from HupData.

InboundCycle. (2023, Fevereiro 1). *O que é fidelização de clientes: vantagens e estratégias para alcançá-la*. Retrieved from InboundCycle: https://www.inboundcycle.com/pt/dicionario-marketing-online/fidelizacao-de-clientes

Loyty. (n.d.). *A Loyty*. Retrieved Janeiro 21, 2023, from https://loyty.pt/sobre-software-fidelizacao

Microsoft. (n.d.). *Conceitos básicos do Azure Key Vault*. Retrieved Janeiro 10, 2023, from Microsoft: https://learn.microsoft.com/pt-pt/azure/key-vault/general/basic-concepts

Microsoft. (n.d.). *O que é o DevOps?* Retrieved Janeiro 29, 2023, from Microsoft: https://azure.microsoft.com/pt-pt/resources/cloud-computing-dictionary/what-is-devops/

Outsystems. (2023, Janeiro 22). Retrieved from Outsystems: www.outsystems.com

Outvio. (n.d.). *6 tipos de programas de fidelização ideais para o teu negócio*. Retrieved Janeiro 05, 2023, from Outvio: https://outvio.com/pt/blog/programas-de-fidelizacao/#:~:text=Vantagens%20dos%20Programas%20de%20Fideliza%C3

%A7%C3%A3o%20de%20Clientes%20Permitem,cliente%3B%20Aumentam%20os%20n%C3%ADveis%20de%20satisfa%C3%A7%C3%A3o%20do%20cliente.

Pis Marketing. (n.d.). *Programas de Fidelização de Clientes: Exemplos e Vantagens*. Retrieved Janeiro 25, 2023, from Pis Marketing: https://pismarketing.pt/programas-de-fidelizacao-de-clientes-exemplos-e-vantagens/

ReferralRock. (n.d.). *16 Customer Loyalty Program Software Options for Your Business*. Retrieved Janeiro 05, 2023, from Referral Rock: https://referralrock.com/blog/customer-loyalty-program-software/

RESTFulApi. (n.d.). *What is REST*. Retrieved Dezembro 26, 2022, from REST API Tutorial: https://restfulapi.net/

Scrum Portugal. (n.d.). *Metodologia Scrum*. Retrieved Dezembro 22, 2022, from Scrum Portugal: https://www.scrumportugal.pt

ScrumPortugal. (n.d.). *Metodologia Agile*. Retrieved Dezembro 22, 2022, from Scrum Portugal: www.scrumportugal.pt

Silva, R. (2022, Julho 6). *A evolução das plataformas de desenvolvimento low code*. Retrieved from Computer World: https://www.computerworld.com.pt/2022/07/06/a-evolucao-das-plataformas-de-desenvolvimento-low-code/

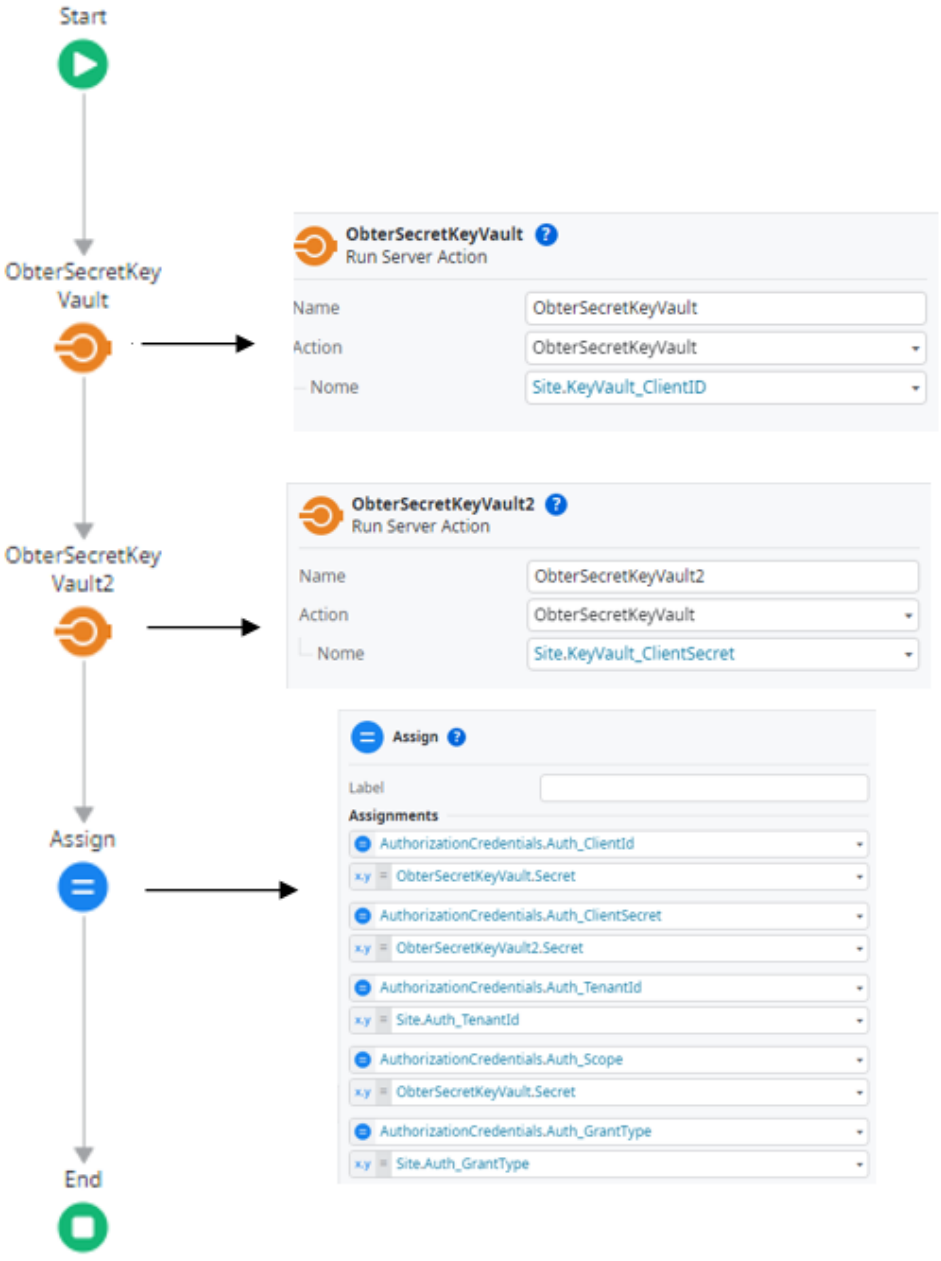
Sogec. (n.d.). *Quem somos*. Retrieved Janeiro 21, 2023, from Sogec: https://www.sogec.pt/

w3. (2022, Janeiro 23). *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. Retrieved from W3: https://www.w3.org/TR/wsdl/

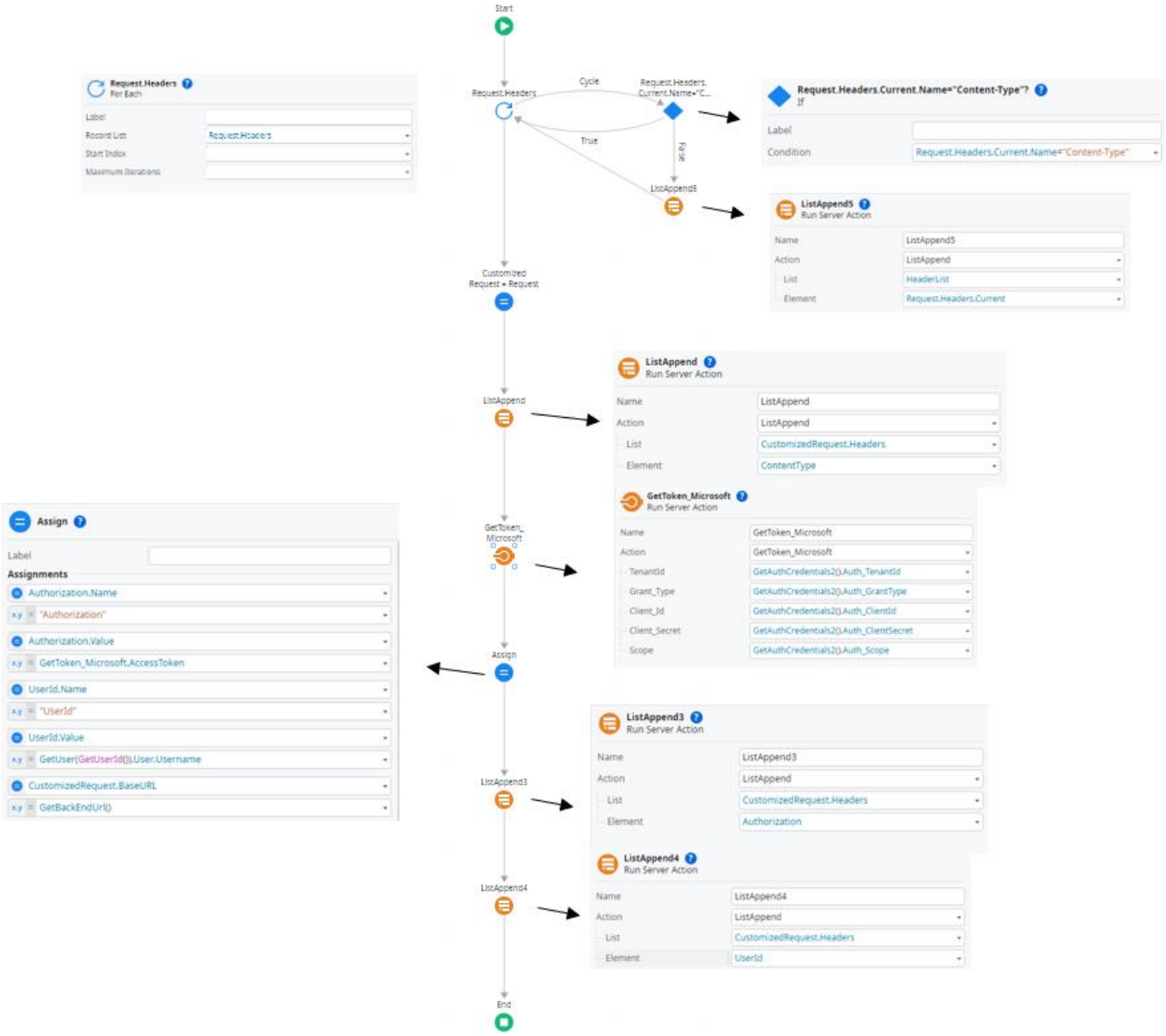
Anexos

A 1. Código

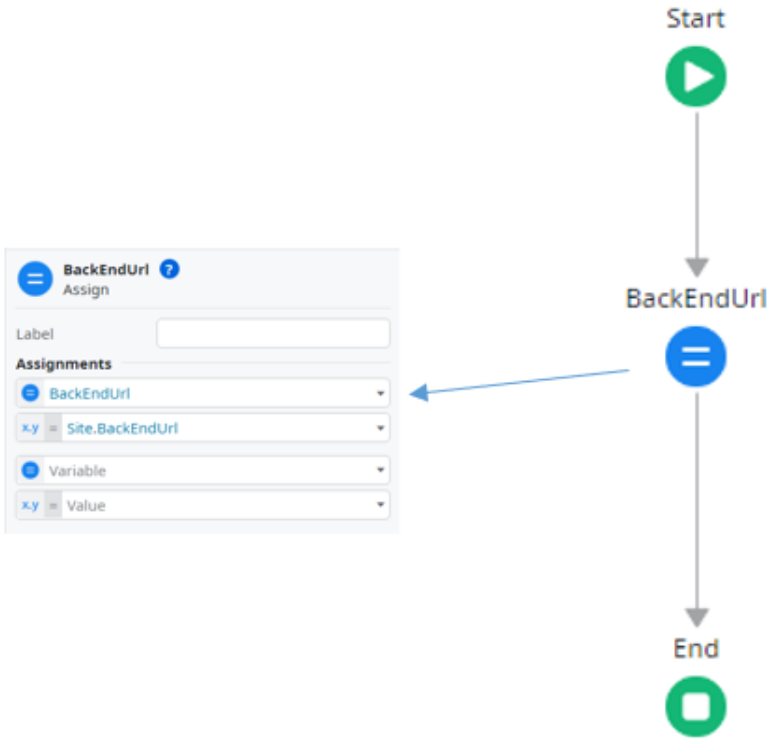
Ação GetAuthCredentials



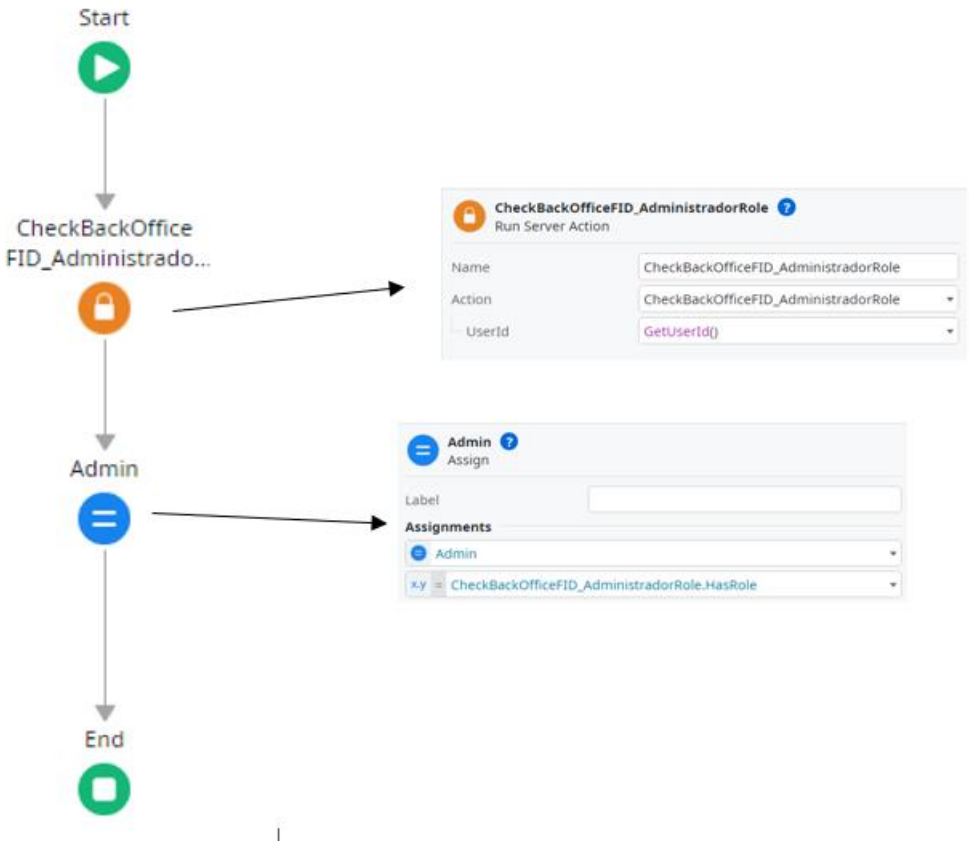
Ação *OnBeforeRequest*



Ação GetBackEndUrl()

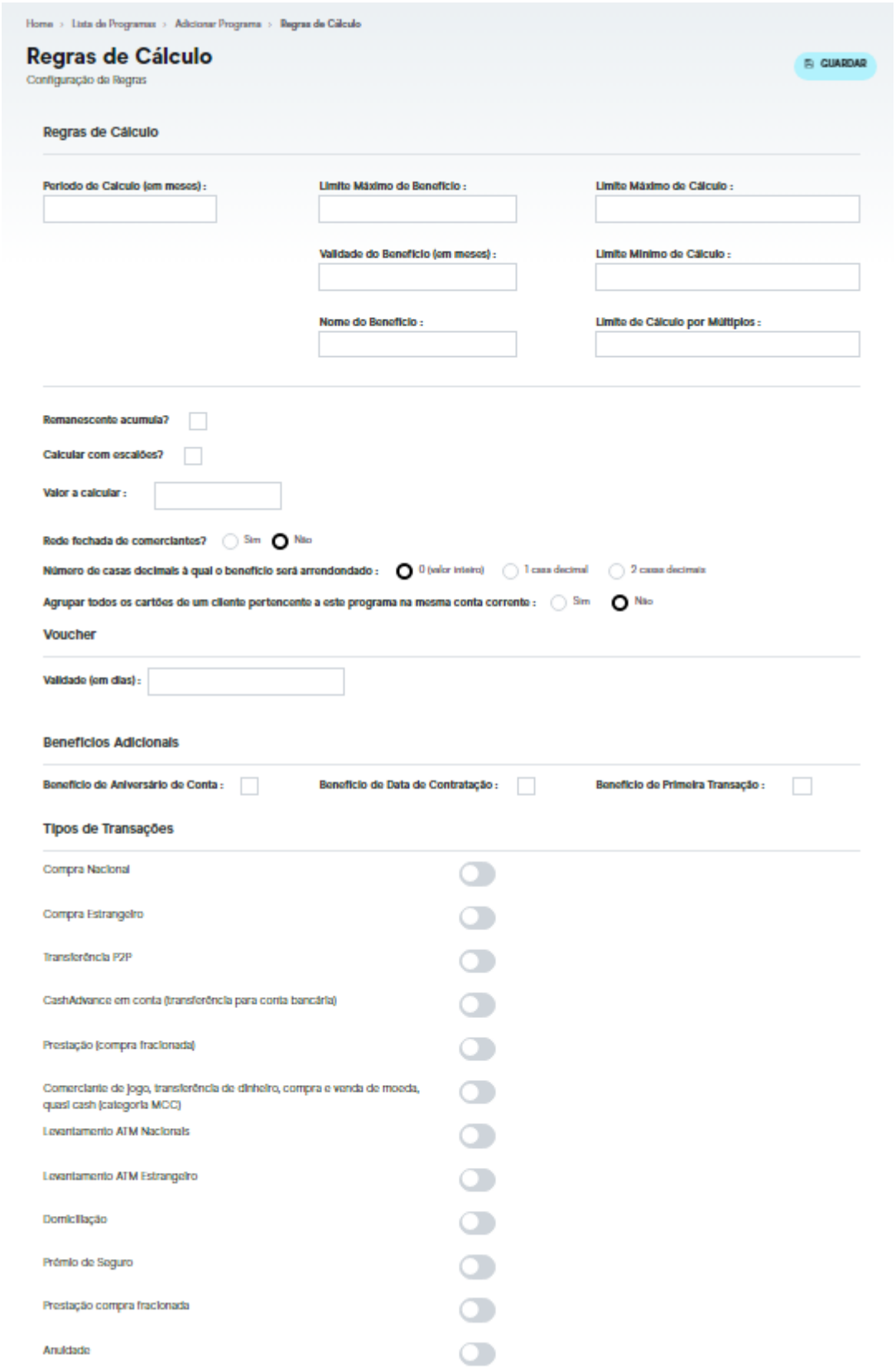


Ação HUBFid_Roles



A 2. Interface

Ecrã Regras de Cálculo



A 3. Descrição dos diagramas de casos de uso

Descrição do Caso de Uso "Eliminar um Programa"

Nome	Eliminar um programa.
Descrição	O objetivo é o ator poder eliminar uma campanha de fidelização.
Pré-Condição	1. O ator tem de efetuar um login válido. 2. A campanha de fidelização a eliminar, tem de ter a data de início de campanha maior que a data atual.
Caminho Principal	1. O ator encontra-se no ecrã "Lista de Programas". 2. O ator seleciona o ícone "*trashcan*" que se encontra na coluna mais à direita da lista. 3. O sistema apresenta uma janela *pop-up* com uma confirmação de eliminar o programa. 4. O ator seleciona o botão "Apagar". 5. O sistema imprime uma mensagem de sucesso, em como o programa foi eliminado.
Caminhos Alternativos	O ator cancela a janela *pop-up* apresentada pelo sistema.
Suplementos ou adornos	N/A
Pós-Condição	O sistema apaga a campanha de fidelização da base de dados.

Descrição do Caso de Uso "Importar um ficheiro *CSV* de clientes no programa"

Nome	Importar um ficheiro *CSV* de clientes no programa.
Descrição	O objetivo é o ator poder inserir um ficheiro *CSV* numa campanha de fidelização já ativa.
Pré-Condição	- O ator tem de efetuar um login válido. - Tem de haver uma campanha de parametrização criada com a tipologia do programa "Segmentado".
Caminho Principal	1. O ator tem de seguir todos os passos descritos no caso de uso "Consultar Programa". 2. O ator seleciona o botão "Importar ficheiro". 3. O sistema apresenta uma janela para a seleção do ficheiro. 4. O ator seleciona o ficheiro que deseja importar. 5. O ator seleciona o botão "Guardar". 6. O sistema imprime uma mensagem de sucesso, em como os clientes foram carregados com sucesso e a aplicação os está a importar para a campanha. 7. O sistema fecha a janela de importação de ficheiros.
Caminhos Alternativos	N/A
Suplementos ou adornos	- Verificar se o ficheiro se encontra no formato correto para importação.
Pós-Condição	O sistema guarda na base de dados todos os clientes que foram importados e associa-os àquela campanha de fidelização.

Descrição do Caso de Uso "Inativar um programa"

Nome	Inativar um programa.
Descrição	O objetivo deste caso de uso é o administrador poder inativar uma campanha de fidelização.
Pré-Condição	- O ator tem de efetuar um login válido. - A campanha a inativar tem de estar ativa para ser inativada.
Caminho Principal	1. O ator tem de seguir todos os passos descritos no caso de uso "Consultar Programa". 2. O ator seleciona o botão "Inativar Programa". 3. Se o programa for diário, o sistema apresentar uma janela *pop-up* a confirmar se o ator deseja mesmo inativar o programa. 4. O ator seleciona, novamente, o botão "Inativar". 5. O sistema apresenta uma mensagem de sucesso, e deixa o programa inativo apenas para consulta de parametrização.
Caminhos Alternativos	3. Se o programa for mensal ou trimestral, o sistema apresenta uma mensagem com uma sugestão de uma data, para a campanha assim acabar os cálculos do mês ou do trimestre. 4. O ator, se concordar com a data sugerida pelo sistema, seleciona o botão "Editar", altera a a data de fim do programa e grava. 4. O ator se pretender inativar a campanha imediatamente, seleciona o botão "Inativar Agora". 5. O sistema volta a apresentar uma janela *pop-up,* com uma mensagem de confirmação se realmente pretende inativar o programa naquele momento. 6. O ator, voltar a selecionar o botão "Inativar". 7. O sistema apresenta uma mensagem de sucesso, e deixa o programa inativo apenas para consulta de parametrização.
Suplementos ou adornos	- Verificar a tipologia do programa, se é diário, mensal ou trimestral.
Pós-Condição	O sistema guarda na base de dados a alteração feita pelo ator.

Descrição do Caso de Uso "Listar Clientes com Paginação"

Nome	Listar clientes com paginação.
Descrição	O objetivo deste caso de uso é o ator, consultar uma lista de clientes.
Pré-Condição	- O ator tem de ter um login válido. - Tem de haver no mínimo uma campanha de parametrização criada, com clientes já associados à mesma.
Caminho Principal	1. O ator encontra-se na página principal. 2. O ator seleciona o botão "Lista de Clientes". 3. O sistema faz uma passagem entre ecrãs e apresenta uma lista de clientes paginados, ordenados por número de identificação fiscal.
Caminhos Alternativos	N/A
Suplementos ou adornos	N/A
Pós-Condição	N/A

Descrição do Caso de Uso "Pesquisar um cliente"

Nome	Pesquisar um cliente.
Descrição	O objetivo deste caso de uso é o ator pesquisar um cliente.
Pré-Condição	O ator tem de ter um login válido.
Caminho Principal	1. O ator segue os passos todos descritos no caso de uso "Listar Clientes". 2. O ator seleciona o botão "Pesquisar". 3. O sistema faz uma passagem entre ecrãs, e apresenta alguns campos de filtro, assim como: • Nome • Número de identificação fiscal • *Token* do cartão/conta • Número TAP 4. O ator, preenche os campos desejados. 5. O ator seleciona o botão "Pesquisar". 6. O sistema apresenta, uma lista de cartões referentes aos clientes que correspondem aos filtros pesquisados.
Caminhos Alternativos	6. Se não existir nenhum cliente que corresponde aos filtros, o sistema apresenta uma mensagem de aviso.
Suplementos ou adornos	N/A
Pós-Condição	N/A

Descrição do Caso de Uso "Consultar cartões por Programa"

Nome	Consultar cartões por Programa.
Descrição	O objetivo deste caso de uso é o ator poder consultar que cartões pertencem a cada campanha de fidelização dentro dos detalhes de cada cliente.
Pré-Condição	O ator tem de efetuar um login válido.
Caminho Principal	1. O ator tem de efetuar todos os passos descritos no caso de uso "Pesquisar um cliente". 2. O ator seleciona o botão "Detalhes", no cartão referente ao cliente que se quer selecionar. 3. O sistema faz uma passagem entre ecrãs, e apresenta as informações detalhadas do cliente, assim como, a que campanha pertence cada cartão do mesmo.
Caminhos Alternativos	1. O ator encontra-se no ecrã "Lista de Clientes", e seleciona o ícone da "Lupa", para consultar os detalhes do mesmo.
Suplementos ou adornos	N/A
Pós-Condição	N/A

Descrição do Caso de Uso "Consultar Conta-Corrente"

Nome	Consultar Conta-Corrente.
Descrição	O objetivo deste caso de uso é o ator poder consultar a conta-corrente de um cartão de um cliente, dentro de uma campanha de fidelização.
Pré-Condição	O ator tem de efetuar um login válido.
Caminho Principal	1. O ator tem de efetuar todos os passos descritos no caso de uso "Consultar cartões por programa". 2. O ator seleciona ícone "Lupa", na coluna "Detalhes". 3. O sistema faz uma passagem entre ecrãs, e apresenta um ecrã com 3 abas. 4. O ator seleciona na aba "Conta-Corrente". 5. O sistema apresenta uma tabela com a conta-corrente, daquele cartão naquela campanha.
Caminhos Alternativos	N/A
Suplementos ou adornos	N/A
Pós-Condição	N/A

Descrição do Caso de Uso "Consultar registos de benefícios atribuídos"

Nome	Consultar registos de benefícios atribuídos.
Descrição	O objetivo é o ator consultar registos de benefícios atribuídos por uma campanha de fidelização, numa determinada conta-corrente.
Pré-Condição	O ator tem de efetuar um login válido.
Caminho Principal	1. O ator tem de efetuar todos os passos descritos no caso de uso "Consultar cartões por programa". 2. O ator seleciona ícone "Lupa", na coluna "Detalhes". 3. O sistema faz uma passagem entre ecrãs, e apresenta um ecrã com 3 abas. 4. O ator seleciona a aba "Registos de Conta Corrente" 5. O sistema apresenta uma tabela com os registos paginados com quantidade, descrição e data de atribuição do benefício.
Caminhos Alternativos	N/A
Suplementos ou adornos	N/A
Pós-Condição	N/A

Descrição do Caso de Uso "Consultar resgates de vouchers/prémios"

Nome	Consultar resgates de vouchers/prémios.
Descrição	O objetivo é o ator consultar registos de vouchers/prémios já resgatados por um cartão, numa determinada campanha de fidelização.
Pré-Condição	O ator tem de efetuar um login válido.
Caminho Principal	1. O ator tem de efetuar todos os passos descritos no caso de uso "Consultar cartões por programa". 2. O ator seleciona ícone "Lupa", na coluna "Detalhes". 3. O sistema faz uma passagem entre ecrãs, e apresenta um ecrã com 3 abas. 4. O ator seleciona a aba "Prémios/Vouchers Resgatados" 5. O sistema apresenta uma tabela com os registos paginados.
Caminhos Alternativos	N/A
Suplementos ou adornos	N/A
Pós-Condição	N/A