NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

TIAGO ALEXANDRE MACIEL VENTURA

Licenciado em Engenharia Informática

TRANSFORMAR PRODUTO LEGADO NUM SERVIÇO CLOUD

MODELOS DE EXTENSÕES NUM SISTEMA SAAS MULTITENANT EM CLOUD

Dissertation Plan
MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Outubro, 2023

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

TRANSFORMAR PRODUTO LEGADO NUM SERVIÇO CLOUD

MODELOS DE EXTENSÕES NUM SISTEMA SAAS MULTITENANT EM CLOUD

TIAGO ALEXANDRE MACIEL VENTURA

Licenciado em Engenharia Informática

Orientador: Filipe Mendes Correia
Engenheiro Informático, Link Consulting

Coorientador: Artur Miguel Dias
Professor Auxiliar, Universidade NOVA de Lisboa

Dissertation Plan
MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Outubro, 2023

Resumo

A Link Consulting é uma empresa de consultoria e de desenvolvimento de software, sendo que um dos seus produtos principais é o sistema edoclink, um software de gestão documental e de processos. Este produto foi subscrito e está em uso em muitas empresas, particularmente nos países de expressão portuguesa.

A motivação para melhorar o edoclink é muito grande. Está em desenvolvimento uma nova versão baseada numa arquitetura SaaS multitenant. Uma funcionalidade em que existe a intenção de investir muito esforço é a possibilidade de dar ao utilizador a opção de escrever, ele mesmo, algum código para estender o sistema com novas funcionalidades que resolvam problemas locais. Este é o tema central desta dissertação.

Este é um problema interessante, visto existir muito pouca literatura técnica sobre este assunto. A par deste facto, são raras as empresas que desenvolvem software SaaS multitenant que tenham suporte com extensibilidade.

Este documento apresenta uma solução possível para este problema, implementada na nova versão do edoclink. As ideias principais são as seguintes:

- Todas as extensões são executadas localmente, no servidor do cliente.
- Desenvolver SDKs para as linguagens de programação mais usadas por forma a ajudar o utilizador a criar extensões sem demasiada dificuldade, maximizando a sua produtividade.
- Cada SDK conterá métodos tanto para o envio de dados, utilizando a API disponibilizada pelo edoclink, como para receber dados, através de eventos utilizando WebHooks. Além disso, a questão da segurança terá de ser tratada.
- Aproveitar a opção que agora existe de gerar SDKs completos através do *OpenAPI codegen*. Muito recentemente ocorreu um importante anúncio de uma evolução da especificação do OpenAPI que passa a permitir o uso de WebHooks.

Este relatório também contém uma introdução, uma descrição bastante completa de conceitos, técnicas que foram necessárias considerar, metodologia de trabalho da equipa do edoclink e um resumo bastante completo da implementação deste projeto.

Palavras-chave: SaaS, extensibilidade, multitenant, WebHook

Abstract

Link Consulting is a consulting and software development company, and one of its main products is the edoclink system, a document management and process software. This product has been subscribed and is in use by many companies, particularly in Portuguese-speaking countries.

The motivation to improve edoclink is very high. A new version based on a multitenant SaaS architecture is being developed. One functionality in which there is a intention to invest a lot of effort is the possibility of giving the user the option to write their own code to extend the system with new functionality that solves local problems. This is the central theme of this dissertation.

This is an interesting problem, as there is very little technical literature on this subject. In addition to this fact, there are few companies that develop multitenant SaaS software with extensibility support.

This document contains a possible solution for this problem, implemented in the new edoclink version. This dissertation main ideas are as follows:

- Every extension will be run on the user environment
- Develop SDKs for the most commonly used programming languages in order to help the user create extensions without too much difficulty, maximizing their productivity.
- Each SDK will contain methods both for sending data using the API provided by edoclink, as well as for receiving data through events using WebHooks. In addition, the issue of security will have to be addressed.
- Take advantage of the option that now exists to generate complete SDKs through *OpenAPI codegen*. Indeed, very recently there was a significant announcement of an evolution of the OpenAPI specification that now allows the use of WebHooks

This report also contains an introduction, a fairly comprehensive description of concepts and techniques that were necessary to consider, edoclink team work methodology and a very comprehensive summary of the implementation of this project.

Keywords: SaaS, extensibility, multitenant, WebHook

Índice

Índice de Figuras

Índice de Listagens

1 Introdução

1.1 Sobre a Link Consulting

Link Consulting é uma empresa portuguesa criada em 2000, separando-se do grupo INESC, o principal instituto tecnológico de IT na área de Lisboa. Faz parte do grupo Aitec, que é o grupo baseado em tecnologia mais antigo de Portugal.

1.2 Sobre o edoclink

O edoclink é um software de gestão documental criado e mantido pela Link Consulting. Este sistema está a ser utilizado em empresas de vários países, incluindo Portugal, Angola e Moçambique. Além de se tratar de um sistema de gestão documental, o edoclink é também um software de gestão de processos. Neste contexto considera-se que um processo consiste num conjunto de etapas de uma tarefa cada uma executada por uma ou mais pessoas, sendo este conceito mais explorado nas secções seguintes, mais em concreto na secção 1.4.

1.3 Motivação e contexto

Sendo o edoclink uma ferramenta de automatização de fluxos e documentos, existe uma grande motivação para melhorar esse serviço, dando ao utilizador a opção de executar

código feito pelo cliente para acelerar e adicionar funcionalidades não existentes ao edoclink. Esta funcionalidade já existe em parte na versão atual do edoclink, porém cada cliente tem a sua própria instância do servidor e da base de dados. Assim sendo, apenas é necessário adicionar um ou mais DLLs à instância do servidor de cada cliente.

Estando esta nova versão a ser desenhada de modo a suportar tanto multitenant como SaaS (ou *Software as a Service*), esta forma de adicionar DLLs torna-se impossível de concretizar, pois além de ser mais complexo de adicionar o tal DLL, essa extensão estaria disponível a todos os *tenants*, sendo que algumas destas funções são bastante focadas às necessidades de um determinado cliente. Com isto tornou-se obrigatório pensar numa nova tecnologia para este requisito de criar extensões.

Uma arquitetura multitenant [7] é uma arquitetura em que todos os clientes partilham a mesma instância, havendo portanto alguma lógica de separação de dados, quer seja a criação de múltiplas instâncias de bases de dados, quer pela introdução de um identificador em cada entrada da base de dados para associar essas entradas a um cliente. Com esta arquitetura temos uma maior facilidade de resolução de erros, visto que podemos utilizar um sistema de integração contínua para fazer a atualização do sistema constantemente a para todos os clientes ao mesmo tempo, perdendo-se assim a necessidade de trabalhar em várias versões visto que todos os clientes trabalharam na mesma.

Um aspeto a ter em conta com uma arquitetura multitentant é a segurança. Visto que todos os clientes terão acesso ao mesmo servidor a privacidade e segurança dos dados de todos os clientes é uma obrigação, sendo um aspeto bastante importante na implementação e análise do tema proposto nesta dissertação.

SaaS, ou *Software as a Service*, é uma arquitetura em que um serviço (ou aplicação) é disponibilizado pela Internet para a realização de todas as tarefas disponíveis em tal sistema, de forma remota, quer seja pelo computador, quer por dispositivos móveis. Isto elimina a necessidade dos clientes terem de instalar servidores em instâncias locais, sendo os servidores hospedados pela própria empresa (quer seja em computadores da empresa ou em serviços na nuvem disponibilizados por terceiros).

Nesta dissertação iremos falar do edoclink que está atualmente disponível para clientes como antigo edoclink e o edoclink que está a ser desenvolvido como novo edoclink.

1.4 Funcionamento do edoclink

O edoclink é um software de gestão documental criado pela Link Consulting. Este é um programa cliente/servidor que atualmente está a ser executado num modelo *single-tentant*, estando a instância de cada cliente hospedada num servidor na rede de cada cliente. A sua interface é acedida através de um browser, sendo que também existe um programa agente para os computadores de cada utilizador de modo a poder ser executado algum código fora do browser (como abrir programas quando se corre alguma ação). Além da interface web e do agente, o edoclink também tem extensões para as aplicações Microsoft

Office, tendo assim uma melhor integração entre o sistema de gestão documental com o sistema de edição de documentos.

Para além de fazer a gestão documental de uma empresa, também têm funcionalidades de gestão de processos. Processos são um conjunto ordenado de tarefas, que podem ser executadas por uma ou mais pessoas, que tem como finalidade automatizar o fluxo de trabalho entre entidades. Tanto os processos como cada etapa poderão ter campos adicionais. Estes campos adicionais são campos personalizados pelo cliente, sendo que estes podem ser de vários tipos de dados, como alfanumérico, tabela, lista, numérico, entre outros. Cada campo tem um nome, descrição e valor por defeito.

Um exemplo simples de um processo é o lançamento de uma newsletter. Este processo começará com o departamento de compras, que deverá ver quais os produtos se poderá fazer um bom preço para a newsletter em questão. Esta etapa poderá ter uma tabela com os produtos e os preços. Assim que esta etapa estiver concluída, o utilizador que a está a executar pode então dar como finalizada, o que inicia a etapa seguinte. Esta etapa será dedicada à produção da newsletter em si, que no final deverá ser adicionada como documento à etapa. A etapa seguinte poderá ser de validação do documento, onde uma entidade terá de validar a newsletter, desde erros ortográficos a preços. Caso exista algum erro, o utilizador poderá adicionar o erro num campo personalizado e reabrir a etapa anterior, sendo que as entidades responsáveis por esta serão alertadas. Caso esteja tudo correto poderá seguir para etapa seguinte que será a distribuição da tal newsletter, ficando assim finalizado o processo.

Dado que esta ferramenta poderia servir para gerir grande parte da empresa, deu-se a necessidade de a expandir. Com isto foram criadas ações dentro de uma etapa, uma funcionalidade poderosa que permite que o utilizador possa executar qualquer função dentro destas. Alguns exemplos de ações poderão ser abrir um formulário personalizado com a etapa em questão, enviar um pedido para uma API com alguma informação da etapa e o resultado deste alterar um campo da etapa ou até enviar um email à entidade responsável pela etapa anterior.

Depois de concluir este módulo, alguns clientes necessitaram de mais controlo sobre alguns aspetos destes contextos, aparecendo a necessidade de criar páginas personalizadas com este efeito. Tendo em conta o número de clientes com esta necessidade, ficou decidido criar um construtor de formulários, chamado de FormBuilder. Estes formulários iriam ficar ligados com determinados contextos (sendo um contexto uma etapa, processo, pasta ou registo), ou seja, os campos do formulário iriam ler o valor do contexto e ao submeter iriam atualizar o valor dos campos no contexto. Com a introdução de lógica no FomrBuilder ficou possível de os clientes poderem criar páginas personalizadas com facilidade e estas tornaram-se relativamente rápidas de se construir. Visto a modularidade que esta ferramenta trouxe ao edoclink continuou a ser expandida desde então, sendo agora possível alterar estilos, utilizar "páginas"virtuais dentro de um formulário, entre muitas outras funcionalidades.

1.4.1 O FormBuilder

Como dito anteriormente, o FormBuilder é um sistema que faz parte do edoclink. O principal objetivo deste software é substituir grande parte das páginas personalizadas que os clientes têm (quer sejam feitas pela equipa do eDoc a pedido de clientes ou feitas pelos próprios clientes).

O FormBuilder tem uma interface *drag&drop* onde se pode adicionar, remover, editar e alterar a ordem de campos num formulário. Ao se criar um campo novo este é automaticamente criado no eDoc, ficando assim ligado com o resto do sistema. Também dá para inserir um campo já criado, sendo só preciso arrastar a entrada "Campo do edoclink"para o formulário.

Para além de campos de dados, existem campos que não são mapeados para o edoclink. Estes têm duas funções, sendo estas de estética (campos de imagem, título, colunas, conjunto) e de organização (campos colunas e conjunto). Estes últimos são áreas onde se podem colocar campos, dentro de um conjunto com um título (no caso do conjunto) ou em colunas. Outro tipo de campos que não são mapeados para o eDoc são os botões. Estes têm a funcionalidade de executar ações, sendo que estas podem ser algumas funções pré-configuradas de ligação com o eDoc e podem executar simples lógica. Esta lógica são simples funções que são executadas a nível dos campos, tais como atualizar os seus dados e alterar a visibilidade.

A lógica no FormBuilder são funções que têm condições e ações, que são corridas sempre que algum valor é alterado. Pode ser executada lógica em todos os campos, porém alguns têm algumas opções bloqueadas (por exemplo alterar o valor de um campo imagem). Dentro das ações algumas podem ser executadas em todos os campos, como por exemplo a visibilidade. As ações, para além de serem executadas na lógica, também podem ser executadas em botões, ficando assim possível executar algumas funções mais complexas, como por exemplo uma paginação. Neste exemplo podemos ter vários conjuntos de campos (que podemos chamar de páginas) e botões em cada um que permite a alteração de página (um botão tem lógica para esconder o conjunto atual e mostrar o conjunto seguinte).

Dentro dos botões existe mais lógica, como gerar PDF com os campos do formulário (de forma a gerar um relatório com os valores), submeter o formulário, criar contextos, atualizar etapas, alterar página, entre muitos outros.

Com estas funcionalidades, estes formulários podem substituir a maior parte das páginas personalizadas, aumentando assim a velocidade com que os clientes podem ajustar o edoclink às suas necessidades.

1.4.2 Integração do edoclink nas empresas

Os gestores de documentos são ferramentas bastante importantes para as empresas visto que todos os documentos legais e não legais ficam guardados de forma digital, aumentando a facilidade de procura de documentos e removendo o risco de se perder documentos importantes. Algo também muito importante para as empresas é a partilha fácil e segura de documentos, sendo que desta forma nunca nenhum documento será perdido ou destruído.

Outra funcionalidade do edoclink muito procurada pelas empresas hoje em dia é a gestão de processos ou fluxos, que como explicado anteriormente agiliza a passagem de tarefas de pessoas para pessoas ao longo de um projeto. Isto dificulta o esquecimento de passos numa tarefa, ao mesmo tempo que as impõe, não dando portanto para saltar um passo (uma newsletter sair sem a validação dos preços por exemplo). Estas funcionalidades são utilizadas por empresas de todos os setores. Por exemplo, uma faculdade poderia criar um fluxo para os documentos da dissertação. Ao contrário do que acontece atualmente (em que os documentos passam de um lugar para outro sem nenhuma automação), poderia existir um fluxo com os documentos digitais em que as tarefas passam de divisões e conseguimos, de forma simples, analisar o progresso de cada fluxo e qual é a etapa em que cada fluxo se encontra. Uma mais-valia do edoclink é também a assinatura digital, ficando assim possível assinar um documento sem ser preciso imprimir e digitalizar o documento depois de assinado.

No final, estes sistemas aumentam a produtividade e reduzem o número de erros que ocorrem dentro das empresas, resultando num aumento de rendimento.

1.5 *Software as a Service*

SaaS, ou Software as a Service, é um modelo de negócio em que toda a estrutura de um projeto está hospedado na cloud e acessível a todos os utilizadores.

Algumas vantagens deste modelo são o menor custo de utilização, o facto de o cliente não necessitar de máquinas para hospedar o software, o acesso em qualquer lugar, as atualizações automáticas e as integrações com outros sistemas de forma facilitada. [24]

1.6 *Multi-tenancy*

Um sistema multi-tenant é um sistema em que vários clientes partilham um backend partilhado. Com isto, em vez de cada cliente ter a sua própria instância, iram utilizar um sistema partilhado onde apenas existe uma separação lógica dos dados. Estes dados poderão estar na mesma base de dados ou poderão estar em bases de dados separadas, sendo que o backend terá de escolher a base de dados apropriada para cada pedido. [22]

1.7 Contribuições

No contexto desta dissertação, as minhas contribuições para este projeto foram:

- Análise dos mecanismos de extensibilidade do antigo eDoc.
- Análise e desenho da implementação de cada tipo de extensão, endereçando as necessidades de segurança de um sistema SaaS multi-tenant.
- Implementação da arquitetura, sendo que alguns dos componentes ficaram apenas sobre a forma de *POCs (Proof Of Concept)*, com diferentes graus de desenvolvimento, pois dependiam de funcionalidades que ainda não existem no produto, apesar de estarem planeadas.

1.8 Sobre este documento

Este documentos está dividido em 5 capítulos, sendo estes:

- **Capítulo 1: Introdução** - Neste capítulo faço uma introdução geral do edoclink e do FormBuilder e a sua importância para a gestão interna de uma empresa. É também explicado neste capítulo a motivação para a realização desta dissertação.
- **Capítulo 2: Estado da arte** - Neste capítulo começo por apresentar o estado da arte, sendo que no final não foi encontrado nenhum projeto com as mesmas restrições que este requer, como ser um sistema SaaS multitenant. De seguida, faço uma análise de algumas tecnologias de extensibilidade para aplicações multitenant, sendo que no final escolho a que melhor se adequa ao problema dado.
- **Capítulo 3: Extensibilidade no antigo edoclink** - Neste capítulo apresento uma breve introdução de todos os tipos de extensão que existem no antigo edoclink. Para a realização desta dissertação, algumas das extensões presentes no antigo edoclink não foram implementadas
- **Capítulo 4: Requisitos e a nova arquitetura** - Aqui faço uma breve solução que foi proposta para ser implementada. A proposta contém cinco módulos para o backend e dois para o frontend.
- **Capítulo 5: Metodologia de desenvolvimento da equipa/produto** - Neste pequeno capítulo é apresentada a metodologia de trabalho da equipa, desde procedimentos a algumas ferramentas de apoio ao desenvolvimento. Também é apresentada um pouco da dinâmica entre as várias equipas do produto.
- **Capítulo 6: Estrutura dos projetos eDoc** - Aqui farei um breve resumo da estrutura de todos os projetos que fazem parte do sistema eDoc. Estes são o frontend, o backend e o FormBuilder.
- **Capítulo 7: Estrutura da solução** - Neste capítulo apresentarei todos os pontos analisados antes da implementação das soluções. Também foram analisadas algumas questões de segurança referentes a cada tipo de extensão.
- **Capítulo 8: Implementação da solução** - Aqui apresentarei a forma como foi implementada cada solução, tal como alguns problemas encontrados durante a dita

implementação e as suas soluções.

- **Capítulo 9: Validação dos resultados** - Neste capítulo final está apresentado a forma como foram validados os resultados.

2

Estado da arte

Neste capítulo vou apresentar algumas tecnologias já criadas (tanto por outras empresas como protocolos) disponíveis para a realização deste projeto e apresentar algumas implementações de outros sistemas.

2.1 Exemplos de implementações de extensibilidade por outras empresas

Para uma primeira análise deste problema, primeiro foram analisadas diversas ideias sobre como outros sistemas resolveram este problema. Dentro destas, a primeira que foi analisada foi o modelo de extensões do Azure DevOps.

Neste capítulo, o termo evento aplica-se a acontecimentos de alto nível ou relacionados com a lógica da aplicação. Temos como exemplo quando uma compilação acaba de executar ou quando um *pull request* é aprovado.

2.1.1 Azure DevOps

O Azure DevOps Server [2] é um produto da Microsoft que fornece um serviço de controlo de versão (através de um serviço git), gestão de projetos e de requisitos, serviço CI/CD, progressão de requisitos com validação de testes e recursos de gestão de versões. Resumindo, este sistema cobre todo o ciclo da vida de uma aplicação e habilita os recursos de DevOps.

No modelo do Azure DevOps nenhuma ação é executada no servidor, sendo que todas as alterações necessárias de fazer nos dados serão feitas através de uma API. Este modelo é bastante positivo nas questões de segurança, utilizando páginas customizadas e ServiceHooks para todas as extensões pretendidas. ServiceHooks é uma tecnologia *pub/sub* (existe um serviço subscrito a um evento e outro que publica um evento) da Microsoft que permite executar tarefas noutro serviços quando algum evento acontece dentro de um projeto do Azure DevOps. Por exemplo, podemos criar um cartão no Trello quando um item for criado e enviar uma notificação para o Temas quando uma compilação falha.

Um tipo ServiceHook importante de se analisar são os WebHooks, que permitem criar aplicações personalizadas que utilizem os recursos do Azure DevOps. Estes WebHooks terão o formato de POST que o servidor do Azure DevOps fará para um servidor cliente, sendo que este deverá ser configurado pelo utilizador.

2.1.2 Visual Studio Code

Para além de sistemas *SaaS*, também podemos tirar inspirações de outros tipos de software, tal como o ambiente de desenvolvimento de software Visual Studio Code. Neste sistema, o modelo de extensões permite adicionar extensões de linguagens (como codificação por cores e auto-completar), depuradores e outras ferramentas à instalação local. Desta forma, um autor pode ligar extensões diretamente à UI do VS Code e contribuir para a sua funcionalidade utilizando as mesmas APIs utilizadas por este sistema. Uma inspiração que podemos recolher desta implementação é o uso de um SDK para assistir o utilizador na criação de extensões, dispondo de uma série de métodos pré-feitos com as funções mais utilizadas. Desta forma, podemos aumentar a produtividade do programador por duas vias: minimizar a quantidade de erros por parte do cliente; e reduzindo o tempo de programação utilizado para criar as suas extensões.

2.1.3 Software Open Source: OneDev

OneDev [19] é um servidor Git com CI/CD e Kanban. Este sistema tem duas formas de extensão, sendo estas o uso de WebHook e a criação de *scripts* Groovy. Neste software não há a possibilidade de criar páginas *web*, sendo a única forma de customizar algo no frontend será a edição de campos em diversos conceitos.

Com os WebHooks podemos tratar diversos eventos, sendo que estes são customizados por projeto e por evento, cada um tendo a sua chave. Para alterar dados no sistema tem de ser utilizada a API. Desta forma a extensibilidade é criada num sistema fora do serviço, sendo assim mais seguro e mais eficiente por parte do servidor.

2.1.4 YouTrack

YouTrack [34] é um sistema de gerenciamento de projetos e rastreio de bugs comercial e proprietário. A interface é baseada numa tecnologia web, sendo que pode ser gerido *on premises* ou na *cloud* (sendo esta *cloud* propriedade da JetBrains).

A forma como este sistema pode ser estendido pelo utilizador final é através de WebHooks, sendo que a forma dele funcionar é igual tanto ao Azure DevOps como ao OneDev. Quando um dado evento é acionado, este é enviado pelo WebHook para os sistemas que estão à escuta. A alteração de dados é feita à base de uma API disponibilizada pelo sistema.

2.1.5 edoclink Formbuilder

O edoclink Formbuilder é o sistema proprietário da Link Consulting. Este é um construtor de formulários *no-code* (em que um utilizador não necessita saber programar) com ligação ao edoclink. Esta ferramenta veio da necessidade de ajudar os utilizadores a criarem páginas personalizadas de forma rápida e segura. Assim, em vez de um cliente ter de criar extensões no edoclink em ASP.Net (sendo que um cliente que não soubesse utilizar a tecnologia teria de pedir a equipa da Link para criar essa página) pode apenas arrastar os campos necessários para o formulário, sendo que estes estão ligados a campos e contextos do eDoc, de forma a facilitar a integração com o sistema principal.

Uma das principais funcionalidades deste serviço é a própria integração com o eDoc, sendo este sistema o único com esse efeito. Outra funcionalidade importante é o sistema de lógica, que permite de uma forma simples criar funções complexas dentro do formulário. Este também tem integração com BrightServices, que consiste numa ligação a um sistema WSO2 para chamar APIs externas ao edoclink.

Este software foi construído em Angular, sendo este depois compilado e colocado numa página ASP.Net no sistema principal, sendo acessível através de uma página dentro do edoclink.

2.1.6 Camunda Forms

Camunda forms [6] é um sistema simples de construção de formulários online baseado numa biblioteca *open-source* chamada bpmn-io form-js library.

Este sistema é bastante simples, apenas permitindo lógica de visibilidade de campos, não permitindo algumas funções mais avançadas como receber dados através de uma API. Também não tem integrações, sendo assim um sistema bastante fechado.

2.1.7 Jotform

Jotform é um construtor de formulários online que permite construir formulários sem exigir qualquer conhecimento de programação [13]. Tem uma interface intuitiva e faz uso do *drag&drop*, o utilizador define os campos a integrarem o seu formulários a partir de uma lista que se encontra à esquerda, dividida em campos simples(como um título ou uma caixa numérica), ou campos pré-definidos, dividindo-se em campos especiais(como campos para pagamentos e *widgets*) e campos de utilização mais simples (como morada, *rating* ou marcar data e hora). Cada campo possui um conjunto de propriedades onde é possível definir um *placeholder*, uma descrição, um valor predefinido, ser obrigatório, estar visível e não ser um campo de escrita. Para lógicas mais complexas (como mudar o valor de um campo a partir de outro ou esconder um campo com base no que está ou não está escrito noutro), esta ferramenta oferece uma solução para isso, onde basta escolher o campo da condição, o estado em que a condição acontece e o valor, depois escolhemos o que queremos fazer escolhendo o campo e a ação a tomar. O Jotform tem também a

possibilidade de integração via WebHook para poder preencher campos do formulário. Esta ferramenta oferece ainda integração com diversos serviços de pagamento (como PayPal, Skrill, Venmo, Apple Pay, entre outros). A criação de PDFs e o envio de emails são também possíveis de serem feitos pelo Jotform [13].

2.2 Tecnologias disponíveis

Sendo este software um sistema multitenant, a atenção à autorização e segurança tem de ser reforçada. Com isto, o uso de *assemblies* (pedaços de software que são lidos e executados em *runtime*) terá de ser eliminado caso possível. Visto isto, a melhor solução será utilizar um serviço *pub/sub* no servidor cliente, tal como é feito no Azure DevOps.

Para esta tecnologia pensamos em diversos caminhos.

2.2.1 HTTP Long Polling

Primeiramente, pensamos em HTTP Long Polling [1], uma tecnologia em que o cliente faz um pedido HTTP ao servidor de destino. Porém a resposta é atrasada até haver a informação que o utilizador pediu. Assim sendo, o servidor não tem de esperar que o cliente faça um pedido, o que consome recursos desnecessários. Neste caso, o cliente teria de perguntar se havia alguma novidade nos dados repetidamente, o que consome não só recursos do servidor mas também do cliente, visto que viriam respostas negativas que poderiam apenas ser omitidas. Isto também não é instantâneo para o cliente, que dependendo do tempo entre pedidos poderia ter uma resposta bastante atrasada. Porém, esta ferramenta não é a ideal para estes casos, gastando mais recursos que outras.

Uma analogia que se pode fazer com esta tecnologia é uma pessoa fazer uma pergunta a outra e ficar à espera que essa pessoa responda. Neste caso, a pessoa que fez a pergunta terá de ficar à espera da outra, não podendo fazer mais nada entretanto.

2.2.2 Websocket

De seguida, pensamos em websockets [30]. Esta tecnologia consiste num simples socket bidirecional. Esta tecnologia é bastante útil quando temos alguma comunicação e com pedidos e respostas dos dois lados, ou se houver bastante comunicação unidirecional, que não é o nosso caso.

Uma analogia correspondente a esta tecnologia pode ser uma conversa entre duas pessoas. Qualquer pessoa pode iniciar um diálogo e a outra pode ou não responder, acabando a conversa ou continuando com outras respostas de ambas as partes.

2.2.3 WebHook

Uma outra tecnologia bastante relevante é a tecnologia WebHooks [3]. Esta tecnologia foi pensada precisamente para este tipo de uso (*event-driven architecture*), sendo bastante

utilizada e facilmente encontramos documentação [32] de como usar e de como melhor a proteger contra qualquer tipo de ataque. Outra vantagem de se usar uma tecnologia globalmente utilizada é a facilidade de implementação, sendo que já está implementada em praticamente todas as plataformas para desenvolvimento de backends. Neste projeto estamos a usar ASP.Net, que já tem a implementação desta tecnologia.

Os WebHooks funcionam da forma contrária ao protocolo HTTP comum. Em vez de um cliente fazer um pedido ao servidor e esperar uma resposta, um WebHook faz um pedido ao servidor para este enviar uma resposta quando existir um novo evento, sendo que o servidor enviará um pedido HTTP quando este acontecer. Assim sendo, um WebHook é um pedido que um cliente faz a um servidor para este submeter um pedido HTTP após um determinado evento. No caso do eDoc, o cliente também terá de conter um servidor HTTP, que no caso deste projeto teria a arquitetura REST.

Temos como exemplo de um WebHook uma aplicação que envia um email sempre que for adicionado um registo. Neste caso teríamos um servidor no cliente que regista no servidor que está à espera de um evento (neste caso a criação de um registo). Quando um registo for adicionado o servidor lança um evento (pedido HTTP) para todos os clientes que estejam subscritos a este. Depois desse evento ser despoletado, o cliente recebe a informação atribuída ao evento e valida se o registo tem as propriedades pretendidas e envia um email nesse caso.

Um ponto positivo sobre WebHooks é a fácil implementação de clientes. Um backend desenhado em C# poderá emitir eventos que sejam recebidos por clientes escritos em Node.JS ou Java, por exemplo. [33]

Uma analogia simples poderá ser uma pessoa pedir a outra para a avisar quando algo acontecer. Enquanto tal evento não acontecer a pessoa que fez a pergunta pode continuar a fazer outras tarefas, sabendo que quando o evento acontecer esta vai ser avisada

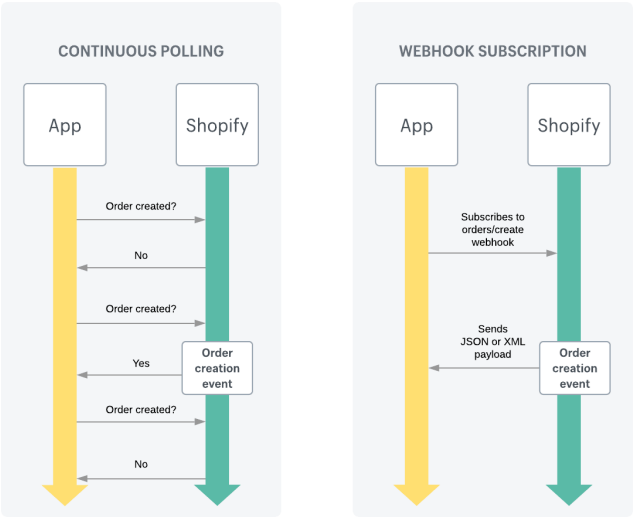

Figura 2.1: Como os WebHooks funcionam [11]

2.2.4 Conclusões sobre a análise de tecnologias para backend

Decidimos então utilizar WebHooks para esta extensão, pela facilidade de implementação e pelos nossos requisitos, dentro destes o facto da tecnologia ser leve para o servidor e ter pouca latência para o cliente, sendo que o WebHook cumpre todos.

Com a modularidade do WebHook e a utilização da API, qualquer código que anteriormente era executado dentro do backend pode ser executado num servidor no cliente, sendo que os eventos são lançados via WebHook e a alteração de dados será feita através da API.

2.2.5 Alteração de dados no servidor por parte de extensões

Sendo a redução de código corrido no servidor um requisito importante tornou-se necessário procurar uma alternativa para a alteração de dados no servidor por parte dos clientes. Visto o eDoc já conter uma API pública, ficou decidido que a melhor solução para este problema seria utilizar a API já criada. Esta API já contém um sistema de autenticação.

2.3 Tecnologias Frontend

2.3.1 Micro-Frontend

Um sistema monolítico é um sistema grande e inflexível, baseando-se normalmente numa única estrutura, plataforma e padrão.

Ao contrário de sistemas monolíticos, Micro Frontend é uma arquitetura que se baseia na separação da aplicação de frontend em "pedaços"pequenos (os micro frontends), sendo que cada um destes é isolado, podendo continuar a crescer se afetar a sua qualidade. Uma vantagem desta arquitetura [21] é a possibilidade de cada Micro Frontend ser desenvolvido numa plataforma diferente dos outros, como por exemplo temos o menu principal escrito feito em Angular e a página de configurações ser escrita em React. Cada Micro Frontend pode ser escrito por uma equipa diferente, dando assim mais liberdade na implementação por parte de cada equipa. [9]

Outro aspeto positivo desta arquitetura é a possibilidade de adicionar novas partes da aplicação sem ser necessário construir a raiz, sendo esta propriedade que será aproveitada para habilitar os clientes a construírem páginas customizadas. Dando acesso aos clientes as bibliotecas utilizadas no frontend da aplicação (como a biblioteca de componentes e a biblioteca de integração) estes podem criar páginas, sendo que para adicionar ao edoclink será apenas necessário hospedar os ficheiros compilados e criar um apontador dentro do edoclink para esses ficheiros, de modo a que estes sejam lidos nos browsers dos clientes.

Como podemos ver na figura 2.2, podemos ter três equipas (neste caso a *Team Product*, a *Team Checkout* e a *Team Inspire*) a trabalhar em três projetos diferentes que, quando apresentados ao cliente, aparecem os três na mesma página. Também podemos observar na imagem que os projetos não têm obrigatoriamente de ser utilizados apenas dentro da

aplicação principal. Neste caso, vemos que o projeto da *Team Checkout* e da *Team Inspire* estão a ser utilizados pelo projeto da *Team Product*.

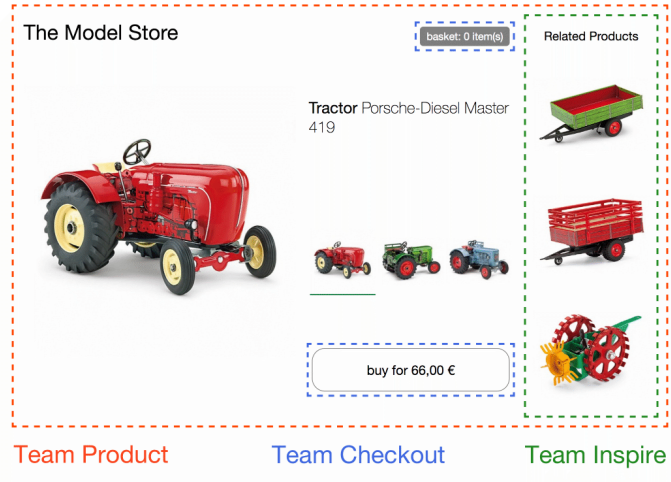

Figura 2.2: Exemplo de micro-frontend com três equipas [27]

2.3.2 Biblioteca de componentes frontend

Componentes são módulos que expandem o programa, tanto visualmente como funcionalmente. Para utilizar os componentes será apenas necessário adicionar a tag ao template HTML da página em questão, adicionando as propriedades do componente como parâmetros na tag deste. Em complemento da tradução já feita para todos os componentes, um cliente pode adicionar traduções adicionais para algum campo que necessite.

No contexto do edoclink, para uniformizar a interface e a funcionalidade do edoclink foi criada uma biblioteca com componentes standard que devem ser utilizados sempre que necessário. Desta forma temos algumas vantagens em relação ao componentes standard do HTML como estilos, lógica e localização, sendo que esta última é gerada para várias linguagens de todo o mundo, como português, inglês, francês e árabe.

Para a criação de páginas personalizadas deverá ser utilizada esta biblioteca, que ajuda tanto a uniformização de estilos de uma nova página, retirando a preocupação por parte do cliente de estilos e lógica, como facilidade em produzir novo conteúdo visto que a lógica mais complexa já está implementada.

Um exemplo de uma utilização de um componente poderá ser um botão, que em vez de utilizarmos a tag padrão do HTML utilizamos uma personalizada (<*edoclink-button*>) com as propriedades dentro dos parâmetros. Nesta biblioteca, para além de botões, existem diversos componentes como vários tipos de botão, listas, caixas de texto, data, entre muitos outros.

```
<edoclink-label text="Dropdown Multiselect Tree">
    <edoclink-dropdown-multi-select-tree
        [data]="multiselectDropdownTreeData"
        textField="text"
        valueField="id"
        [formControl]="formGroup.get('type') | formControl"
    ></edoclink-dropdown-multi-select-tree>
</edoclink-label>

<edoclink-label text="Cor">
    <edoclink-colorpicker
      [formControl]="formGroup.get('color') | formControl"
    ></edoclink-colorpicker>
</edoclink-label>

<edoclink-label text="Password">
    <edoclink-input-password
      [formControl]="formGroup.get('color') | formControl"
    ></edoclink-input-password>
</edoclink-label>

<edoclink-button title="Submeter" (onclick)="submeter()"/>
```

Listagem 2.1: Código de exemplo de uma página no eDoc

3

Extensibilidade no antigo edoclink

Neste capítulo irei falar das extensões que estão feitas no antigo edoclink, sobre o que são e como funcionam. Na introdução já foi feita uma pequena introdução. Com isto, neste capítulo vamos apenas falar sobre a extensibilidade do sistema.

3.1 Tipos de extensões de backend disponíveis no edoclink

Dentro do edoclink atual, já existem vários tipos de extensões diferentes. Porém, como iremos passar para um sistema multitenant, a base de todos terá de ser repensada.

3.1.1 Tipos de extensões disponíveis para backend

- Ações customizadas (Custom Actions)
- Conectores de extensão
- Gatilhos customizados (Custom triggers)
- Parâmetros customizados (Custom parameters)
- Relatórios customizados (Custom reports)
- SDK
- API REST
- BrightServices

3.1.2 Custom Actions

Estes são componentes que são executados por alguma ação do utilizador. Estas extensões podem ser executadas tanto no posto local como no servidor. Neste ponto apenas nos vamos focar na execução no servidor.

Estas ações são chamadas a um programa externo, sendo que são lançadas por alguma ação de um utilizador.

No antigo edoclink estas extensões são feitas através de assemblies escritos em C#. Estes são carregados diretamente no servidor através de um ficheiro DLL.

3.1.3 Conectores de extensão

Estes são componentes que se podem desenvolver e destinam-se a externalizar a gestão de entidades, autenticação e armazenamento de documentos.

De momento, estão feitos através da injeção de um DLL no sistema, sendo o DLL a implementação de uma interface disponibilizada pela equipa do edoclink.

3.1.4 Custom Triggers

Triggers são funções externas executadas perante vários eventos. Um exemplo de um trigger pode ser uma chamada a uma API externa na criação de um registo, e quando a resposta for recebida alterar um campo de um registo.

De momento estes triggers estão feitos com recurso a assemblies DLL escritos em C#

3.1.5 Custom parameters

Os custom parameters são parâmetros na base de dados do edoclink que podem ser utilizados em desenvolvimentos específicos e que fornecem uma página para a sua edição.

Estes campos são geridos através da uma página no frontend, sendo que não é necessário nenhum tipo de programação por parte do cliente para gerir estes parâmetros.

3.1.6 Custom reports

Os custom reports são pesquisas personalizadas na base de dados que consistem em *Stored Procedures* na base de dados que são depois convertidas para Microsoft Excel.

3.1.7 SDK

O SDK do edoclink é um biblioteca escrita em C# com várias funções que implementam praticamente todas as funcionalidades do edoclink. Desta forma, um cliente que necessita de fazer alguma alteração aos dados através de programa injetado por DLL pode-o fazer com um nível de abstração elevado, de modo a que não necessite de perceber o modo como o edoclink está estruturado.

3.1.8 API REST

A API REST do edoclink é uma API que, tal como o SDK, implementa várias funções para trabalhar com o eDoc. Ao contrário do SDK, a API não está desenhada para ser utilizada em código executado no servidor, mas sim para fazer programas externos que comuniquem de algum modo com o eDoc.

3.1.9 BrightServices

BrightServices são uma ligação entre o edoclink e APIs de sistemas externos, através de um *ESB* (Enterprise Service Broker) [12]. Neste caso é utilizado o WSO2, um sistema *open-source* que gere e publica APIs no modelo de nuvem, o que permite o uso destas APIs de forma segura e controlada. Com isto, o edoclink pode chamar "serviços" do WSO2 e este trata do resto.

3.2 Tipos de extensões de frontend disponíveis no edoclink

Para um modelo de extensões completo também será necessário implementar algumas tecnologias para a criação de páginas personalizadas. Para esta finalidade teremos duas implementações distintas, as páginas personalizadas e o FormBuilder.

3.2.1 Páginas Personalizadas

As páginas personalizadas são algumas páginas feitas externamente ao eDoc que se encontram dentro do sistema. Estas contém uma rota dentro do sistema e podem comunicar com o backend sem o uso de uma API (utilizando o sistema ASP.Net).

Visto o ASP.Net conter todas as páginas configuradas no backend, a implementação deste tipo de extensão é bastante semelhante às ações customizadas. A implementação destes dois tipos de extensão apenas difere no local onde os DLLs são injetados.

3.2.2 FormBuilder

O FormBuilder é algo que já existe no edoclink atual, porém está bastante isolado do resto do sistema. Esta extensão consiste num criador de formulários *drag&drop* que está ligado ao eDoc. Neste podemos adicionar campos (sendo que alguns destes mapeiam para campos do eDoc) ao formulário, alterar as suas propriedades e alterar a ordem. Mais recentemente foi também adicionada a funcionalidade de temas e estilos, que consiste em poder alterar cores, fonte e tamanho da fonte de vários campos e da própria página.

4

Requisitos e a nova arquitetura

Ao contrário do edoclink antigo, o novo edoclink terá uma arquitetura SaaS baseada em cloud. Com isto, uma nova arquitetura do sistema completo teve de ser pensada, tendo sido tomado em consideração todos os aspetos desta, como a possibilidade de estender no futuro.

Nesta arquitetura há uma separação entre o backend e o frontend, algo que não existia no edoclink antigo. O frontend também está divido em várias partes, chamadas micro-frontends, de modo a agilizar o processo de atualização de sistema e adição de novas funcionalidades.

À semelhança do frontend, o backend também é modular, utilizando micro-serviços. Dentro destes módulos, alguns foram criados para estabelecer a ligação entre o edoclink e outros serviços, como bases de dados, cacho e autenticação. Desta forma são facilmente adicionados novos módulos, fazendo com que a aplicação possa ser melhorada no futuro.

Este projeto tem um foco na parte da extensibilidade, sendo que a esta solução apenas apresenta soluções para este problema.

Para a implementação deste projeto foram implementados vários sistemas, tanto no backend, frontend e bibliotecas para apoio a criação de extensões no cliente.

Algo que é importante referir é que uma extensão não precisa de ter uma parte de backend e outra de frontend, por isso é que não há problema em estarem isoladas.

Com isto, os requisitos desta dissertação irão ser a implementação de todas as extensões descritas de seguida.

4.1 Backend

Para a implementação do backend, foram implementados vários sistemas, sendo que cada um destes será responsável por um ou mais tipos de extensões que existem no edoclink atual.

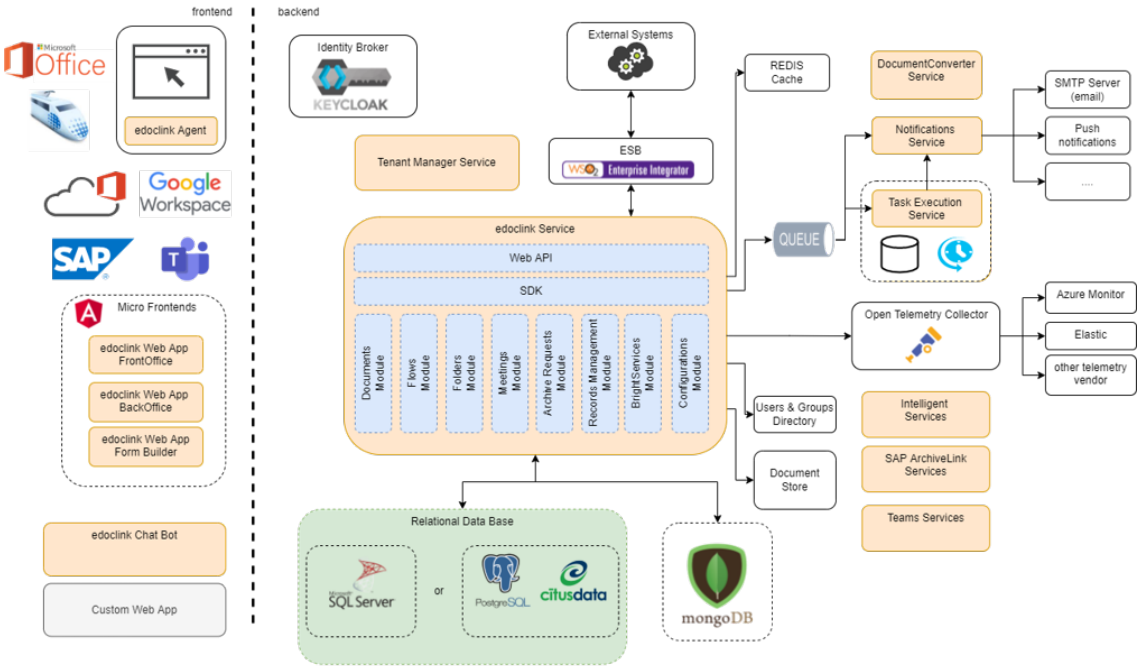

Figura 4.1: Arquitetura do novo edoclink

4.1.1 WebHook

Para resolver os custom triggers e custom actions, foi implementado um sistema de WebHooks. A primeira fase desta tarefa foi levantar todos os eventos que iriam ficar disponíveis no edoclink. De seguida, foi necessário implementar o servidor geral, sendo que nesta fase foi bastante importante a inspeção a nível de segurança e privacidade.

De forma a permitir a adição de eventos no futuro, foi necessário a criação de um sistema modular para se adicionarem eventos à medida que se vai adicionando funcionalidades.

4.1.2 Custom Parameters

Para a implementação da extensão de parâmetros customizados foi implementada uma interface para criação, edição e remoção de parâmetros numa base de dados. Estes parâmetros serão atribuídos a uma linha na base de dados através do uso de uma simples chave-valor.

4.1.3 Custom Reports

Por razões de segurança e pela pouca utilização desta extensão por parte de clientes ficou decidido que esta extensão não vai ser implementada na nova versão do edoclink.

4.1.4 SDK

Ao contrário do antigo edoclink, em que o SDK é implementado na mesma linguagem do servidor para ajudar os clientes a criarem código para serem injetados diretamente no servidor, aqui foram criados vários SDKs para algumas das linguagens mais utilizadas. Estes SDK têm interfaces para aceder à API e outras interfaces para o uso de WebHooks.

Com este SDK, para além de garantir algum nível de segurança no cliente, podemos dar um apoio à implementação de extensões do lado do cliente, visto que o cliente não precisará de perceber como o edoclink funciona, sendo que para executar alguma ação será apenas preciso executar algum método no SDK.

Para aumentar a velocidade na implementação do SDK, utilizá-mos o *OpenAPI CodeGen* para gerar o cliente API e WebHook em diversas linguagens.

4.1.5 BrightServices

Este módulo foi desenvolvido da mesma forma como no antigo edoclink, sendo a única diferença a possibilidade de configurar um sistema WSO2 para cada *tenant*.

Assim sendo, para a realização deste projeto ficou definida a criação de uma página de configuração e os métodos necessários no backend para concluir a comunicação com o serviço no servidor do cliente.

4.1.6 Conectores de extensão

Sendo este sistema administrado pela equipa do edoclink, ficou decidido que os conectores de extensões ficarão apenas disponíveis na versão *on-prem* do novo edoclink.

4.2 Frontend

No modelo de extensões do frontend, foram implementados dois sistemas, as páginas customizadas e o módulo FormBuilder.

4.2.1 Páginas customizadas

Visto que já está a ser utilizado um sistema de micro-frontend na implementação do frontend do novo edoclink, podemos apoiar a implementação deste tipo de extensões nesse sistema.

Foi então implementado um método de leitura dinâmica de módulos, sendo que estes módulos serão configurados na interface de configuração no frontend. Estes módulos serão hospedados pelo cliente, sendo depois lidos com recurso ao URL que os especifica. Para o acesso a estas páginas o cliente também deverá indicar o caminho no URL para o acesso ao módulo dentro do sistema edoclink.

Quando a aplicação abre, esta descarrega todos os módulos e adiciona as páginas necessárias ao menu.

4.2.2 FormBuilder

Atendendo a que este módulo já existe no edoclink, a sua implementação no novo sistema deveria ser simples. No entanto, pela forma como está integrado com o edoclink, tiveram de ser feitas bastantes alterações na raiz do projeto, algo que poderia trazer problemas em bastantes locais. Neste sentido, ficou decidido que irá ser criado outra versão do FormBuilder, feita de raiz para o novo edoclink.

5

Metodologia de desenvolvimento da equipa / produto

Neste capítulo vamos analisar a forma de trabalhar da equipa, mostrando de uma forma superficial a metodologia de trabalho e a organização da equipa.

5.1 Forma de trabalho

Todos os dias antes do almoço temos uma reunião diária. Nesta reunião todos fazem o ponto de situação do trabalho, pedem opiniões e recebem algumas instruções sobre o seu trabalho. Esta é uma parte importante do trabalho, pois além de conseguirmos ter uns apontamentos sobre o que estamos a fazer, também temos uma visão mais geral da aplicação em si, tal como os desenvolvimentos que estão a ser criados. Também cria uma dinâmica maior entre os colaboradores, visto que, não sendo a reunião formal, fala-se no início também um pouco fora do contexto do trabalho, ou seja, mais pessoais. Isto ajuda a conhecermo-nos melhor ao mesmo tempo que facilita a interação entre colaboradores. Para além dessa pequena introdução da reunião, mesmo durante a reunião por vezes fazem-se alguns comentários menos formais, estes mais relacionadas com o produto. Isto, além de ajudar a encontrar de forma menos crítica alguns pontos a melhorar, também ajuda a reunião, visto que se esta se torna menos monótona.

Para além da reunião, o produto encontra-se num repositório *GIT*, sendo que a gestão dos *bugs* e pedidos de novas funcionalidades estão num serviço próprio (o *Microsoft DevOps*). Aqui podemos encontrar a lista de tarefas a fazer, tal como detalhes, versões em conflito, histórico da tarefa, tarefas pendentes, estado e muitos outros parâmetros.

Além da reunião diária, também temos algumas chamadas durante o dia, onde podemos ajudar a resolver problemas, pedir ajuda com os nossos problemas ou falar sobre outro tipo de situações no trabalho. Estas reuniões tendem a ser esporádicas e curtas, sendo uma ferramenta útil para facilitar o trabalho em equipa.

Para toda a comunicação de trabalho entre colaboradores é utilizada a aplicação *Microsoft Teams*, um sistema de comunicação, videoconferência e chamadas desenhado para

equipas de trabalho. Já a comunicação oficial e geral para grande parte dos colaboradores é efetuada através de emails.

O projeto do eDoc está divido em 2 sub-projetos, sendo estes o frontend e o backend (cada um destes está ainda subdividido, mas para esta parte vamos ver os projetos como um todo). Estando o frontend a ser desenvolvido em Angular, para o desenvolvimento estamos a usar o editor de código *Visual Studio Code* com algumas extensões para facilitar o seu desenvolvimento. Já para o backend, que está a ser desenvolvido em .NET, utilizando a linguagem de desenvolvimento C#, está a ser utilizado o editor de código *Visual Studio 2022*.

5.2 Fluxo de trabalho

Como dito anteriormente temos o projeto num repositório *GIT*, este guardado no sistema *DevOps*. Para começar uma nova tarefa, entramos no sistema e escolhemos uma tarefa (por ordem de importância), indicando que essa tarefa fica ativa e que somos nós que a vamos resolver. De seguida criamos um novo ramo onde iremos resolver o problema ou desenvolver a funcionalidade. Ao criar o ramo, temos de ter em atenção a versão em que vamos resolver, pois como existem várias versões simultâneas no produto, podemos ter de resolver o problema em mais do que uma.

Após resolvermos a tarefa, criamos um *pull request*, onde colocamos algumas informações sobre a solução. Este é analisado por outros membros da equipa, em que leem o código que alteramos para ver se não existe nenhum problema. Depois de aceite, é compilado para a versão de teste interna onde a equipa de testes validará o problema. Caso este esteja resolvido, é colocada essa informação no *DevOps*, sendo que quando criada uma nova versão para clientes, todas as tarefas que tenham sido marcadas como resolvidas e testadas vão para os clientes.

5.3 Tratamento de bugs

Quando um cliente indica um determinado problema com a aplicação, a equipa de testes entra na instância de testes com a versão do cliente para validar se é um problema de configuração ou um problema com a aplicação. Quando determinado que é um problema com a aplicação, é criado uma tarefa no serviço *devops* onde é guardado o bug, detalhes de reprodução e área do projeto. Quando tal bug é criado, entra para a lista de tarefas onde alguém o irá resolver. Para resolver o problema segue-se os mesmos passos descritos em cima.

5.4 Apoio técnico

Com o intuito de ajudar no apoio técnico, existe uma equipa pronta para ajudar o cliente com qualquer problema. Quando o problema não tem solução simples, a equipa de suporte

comunica com a equipa de desenvolvimento sobre o problema, pois esta tem uma visão mais específica da aplicação. Assim, é possível também analisar o código para ajudar a criar uma solução simples para o cliente.

Para ajudar alguns clientes com algumas configurações do sistema, temos alguns membros do apoio técnico que estão preparados para configurar alguns pontos do eDoc. Temos como por exemplo a ferramenta *FormBuilder*, que alguns clientes ainda têm alguma dificuldade e receio de utilizar. Assim, tendo alguns membros da equipa preparados a configurar formulários, a utilização desta ferramenta por parte dos clientes é maior, sendo positivo pois diminui a necessidade dos clientes de precisarem de páginas customizadas.

5.5 Apresentação do trabalho efetuado

Por vezes, quando é criada uma nova funcionalidade diferente, é nos sugerido fazer uma apresentação interna (para todos os colaboradores da empresa) sobre essa ferramenta. Estas apresentações são bastante positivas, pois por vezes podemos encontrar soluções a problemas que nós poderemos vir a ter, ao mesmo tempo que conseguimos ver potenciais soluções para tais problemas. Também conseguimos ter uma visão mais geral de todos os produtos, algo importante para analisar falhas que possam ter, de uma perspetiva externa ao seu desenvolvimento.

Além destas apresentações entre colaboradores, todos os mestrandos e os estagiários, no final do seu projeto, têm de fazer uma apresentação para a empresa. Nestas podemos encontrar problemas diferentes do usual e tirar algumas ideias sobre funcionalidades a implementar no nosso projeto.

Muito à semelhança de ver outras apresentações, também é bastante positivo mostrar o nosso trabalho. Aqui podemos obter alguns pontos a melhorar, algumas ideias que não tinham sido pensadas e outros usos possíveis para a nossa ferramenta.

6

Estrutura dos projetos eDoc

Neste capitulo irei apresentar um pouco da estrutura dos três projetos principais que constituem o sistema eDoc. Nestes, apresentarei a sua arquitetura e um pouco da ideologia por trás da sua implementação.

Em relação a frameworks a serem utilizadas, o frontend está a ser desenvolvido em angular, enquanto que o backend está a ser desenvolvido em .NET (utilizando a linguagem C#).

6.1 Frontend

O frontend, como dito anteriormente, está dividido em vários módulos. Estes módulos são disponibilizados à aplicação principal através do *Module Federation*, sendo que cada um é um micro-frontend hospedado num URL distinto.

Os micro-frontends são bastante importantes nos dias de hoje, aumentando consideravelmente a qualidade do projeto. Na lista seguinte explico com maior detalhe a forma como os micro-frontends melhoram a produção de novos sistemas.

- **Desenvolvimento independente** - Cada micro-frontend é desenvolvido, testado e compilado independentemente. Isto pode acelerar o ciclo de desenvolvimento, pois as equipas podem trabalhar paralelamente sem interferir no trabalho de outras equipas.
- **Isolamento de falhas** - Os bugs num determinado micro-frontend não perturbará o funcionamento de outros.
- **Melhor escalabilidade das equipas** - Como os micro-frontend podem ser desenvolvidos por equipas diferentes, é mais fácil escalar as equipas e trabalhar em recursos simultaneamente.
- **Tecnologia agnóstica** - Diferentes micro-frontends podem ser escritos utilizando diferentes frameworks ou bibliotecas, permitindo que as equipas escolham a melhor tecnologia para a tarefa em questão. Esta vantagem não está a ser aproveitada de momento pela equipa do eDoc, porém, no futuro, pode haver uma limitação da framework utilizada e que seja preciso utilizar outra para um uso específico.

- **Reuso e composição** - É mais fácil reutilizar um micro-frontend em diferentes partes de uma aplicação.
- **Atualizações incrementais** - Em vez de atualizar uma aplicação inteira, é possível atualizar micro-frontends individuais. Isto aumenta a velocidade com que os clientes recebem correções de problemas, visto que é possível apenas atualizar o micro-frontend com problemas.
- **Performance otimizada** - Como os utilizadores não têm de ler a aplicação inteira para a memória ao iniciar, a performance é significativamente otimizada.
- **Manutenção simplificada** - Com componentes menores e mais focados, a manutenção pode ser mais direta e os riscos associados a alterações são frequentemente reduzidos.
- **Maior facilidade de testes** - Sendo cada micro-frontend atualizado individualmente, fica mais fácil testar a aplicação antes de emitir correções de problemas a clientes.

Para a comunicação com o backend, apenas será utilizada uma interface REST. O código cliente para a API é gerado automaticamente através do *Swagger CodeGen*, que recebe a documentação da API do backend e gera o código. Desta forma, qualquer novo método no backend estará disponível rapidamente e facilmente para uso no frontend.

Para aumentar a velocidade da aplicação, um serviço de cache de pedidos REST está a ser utilizado. Este é guardado numa *indexedDB* no browser. Com isto, qualquer pedido repetido não será enviado para o backend, mas sim retornará o valor que está em cache. A cache também tem um serviço de invalidação de dados, porém este apenas funciona por tempo. Isto é um pequeno problema que está a ser resolvido, pois quando existe uma alteração no backend, será necessário invalidar a cache manualmente. Este problema pode ser inconveniente, pois um utilizador pode ficar à espera dos dados de outro quando este já os introduziu no sistema.

6.1.1 Arquitetura da solução

Como dito anteriormente, o frontend está dividido em vários módulos. Alguns destes não são páginas em si, mas módulos partilhados entre todos os outros com alguma lógica de trabalho. Entre estes módulos temos o "core"e a biblioteca de componentes.

O módulo "core"é um dos módulos mais importantes do eDoc, tendo várias funcionalidades importantes associadas. Entre estas encontra-se toda a comunicação com a API. Para isto, existe uma pasta chamada "API"onde se encontram métodos para todas as entradas da API, que é gerado através do *Swagger CodeGen*. Este funciona de uma forma simples, porém é bastante útil para qualquer projeto *full-stack*. Na fonte do backend temos a documentação da API, que é feita através de comentários especiais nos métodos disponíveis. Estes são transformados num ficheiro *JSON*, sendo este a documentação final do *OpenAPI*. É a partir deste ficheiro que o *Swagger CodeGen* gera as classes cliente para gerir a comunicação com o backend.

Além dos métodos API, também é possível encontrarmos a autenticação e autorização

dentro do módulo "core". Esta é gerida através do *KeyCloak*, algo visto em mais detalhes na secção 8.2.1. As outras funcionalidades disponíveis são a gestão da *cache* cliente, a definição e implementação de alguns métodos auxiliares bastante utilizados no restante da aplicação, enumeradores e classes genéricas e bastantes serviços globais. Dentro destes serviços globais temos, enumerando alguns, a tradução, gestão de erros, configuração de tabelas e gestão do perfil (como imagem e nome do utilizador da sessão).

Além destes módulos partilhados, existe um bastante importante para o funcionamento do eDoc, a *shell*. Esta tem a função de gerir as rotas da aplicação, além de ler e apresentar os micro-frontend disponíveis. Outra funcionalidade da *shell* é a apresentação e gestão dos painéis laterais e painel de navegação.

Os módulos "core"e a biblioteca de componentes serão partilhados com os clientes a partir de um pacote disponibilizado num *package manager*. No momento da realização desta dissertação ainda não ficou definido onde será publicado o pacote (pode ficar num gestor público ou no gestor interno da equipa e publicado com autenticação).

6.1.2 Micro-frontends disponíveis

Dentro dos micro-frontends disponíveis encontram-se o *FrontOffice*, *BackOffice*, *RegisterDocument* e *FormBuilder*. As funções de cada módulo e o seu módulo serão apresentadas na lista seguinte.

- **FrontOffice** — O FrontOffice é o micro-frontend principal, visto que será utilizado por todos os utilizadores. Aqui, podemos utilizar as funções principais do eDoc, como leitura, edição e criação de contextos. Aqui podemos também adicionar documentos, fazendo upload ou utilizando um gerador pré-configurado. Também podemos encontrar neste micro-frontend a página inicial, que é composta por um *dashboard* escolhido pelo utilizador de uma lista disponível.
  Estes dashboards são configurados pelos utilizadores, sendo que para além dos partilhados, cada utilizador pode ter o seu individual. A forma como estes dashboards são editados é a partir de uma grelha com campo *drag-and-drop*, disponibilizando uma lista de campos disponíveis. A maioria destes campo também têm alguma configuração associada, desde o URL nos campos de *iFrame* aos valores dos campos gráficos. Na configuração do próprio dashboard podemos atribuir um nome e a lista de utilizadores associados a este e as suas permissões.
- **BackOffice** — O BackOffice é o micro-frontend de administração. Neste módulo é possível configurar o eDoc, desde adicionar tipos de contexto, administrar permissões de utilizadores, adicionar campos entre muitas outras configurações. Este módulo, por norma, estará apenas disponível a administradores e gestores do sistema, visto que apenas contém configuração global a todos os utilizadores. Este também será um dos módulos mais alterados na implementação desta dissertação, visto que todas as configurações dos tipos de extensão estarão associados ao BackOffice.
- **RegisterDocument** — O RegisterDocument é o micro-frontend que contém a página

para registar documentos externos no eDoc. Por exemplo, quando utilizamos a extensão do eDoc para o Microsoft Office, esta é a página onde estão todas as funções necessárias para a introdução do documento no sistema. Como é uma página bastante utilizada e que está sempre em evolução, ficou decidido que ficaria num micro-frontend extra.

- **FormBuilder** — Este micro-frontend contém toda a extensão do FormBuilder. Aqui será implementada toda a lógica do FormBuilder, ficando, portanto, vazia antes da execução desta tarefa no contexto da dissertação. A estrutura do FormBuilder será vista com mais detalhe na secção 6.3.

6.1.3 Distribuição

A estratégia de distribuição escolhida para o sistema do eDoc baseia-se no uso de *containers*. Esta é uma abordagem moderna que visa garantir portabilidade, escalabilidade e isolamento na implantação da aplicação. Ao optar por *containers*, asseguramos que cada componente do eDoc pode ser executado consistentemente, independentemente do ambiente em que é implementado. [5]

No entanto, é essencial enfatizar que não estamos a falar de um único *container* monolítico. Em vez disso, cada parte da aplicação é encapsulado no seu próprio *container*. Esta abordagem modular permite que cada *container* seja projetado, implementado e otimizado conforme as necessidades específicas da parte da aplicação que encapsula, assegurando assim uma maior eficiência e adaptabilidade.

Quanto ao frontend, cada projeto é compilado, transformando o código-fonte numa versão otimizada para a execução. Após compilados, a hospedagem destes projetos é conduzida com o auxílio do servidor *web Nginx*. Este servidor é conhecido pelo seu alto desempenho, segurança e capacidade de gerir grandes volumes de tráfego. Isto torna-o a escolha ideal para garantir que os utilizadores do eDoc tenham uma experiência suave.

A utilização do servidor *Nginx* [18] também tem a vantagem de poder ser utilizado como *load balancer*. Ao implementar o Nginx, podemos facilmente configurar múltiplos servidores para distribuir a carga do tráfego de rede, otimizando assim a capacidade de resposta e a resiliência da arquitetura. Este característica é particularmente valiosa em cenários de tráfego intenso ou picos inesperados na utilização da aplicação, pois permite que sejam adicionados mais servidores ao conjunto, distribuindo automaticamente os pedidos entre eles. Assim, o *Nginx* não só assegura uma entrega consistente e ágil do conteúdo, mas também proporciona uma arquitetura escalável que pode crescer, adaptando-se às necessidades em constante evolução dos sistemas modernos.

6.1.4 Testes e Garantia de Qualidade

Para garantir uma qualidade que os clientes têm vindo a esperar do eDoc, temos alguns mecanismos para nos apoiar.

Enquanto estamos a desenvolver o código, temos alguns apoios a nível do código. Inicialmente, temos um formatador que melhora a visibilidade do código. Este formatador contém algumas regras, como o número máximo de caracteres numa linha antes de separar essa linha e um validador de *REGEX*. Depois do formatador temos um validador de algumas normas na programação, como forçar uma documentação correta do código e validar que os nomes das variáveis e classes está de acordo com a norma. Como exemplo para nomes de classes nas normas temos que a classe tem de acabar com *Component* quando estende uma classe que indica que é componente. Ambos estes mecanismos impedem a submissão do código enquanto estes tiverem problemas, no entanto, caso haja algum problema em que estes possam estar incorretos, pode sempre ser desabilitado linha a linha.

Para além destes dois mecanismos de apoio ao desenvolvimento, também temos alguns mecanismos de validação de código. Em primeiro lugar encontram-se os testes unitários. Existem testes para todas as funções fulcrais, sendo que algumas de apoio ainda não os contém. Estes testes são executados durante a fase de compilação, sendo que esta é corrida em todos os *Pull-Requests*. Estes testes permitem que, caso haja algum erro que passe todas as outras proteções, consigamos encontrá-lo antes de passar para a fase de testes. Outro mecanismo de prevenção de erros é o uso de um inspetor de qualidade de código.

O último tipo de validação disponível dentro do eDoc são os testes visuais. Nestes testes podemos ver todas as alterações efetuadas em qualquer parte da interface gráfica. Com isto, quando existe qualquer alteração não pretendida ou ao alterar algo num local onde existirem outras alterações, somos avisados e podemos validar se todas as alterações são intencionais ou se há algo que ficou errado inadvertidamente. Desta forma, ao submeter o código e depois de validar todas as alterações, diminuímos consideravelmente a probabilidade de haver alguma alteração não intencional ou algum erro gráfico.

6.1.5 Documentação pública

Para o apoio à utilização de alguns módulos partilhados (principalmente o *Core* e a biblioteca de componentes, alguma documentação é partilhada com os clientes. O módulo *Core* tem documentação escrita no próprio JavaScript. Já a biblioteca de componentes tem uma documentação visual bastante mais interessante.

Para a criação desta documentação, foi utilizado uma biblioteca chamada *StoryBook* [20]. Com isto, podemos mostrar aos clientes todos os componentes disponíveis, ao mesmo tempo que apresentamos um exemplo destes. Dentro de cada componente também podemos encontrar alguma documentação sobre os seus parâmetros, tal como podemos editar os valores padrão e ver as alterações apresentadas em tempo real no componente de exemplo. Esta ferramenta é bastante útil pois dá ao cliente uma visão mais geral de todos os componentes existentes no eDoc, ao mesmo tempo que permite testar alguns parâmetros e algumas funcionalidades de cada um.

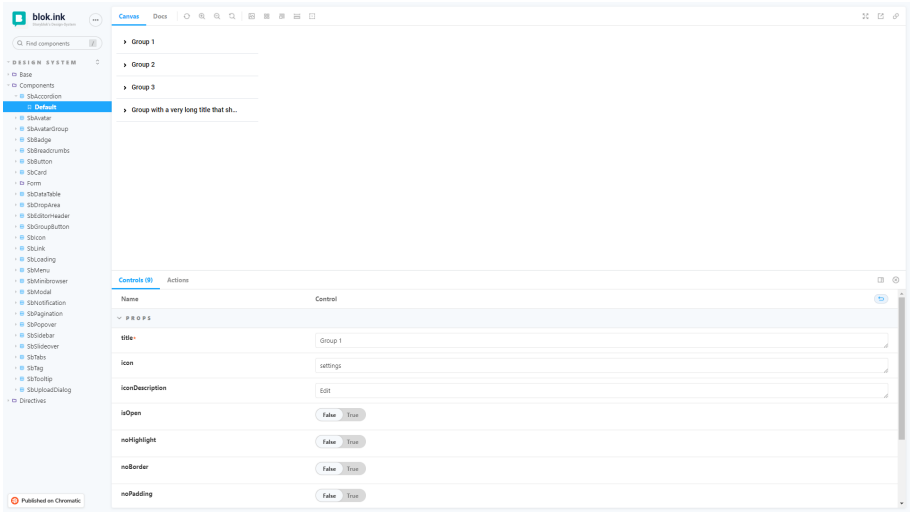

Figura 6.1: Exemplo de uma página de documentação StoryBook [8]

6.2 Backend

À semelhança dos outros projetos, o backend também está dividido em módulos. Estes módulos são projetos distintos, no entanto encontram-se na mesma solução.

Dentro do backend existem os seguintes módulos:

- **WebAPI** - Este é o maior projeto em número de ficheiros, pelo que neste encontra-se uma grande parte das funcionalidades do projeto. Neste módulo encontram-se tudo o necessário para a ligação do sistema eDoc com o exterior. Isso inclui:
  - **DTOs** - Os DTOs são classes que traduzem o tipo de dados interno do eDoc para o utilizado pela API. Por exemplo, se tivermos os dados *username*, *password* e nome na base de dados, o DTO apenas guardaria os dados *username* e nome, pelo que a *password* não deveria ser incluída na resposta da API. À semelhança do exemplo dado, existem DTOs tanto para entrada como para saída, ou seja, existem DTOs que traduzem as respostas da API e outros que traduzem os pedidos.
  - **Exceções** - Tal como nos DTOs, estas exceções são uma tradução das exceções internas para exceções que podem ser enviadas para os clientes. Para além destas não conterem nenhuns dados privados, também têm um código público que pode apoiar o cliente a diagnosticar algum problema que possa ter.
  - **API interna e pública** - Para a comunicação do servidor com outros serviços, são disponibilizadas duas APIs, uma interna e outra pública. A pública contém todos os métodos que possam ser necessários para os clientes utilizarem todas as funções disponibilizadas pelo eDoc. Já a API interna contém os métodos necessários para a equipa do eDoc poder depurar alguma falha que esteja a acontecer com o sistema.
- **Core** - O módulo core inclui todos os dados necessários para a comunicação do servidor com a base de dados. Estes terão os mesmos campos que são necessários

para o uso dos procedimentos criados na base de dados.

O core, por sua vez, encontra-se dividido em dois sub-módulos. Estes são:

  - **Application** - Aqui encontram-se todos os tipos de dados necessários para o uso da aplicação, ou seja, tipos de dados para todas as funções que os clientes possam necessitar.
  - **Domain** - Neste sub-módulo estão guardados todos os tipos de dados necessários para a administração do sistema eDoc por parte da equipa interna.

- **Infrastructure** - Neste projeto encontram-se alguns utilitários para o bom funcionamento do servidor. Existem três tipos de utilitários disponíveis neste projeto, os de cache, leitura e escrita de documentos e os de registo de eventos e dados.
- **Infrastructure.SQLServer** - Neste projeto encontram-se todos os mecanismos de comunicação do eDoc com a base de dados. Dentro destas *queries* encontram-se procedimentos, transações, *logs* e repositórios.
- **SDK** - Toda a comunicação entre as APIs e os módulos intermédios é efetuada pelo SDK. Aqui, podemos encontrar todas as funções que podem ser necessárias mais do que uma vez. Temos como exemplos a leitura e escrita de documentos, fluxos e etapas.
- **Shared** - Tal como no frontend, existem alguns serviços e utilitários que são necessários em todos os módulos. Com esta finalidade, foi criado este módulo, um módulo partilhado por todos.

Outra funcionalidade presente no backend são as regras. Estas são pequenas funções, criadas através de uma interface, que permitem controlar o eDoc automaticamente sem o recurso a extensões. Um exemplo de uma regra é atualizar o campo furação de acordo com a data de início e de fim.

6.3 FormBuilder

Durante a implementação desta dissertação, o FormBuilder será apenas adaptado para a nova versão do eDoc. Assim sendo, a arquitetura e estrutura deste sistema na nova versão será bastante similar ao que está de momento em versões de clientes.

Dito isto, o FormBuilder está dividido em alguns módulos, sendo estes o FormBuilder, FormRenderer, Form, FormValidation e componentes partilhados. Cada módulo tem uma função fulcral no bom funcionamento desta aplicação, no entanto apenas alguns são partilhados com outros módulos. Com isto, podemos ver na figura 6.3 que o FormBuilder e o FormRenderer não são dependências de mais nenhum módulo. Isto é esperado, visto que ambos representam a lógica de apresentação do formulário ao utilizador.

O FormBuilder contém toda a lógica para edição do formulário. Isto inclui lógica, propriedades de campos, definição de ações, adições e remoção de campos e alteração de ordem e descendência de campos. Este último representa a função do FormBuilder onde alguns campos podem conter campos dentro destes. Estes campos são o Fieldset (um conjunto de campos dentro de uma "caixa", que também pode conter um título) e

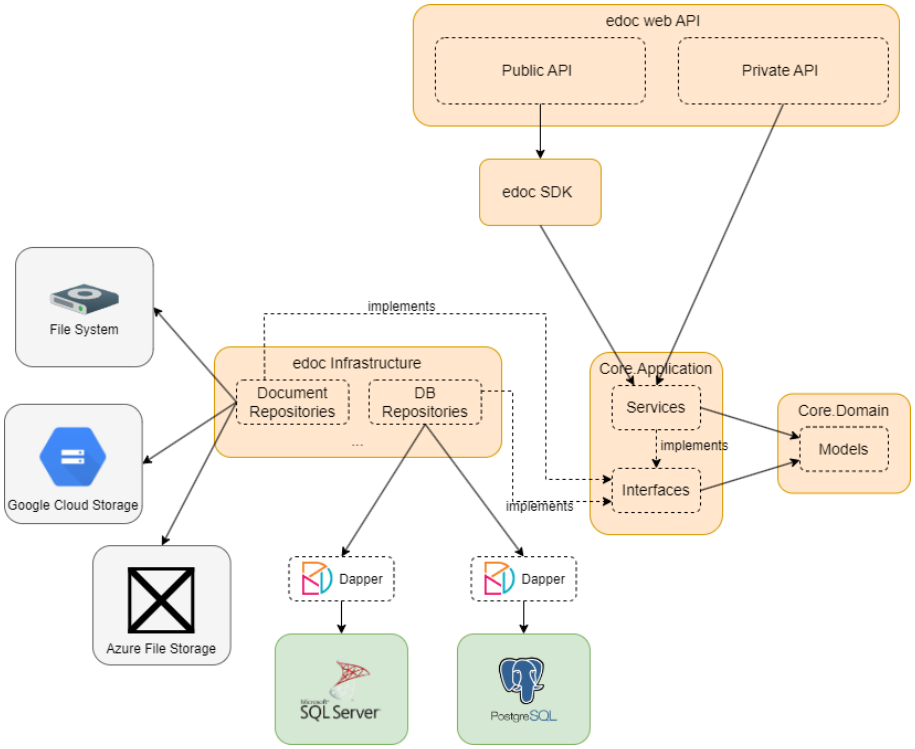

Figura 6.2: Arquitetura do backend do eDoc

colunas. Todos os campos serão vistos com mais detalhes na subsecção 6.3.1. Por outro lado, o FormRenderer contém apenas a lógica de apresentação do formulário e execução de lógica e ações.

Para o bom funcionamento destes dois módulos precisamos de uma boa base de funções e componentes. Aqui podemos encontrar os módulos Form, FormValidation e componentes partilhados. Os componentes partilhados são bastante simples, contendo alguns métodos globais e os serviços necessários para o funcionamento da aplicação. Dentro destes serviços podemos encontrar, por exemplo, a comunicação com o backend, a transformação das respostas do eDoc para valores do FormBuilder. O módulo Form contém todos os campos disponíveis e o módulo FormValidation contém toda a lógica para apresentação de erros na validação de dados dos campos. Esta validação é definida pelo cliente, que pode conter desde comprimento mínimo de caracteres a valor mínimo em campos numéricos.

6.3.1 Campos disponíveis

De seguida encontra-se uma lista com os campos do eDoc e uma pequena descrição da função destes.

- **Booleano** - Campo booleano com 3 valores (Sim, Não ou Sem valor).
- **Botão** - Campo de suporte para executar ações. Estas ações podem ser relacionadas com o edoc (manipulação de contextos) ou com o próprio FormBuilder (execução de lógica ou abertura de nova página).

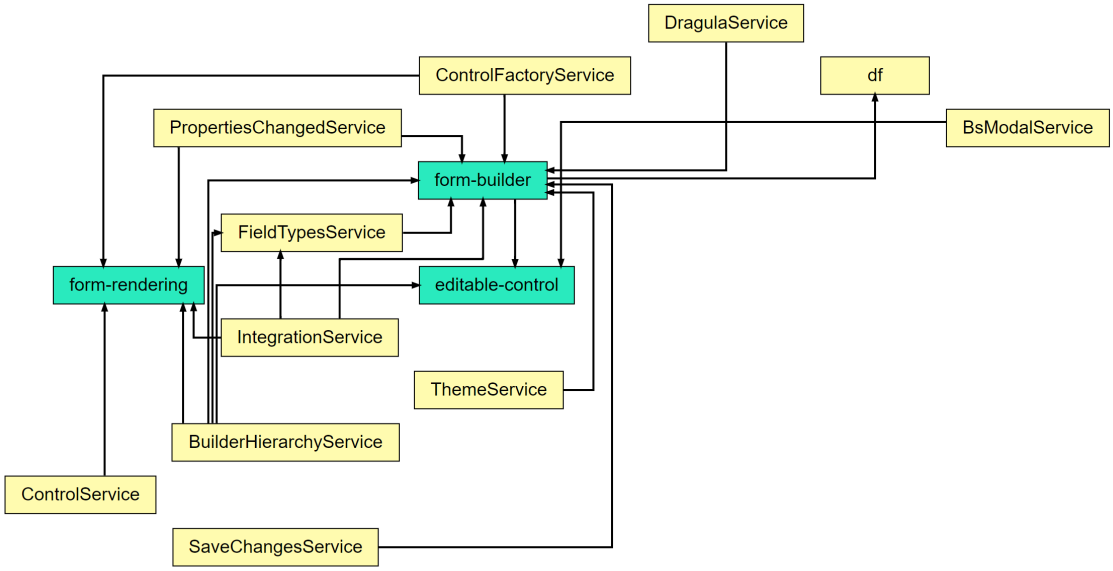

Figura 6.3: Gráfico de dependências do FormBuilder

- **Colunas** - Campo de suporte. Permite colocar vários campos em colunas (lado a lado).
- **Data** - Campo que permite editar e visualizar campos do tipo data. Contém um popup com um calendário.
- **Dropdown** - Campo do tipo lista que contém uma caixa de texto no dropdown para permitir pesquisas.
- **Entidade** - Campo que permite editar as entidades atribuídas a um determinado contexto. Entidades são utilizadores ou grupos.
- **Fieldset** - Campo de suporte que permite colocar vários campos numa "caixa". Este pode conter um título, o que permite uma melhor separação de campos de um formulário. Também é utilizado para esconder vários campos com apenas um comando.
- **Upload de ficheiros** - Campo para enviar ficheiros para o eDoc. Apenas utilizado para ações, sendo que não guarda valores.
- **Texto** - Campo de suporte. Apresenta um dado texto com um dado tamanho. Todos os atributos do texto podem ser alterados.
- **Lista** - Campos do tipo lista. Apresenta vários elementos numa dropdown. De seguida enumero os tipos de listas disponíveis.
  - **Livros** - Campo que lista todos os livros (tipos de registos) disponíveis no eDoc.
  - **Classificações** - Campo que lista todas as classificações (tipos de fluxos) disponíveis no eDoc.
  - **Tipos de distribuições** - Campo que lista todos os tipos de distribuições disponíveis no eDoc.
  - **Lista** - Lista simples. Recebe uma lista de elementos, separados por vírgula. Tais elementos são simples *strings* sem nenhum parâmetro adicional.

  - **Referência** - Lista simples. Recebe uma lista de elementos, sendo que estes contém chave, valor e estado de ativação. Este "estado de ativação"indica se o valor é visível e selecionável.
  - **Utilizador ou grupo** - Campo que lista todos os utilizadores e/ou grupos disponíveis no eDoc.
  - **Tabelado** - Lista simples. Recebe uma lista de elementos, sendo que estes contém chave, valor, data de início e data de fim. Estas datas indicam entre que datas o campo se encontra ativo.
- **Dropdown de multi-seleção** - Igualmente ao campo de seleção, apresenta uma lista com uma caixa de texto no dropdown que permite pesquisar a lista. No entanto, cada elemento contém uma caixa de seleção que permite a seleção múltipla de elementos da lista.
- **Tabela** - Campo que permite visualizar uma tabela avançada. As tabelas têm de ter um esquema definido. Esta é uma lista de colunas e sub-tabelas. Estas "sub-tabelas"é definido como um campo de cada linha da tabela que é apresentado como outra tabela. As sub-tabelas também são definidas com um esquema com lista de colunas e sub-tabelas. As colunas contêm tipo de dados, valor padrão, chave do campo, título da coluna, visível e editável.
- **Área de texto** - Campo com uma área de texto. Esta é um *WYSIWYG*, ou *What You See Is What You Get*. Atualmente está a ser utilizada a biblioteca *TRUMBOWYG* [28] para este efeito no FormBuilder. Isto permite textos mais ricos, que podem conter imagens, listas, itálicos, tamanhos diferentes em diferentes partes do texto, negrito, inserção de HTML e outras funcionalidades. Outra possibilidade é colar partes de um ficheiro *Microsoft Word* e manter a formatação.
- **Caixa de texto** - Campo com uma simples caixa de texto.
- **Hora** - Campo que permite edição e visualização de valores do tipo hora. Contém duas caixas de texto numéricas com setas verticais para editar as horas e minutos apenas com o rato.
- **Árvore** - Campo lista que permite ter uma hierarquia entre elementos. Apenas os elementos na raiz são apresentados inicialmente, porém o utilizador pode ir expandido até chegar ao valor pretendido.
- **URL** - Campo de suporte que contém um texto com uma hiperligação.

7

Estrutura da solução

Neste capítulo irei falar sobre a estrutura definida para cada tipo de extensão, ao mesmo tempo que apresentar os passos encontrados para resolver cada uma. Também irei apresentar algumas falhas de segurança possíveis com alguns tipos de extensão e apresentar as soluções encontradas para as resolver.

Dada a complexidade e a multifacetada natureza da extensibilidade no contexto do eDoc, optei por tratar cada tipo de extensão como um micro-projeto autónomo. Esta abordagem reflete a realidade da implementação destas extensões, pois cada uma representa um conjunto único de desafios e requer soluções específicas. Para refletir isso na estrutura da minha dissertação, decidi separar a implementação de cada extensão em subcapítulos distintos. Este formato permite uma exploração mais aprofundada e precisa de cada extensão, facilitando a compreensão de como cada uma contribui para a extensibilidade global do eDoc. Acredito que essa estrutura também facilitará o leitor a seguir o raciocínio e a progressão do meu trabalho.

7.1 edocHook

Durante a fase de desenho da arquitetura surgiu a necessidade de dar um nome comercial ao conceito de enviar WebHooks para clientes, ficando, portanto, decidido que se chamará edocHook. Assim sendo, a partir deste ponto será utilizada a nova designação.

A implementação deste tipo de extensão foi dividida em frontend e backend. A parte do frontend, como descrito antes, não pode ser finalizada dado o facto do estado atual do novo eDoc ainda não permitir a implementação desta funcionalidade. Este facto deve-se a atrasos no desenvolvimento do projeto to eDoc, sendo que a implementação da tese não teve nenhum impacto neste atraso. No entanto, foi criado um micro-projeto onde foi apresentada a arquitetura da solução, mostrando um pouco como será implementado na versão final. Este projeto serviu como prova de conceito para facilitar a futura implementação no eDoc.

7.1.1 Arquitetura da solução

Sendo este um projeto full-stack, a arquitetura foi concebida de forma a ser dividida em dois componentes principais: frontend e backend. Esta estrutura permite uma separação clara das responsabilidades e facilita o processo de desenvolvimento e manutenção. O frontend é responsável pela interação direta com o utilizador, enquanto que o backend se encarrega da lógica de negócio e do processamento de dados. Esta divisão funcional proporciona uma organização eficiente do código, bem como uma maior flexibilidade na implementação de novas funcionalidades e melhorias.

No backend, o envio e tratamento dos WebHooks foram implementados através de uma classe estática, tendo apenas 2 métodos. Para além do envio dos WebHooks, a configuração ficou guardada na base de dados e é gerida através de uma API.

Para a leitura da configuração internamente (dentro do código do servidor) dos edocHooks, foram adicionados alguns métodos ao SDK. Estes são utilizados maioritariamente para ler os edocHooks existentes de um determinado tenant e os seus eventos subscritos. A lógica para a execução do edocHook vai ser analisada no próximo capítulo.

Dando continuidade à nossa análise, chegamos aos WebHooks enviados pelos edocHooks, cujo conteúdo é composto por três partes. Este arranjo tripartido é crucial para o funcionamento e eficácia dos WebHooks e é onde a nossa atenção se concentra agora.

Primeiramente encontramos o identificador do contexto. Esta parte é fundamental, visto que é aqui que se encontra a origem do evento. Para ilustrar, se nos encontrarmos num evento de etapa, o identificador associado a essa etapa específica é enviado para o cliente.

De seguida temos o tipo de contexto. Este apoia o identificador, indicando ao recetor do edocHook qual o tipo de contexto ao qual o identificador está associado.

Finalmente temos o token de autenticação. Enviando um token temporário para o recetor do edocHook permite que este tenha acesso à leitura e alteração de dados, algo fundamental visto que não são enviados quaisquer dados importantes relativos ao contexto onde foi emitido o evento.

A nossa opção de não enviar todo o contexto vem da multitude de dados associados a cada contexto, sendo que o servidor não tem como decidir o que o recetor necessita. Desta forma, ao enviarmos o identificador, o recetor pode pedir os dados que necessita, facilitando assim a transferência de dados entre o servidor e o recetor.

Para exemplificar um edocHook, temos a submissão de uma etapa. Quando essa etapa é submetida, o servidor envia para o recetor o token, identificador da etapa e emite que o tipo de contexto é etapa. De seguida, o recetor pede ao servidor (utilizando o token temporário enviado no WebHook) os campos da etapa atual, computa um outro campo (como a diferença entre dois campos de data) e faz um pedido POST ao servidor para atualizar um campo na etapa seguinte com esse valor. O fluxo é mostrado como diagrama na figura 7.1.

Algo que também é preciso ter em consideração para a fase da implementação é o

modelo de execução, havendo então a escolha entre a execução síncrona ou assíncrona. Este campo é visto mais em detalhe na secção 7.1.2.

Para além destes campos, que serão guardados numa tabela específica, também teremos os eventos subscritos. Neste caso, um edocHook pode estar subscrito a eventos referentes a um tipo de contexto. Desta forma, um cliente pode subscrever, por exemplo, apenas ao término de uma tarefa específica, o que diminui o uso de recursos tanto do servidor do eDoc como do servidor para o qual é enviado o WebHook.

Os eventos subscritos serão guardados em algumas tabelas extra, sendo que cada tipo de contexto terá uma tabela com os eventos subscritos dos seus elementos. Temos como exemplo os fluxos, que terá uma tabela com os eventos subscritos entre todos os fluxos, que terá uma coluna com o tipo de fluxo, evento subscrito (como envio de etapas e criação de fluxo) e edocHook referente ao evento.

A configuração dos edocHook consiste em 5 campos, sendo estes guardados na base de dados, sendo estes:

- URL - Este campo representa o URL de destino do WebHook. Este é para onde serão enviados os dados quando emitido um evento correspondente a este edocHook.
- Descrição - Este campo contém uma descrição do edocHook de modo a facilitar a pesquisa, leitura e gestão dos WebHooks por parte do cliente.
- Tipo de edocHook - Este campo representa o tipo de eventos possíveis para este edocHook. Temos como exemplo de tipos de edocHook documentos, fluxos e pastas.
- Ativo - Este campo indica se o edocHook está ativo. Útil para ajudar o cliente a desativar uma regra (para testes ou por alguma falha externa) sem apagar a regra, de modo a que quando voltar a ser necessário não se tenha de criar outra vez.
- Modo de execução - Este campo indica o modo de execução do edocHook (pode ser síncrono ou assíncrono). Esta parte vai ser vista mais em detalhe na secção 7.1.2.

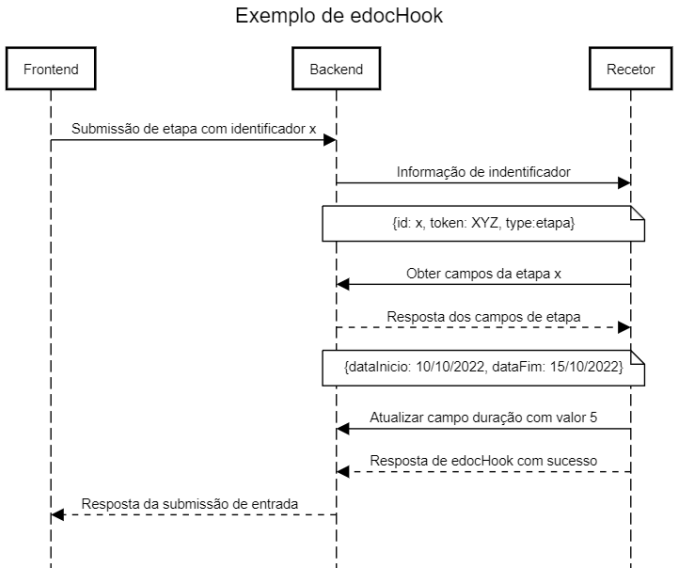

Figura 7.1: Diagrama com exemplo de fluxo de um edocHook

Dentro do frontend terão de ser criadas 2 páginas, sendo estas:

- **Página de listagem de edocHooks** - Esta página lista os edocHooks já criados pelo cliente, tendo também uma pesquisa simples em tipo de edocHook, ativo, e descrição.
- **Página de edição de edocHook** - Esta página permite a edição de um edocHook. Todos os campos podem ser editados exceto os eventos subscritos.

Para além das páginas, também terão de ser criados alguns componentes. Nestes componentes temos:

- **Campo de subscrição de evento** - Este campo permite ao cliente subscrever a um evento. Este campo irá encontrar-se nas páginas de configuração de tipos de contexto.
- **Lista de edocHooks subscritos** - Este campo lista todos os edocHooks subscritos a um determinado evento.
- **Lista de edocHooks existentes** - Este campo lista todos os edocHooks existentes. Utilizado na página de listagem de edocHooks.

Algo de se notar nas listas anteriores é a omissão da página de configuração dos eventos subscritos. Este facto deve-se à utilização da página de configuração de tipos de contexto para este uso. Esta página já tem um campo para eventos subscritos, que atualmente apenas pode ser utilizado com regras. Assim, para a configuração dos eventos subscritos de um tipo de contexto apenas será necessário introduzi-los nos campos já existentes.

Em conjunto com os campos e com as páginas, também será necessário criar uma classe com a comunicação entre o frontend e o backend. Esta será gerada através de um gerador de código, tendo como entrada o ficheiro de especificação OpenAPI do backend. No entanto, será necessário ligar essa classe ao restante do código.

7.1.2 Modos de execução

Existem dois modos diferentes de execução: síncrona e assíncrona.

7.1.2.1 Execução síncrona

Execução síncrona do edocHook funciona através de uma sequência linear de operações. Essencialmente, os pedidos seguem um protocolo "pedido-resposta". Quando um cliente envia um pedido POST para o servidor, o servidor processa o pedido, envia o WebHook e espera pela resposta do WebHook. Esta sequência não é completa até que o servidor receba a resposta do WebHook ou um timeout ocorra. O servidor envia então a resposta para o cliente, enviando também uma informação sobre o sucesso do edocHook.

Esta arquitetura espelha as operações tradicionais síncronas, onde uma operação (neste caso o pedido POST) fica bloqueada até que uma operação que chamou (WebHook) termine.

Há várias vantagens para esta abordagem:

- **Resultado imediato** - Uma vantagem chave dos edocHooks síncronos é o resultado imediato que eles proporcionam. Qualquer alteração de dados feita por um edocHook fica imediatamente visível para o utilizador, o que melhora a experiência do cliente. Por exemplo, se um cliente terminar uma etapa e que parte dos dados dessa etapa são usados para computar um campo da etapa seguinte, utilizando um edocHook síncrono essa alteração fica imediatamente visível após a página ler, não sendo necessário o cliente atualizar a página.
- **Simplificação do tratamento de erros** - Numa operação síncrona, os erros podem ser capturados e transmitidos ao cliente imediatamente. Se o edocHook encontrar um problema, o servidor terá essa informação antes de responder ao cliente, podendo portanto adicionar essa informação na resposta do pedido do cliente. Isto pode simplificar o processo de tratamento de erros, visto que simplifica a depuração do sistema para encontrar a origem do erro. Voltando portanto ao exemplo anterior, após o cliente terminar uma etapa e a etapa seguinte não ter os dados corretos, o cliente pode facilmente saber se o erro veio do edocHook.
- **Conservação da sequência** - EdocHooks síncronos preservam a sequência de operações, o que pode ser importante em situações em que operações subsequentes dependem dos resultados dos edocHooks.

No entanto, os edocHooks síncronos também têm desvantagens:

- **Possibilidade de bloqueio** - Se o edocHook demorar muito tempo para processar ou se houver uma latência de rede alta, o servidor pode ficar bloqueado, piorando o desempenho para outros pedidos que possam ocorrer no entanto. Isto pode degradar o desempenho do sistema e resultar numa pior experiência para o utilizador.
- **Intensivo em recursos** - Operações síncronas normalmente exigem mais recursos, visto que mantém as conexões abertas enquanto se aguarda pela resposta do edocHook.
- **Risco de timeouts** - Se o servidor do WebHook for lento ou não responder, timeouts podem ocorrer, potencialmente interrompendo o fluxo de operações.

Outra vantagem dos edocHooks síncronos é a possibilidade de estes retornarem valores que poderão ser utilizados nas regras. Esta funcionalidade apenas está pensada, mas caso seja aprovada permitirá que os clientes utilizem os edocHooks de forma mais simples, sem ser necessário chamar a API para alterar alguns dados. Estes podem vir como resultado do próprio WebHook, sendo que depois o próprio backend trata da atualização de dados de acordo com regras personalizadas criadas pelos próprios clientes.

Em conclusão, edocHooks síncronos proporcionam resultados imediatos, simplificam o tratamento de erros e preservam a sequência de operações, o que pode ser valioso para determinados tipos de operações. No entanto, eles também podem apresentar problemas

de desempenho e exigir mais recursos do que WebHooks assíncronos. Apenas devem ser utilizados quando os benefícios são fulcrais para a sua execução de modo a não piorar o desempenho do resto da aplicação.

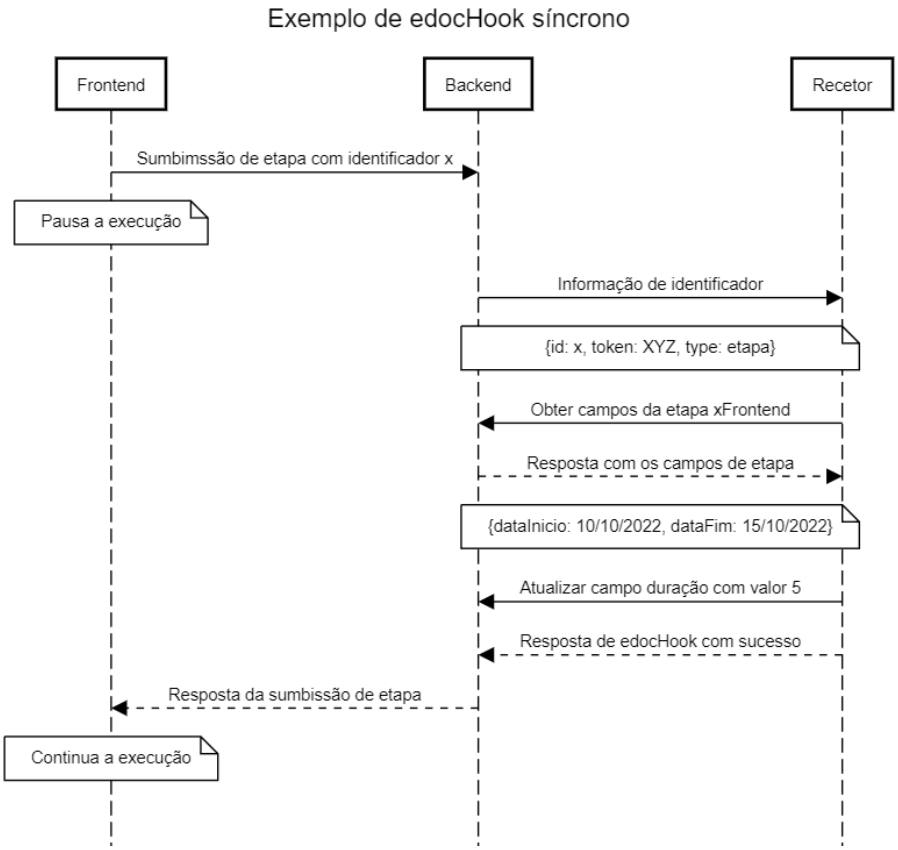

Figura 7.2: Diagrama de execução de um edocHook síncrono

Como podemos observar na figura 7.2, o frontend para a execução após a submissão da etapa com o identificador x. De seguida, o backend envia um edocHook para o recetor e espera que este responda a este pedido. Durante esta fase, o recetor poderá fazer pedidos à API do eDoc. Finalmente, quando o backend recebe a resposta de sucesso do edocHook, este conclui o pedido à API pelo frontend, sendo que a partir deste momento a execução deste continua.

7.1.2.2 Execução assíncrona

Os edocHooks assíncronos operam sob um paradigma diferente dos seus equivalentes síncronos, seguindo um modelo orientado a eventos. Quando um cliente faz um pedido POST para um servidor, este emite o WebHook e envia imediatamente a resposta para o cliente. Neste caso, o servidor não espera a resposta do WebHook, apenas registando se o seu envio foi feito com sucesso ou não.

Este modelo está alinhado com o conceito de operações assíncronas, onde uma tarefa é iniciada e executada independentemente sem bloquear o progresso de outras tarefas.

Os benefícios dos edocHooks assíncronos incluem:

- **Melhoria de responsividade do sistema** - EdocHooks assíncronos podem melhorar significativamente a responsividade de um sistema. O servidor não precisa de esperar pela resposta do WebHook, o que significa que pode continuar a processar outras solicitações recebidas. Isto pode ser um benefício em cenários de alta carga, onde o sistema precisa de lidar com um grande número de pedidos simultaneamente.
- **Melhor experiência do utilizador** - Para operações que possam levar muito tempo a serem processadas ou que dependem de serviços de terceiros com tempos de resposta imprevisíveis, edocHooks assíncronos podem proporcionar uma melhor experiência de utilizador, não bloqueando a aplicação cliente.
- **Maior taxa de transferência** - Operações assíncronas podem processar mais pedidos num determinado período de tempo quando comparado com operações síncronas. Isso acontece pois o servidor não fica bloqueado à espera das respostas dos WebHooks e pode continuar a aceitar novos pedidos.

No entanto, os edocHooks assíncronos também apresentam os seus próprios desafios:

- **Complexidade no tratamento das respostas** - Como as respostas dos WebHooks podem chegar a qualquer momento e em qualquer ordem, lidar com essas respostas e associá-las aos pedidos originais pode adicionar complexidade no sistema. Mesmo as respostas sendo uma verificação de receção (utilizando um identificador para associar uma resposta ao seu pedido), o seu tratamento precisa de passos adicionais quando comparado com um sistema síncrono.
- **Informação de erros atrasada/não enviada** - As informações de erro por parte dos edocHooks não são enviadas para os clientes, sendo que caso haja algum problema o utilizador não é informado. Tais informações também são guardadas de forma atrasada, sendo que caso haja erros, estes não ficam imediatamente disponíveis para o cliente.
- **Possíveis problemas de consistência de dados** - Se não forem geridos com cuidado, o uso de edocHooks assíncronos pode levar a problemas de consistência de dados, especialmente em casos em que as operações são interdependentes. Este fator é menos importante no eDoc, apesar de em certas situações poder fazer a diferença.

Em resumo, edocHooks assíncronos oferecem vantagens em termos de responsividade do sistema, taxa de transferência e experiência do utilizador, especialmente em situações de alta carga, alta latência ou dependentes de serviços de terceiros. No entanto, eles introduzem complexidade adicional no tratamento e processamento das respostas, assim como no tratamento de erros e consistência de dados. Assim sendo, o seu uso é mais adequado para situações em que os benefícios de alta responsividade e melhor experiência do utilizador superam a necessidade de feedback imediato.

Como podemos observar na figura 7.3, quando o frontend envia o pedido à API com a submissão da etapa com o identificador x, este para a execução e espera uma resposta ao pedido. De seguida, o backend trata deste pedido e, após concluir, envia a resposta ao pedido do frontend, sendo que este continua a execução. Após esta resposta, o backend

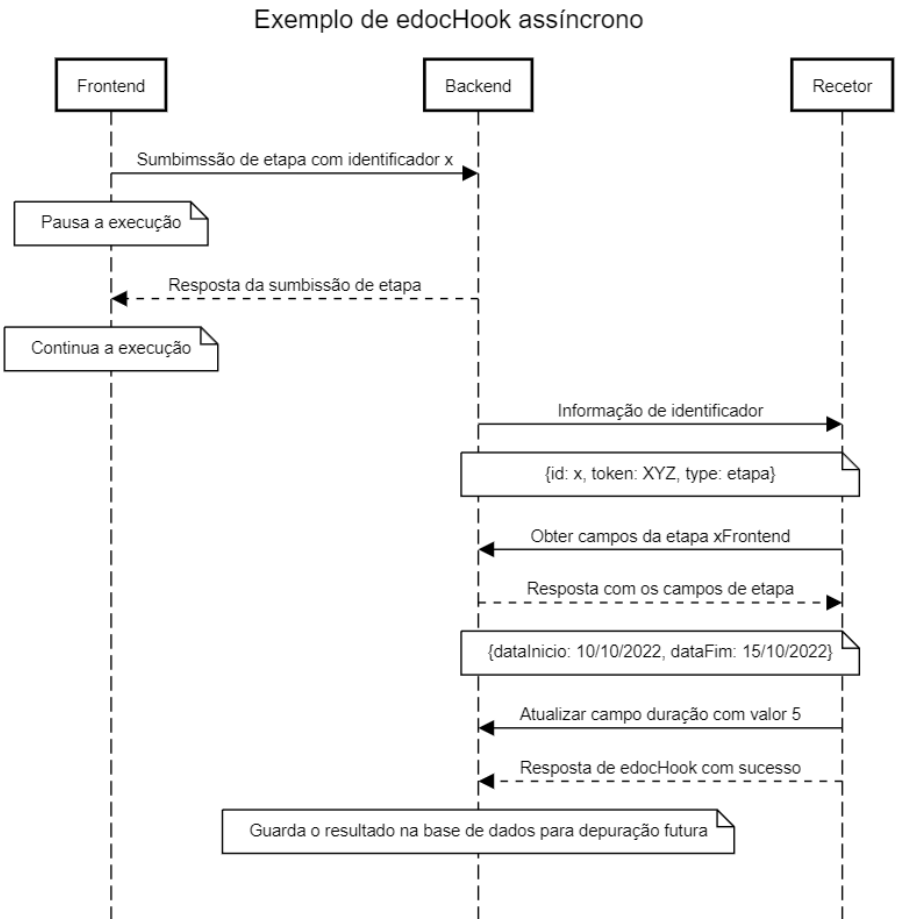

Figura 7.3: Diagrama de execução de um edocHook assíncrono

envia todos os dados ao recetor do edocHook, sendo que este chama a API enquanto este pedido está aberto e, no final, responde ao backend com uma informação de sucesso.

7.1.2.3 Diretrizes para escolha entre edocHooks síncronos e assíncronos

Embora tanto os edocHooks síncronos e assíncronos oferecem vantagens únicas, é a natureza e a exigência da tarefa específica em questão que devem orientar a decisão entre os dois.

- **Requisitos de feedback imediato** - EdocHooks síncronos devem ser escolhidos quando a operação requer feedback imediato. Isto é crucial em instâncias onde a ação do utilizador requer uma resposta direta e imediata. Repetindo o exemplo anterior, se um cliente terminar uma etapa e que parte dos dados dessa etapa são usados para computar um campo da etapa seguinte, será necessário utilizar um edocHook síncrono, visto que necessitamos que essa alteração fique imediatamente visível, sem a necessidade de o utilizador atualizar a página.
- **Consistência de dados** - EdocHooks assíncronos não prometem consistência de dados. Nos casos em que a ordem dos edocHooks tem de ser consistente, deve ser utilizado edocHooks síncronos.

- **Tolerância a latências altas** - Utilizando edocHooks assíncronos, se houver latência alta, essa latência não será transmitida para o utilizador, diminuindo o tempo de espera na aplicação cliente. Isto melhora muito a experiência do utilizador.
- **Alta utilização de recursos** - Visto que edocHooks síncronos consomem mais recursos que os assíncronos, a sua utilização deve ser reduzida o máximo possível.

Tendo em conta estas diretrizes, podemos verificar que o uso de edocHooks assíncronos é sempre preferível, sendo apenas recomendado o uso de edocHooks síncronos quando é necessário feedback imediato e/ou consistência de dados. Por este motivo ficou definido que os edocHooks assíncronos seriam definidos como padrão, sendo necessário colocar manualmente a opção de modo de execução síncrono para o alterar.

7.1.3 Segurança com os edocHooks

Sendo este um projeto em cloud, a segurança é algo bastante importante.

Aquando do uso de WebHooks, os principais ataques com que nos temos de preocupar são ataques do tipo homem no meio, falsificação de pedidos e ataques de repetição [31].

Ataques homem no meio é um tipo de ataque em que os dados trocados entre as duas partes (neste caso o servidor do eDoc e o recetor do edocHook) são de alguma forma intercetados. Depois de intercetados, o atacante pode guardar tais dados e, em certos casos, alterá-los antes de os transmitir para o recetor sem que nenhuma das partes tenha conhecimento. Este ataque é perigoso em 2 frentes. Primeiro, todos os dados podem ser guardados pelo atacante, o que compromete a privacidade dos dados. Por fim, os dados podem ser alterados, sendo que em algum caso crítico pode causar problemas graves. Por exemplo, se o cliente criar uma encomenda tendo em conta os dados de uma etapa, tais dados podem ser alterados de modo a ser criada uma encomenda com produtos que não têm interesse ao cliente, causando problemas financeiros (caso o cliente não consiga devolver) e problemas de recursos, pois o cliente tem de colocar recursos para resolver esse problema.

Uma solução para este tipo de ataques será encriptar todos os dados entre o servidor e o recetor. Esta solução não resolve o problema de haver alguém à escuta do pedido e intercetá-lo, mas neste caso o atacante já não conseguirá ler nem alterar os dados presentes no pedido. Uma forma simples de cumprir com este requisito é criar o requisito de que o cliente utilize o protocolo HTTPS em vez do HTTP. Este protocolo é encriptado utilizando a Camada de Transporte Segura (TLS), ou utilizando a nomenclatura antiga, Camada Segura de Sockets (SSL) [14].

Ataques de falsificação de pedidos são ataques em que um atacante envia um pedido para o recetor do edocHook em nome do edoclink. Os problemas com este ataque são em parte iguais aos de ataque do tipo homem no meio, visto que em ambos os ataques o problema está no envio para o cliente de dados errados ou alterados. Este tipo de ataque não compromete a privacidade dos dados, no entanto cria os problemas em que o cliente pode tomar decisões erradas por ter obtido dados incorretos.

Para este tipo de ataques existem várias soluções. Uma delas, sendo a mais simples para nós (eDoc) a criação de uma *whitelist* no recetor do cliente. Isto obriga a que todos os pedidos referentes aos edocHooks tenham origem no *IP* do backend do eDoc. O problema com esta solução é que esta não é executória, ou seja, não é possível que o backend do eDoc valide se tal *whitelist* está a funcionar e apenas tem os *IPs* de servidores fiáveis. Outra solução, esta mais complexa de se criar, é de enviar em todos os pedidos uma assinatura dos dados através de um par de chaves públicas e privadas. Deste modo, em conjunto com o pedido será enviada uma assinatura dos dados criada com uma chave privada, sendo que qualquer entidade pode validar se o pedido teve origem no servidor do eDoc pois, através da chave pública (sendo esta partilhada anteriormente), pode validar se tal assinatura foi assinada com a nossa chave privada. Sendo que ninguém terá acesso à nossa chave privada, a única forma de a assinatura ser validada com a chave pública é se esta tiver sido encriptada com o uso da chave privada.

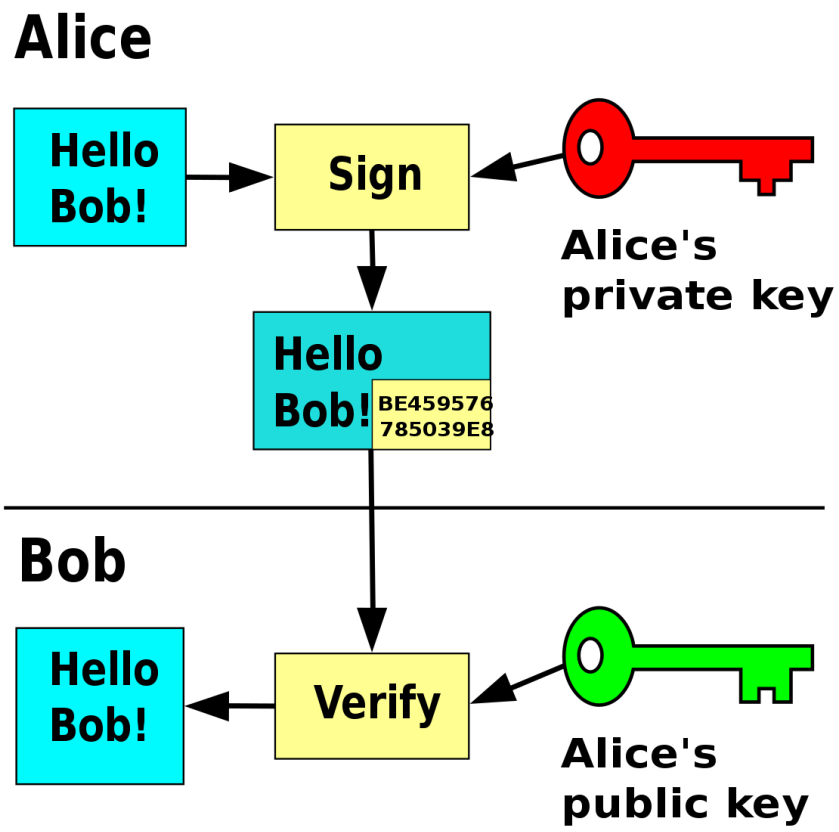

Figura 7.4: Como funciona uma assinatura [10]

Para finalizar, temos os ataques do tipo de repetição. Estes ataques são bastante similares aos ataques do tipo homem no meio, porém estes também funcionam em ambientes encriptados. A forma de uso é similar aos do tipo homem do meio, ou seja, um atacante interceta o comunicação entre o servidor e o recetor. No entanto, a principal diferença é que ataques do tipo homem no meio pretendem guardar e/ou editar os dados,

um ataque de repetição apenas está interessado em repetir o pedido várias vezes. Isto pode ser bastante perigoso visto que, em pedidos críticos, pode criar uma repetição de ações por parte do cliente. Dando o mesmo exemplo que foi dado nos ataques homem no meio, se o cliente criar uma encomenda na emissão de uma etapa, ao invés dos ataques homem no meio que podem alterar os dados para criar uma encomenda maior ou menor do que o suposto, os ataques de repetição podem criar várias encomendas iguais. Como podemos ver neste exemplo, o meio do ataque foi diferente mas o resultado foi bastante parecido.

Tal como nos ataques de falsificação de pedidos, existem várias soluções para este tipo de ataque. A primeira é a inclusão de uma marcação de tempo na assinatura do pedido, sendo que esse pedido fica inválido passado algum tempo depois de emitido. Esta solução não é perfeita, visto que um ataque de repetição pode repetir o mesmo pedido várias vezes por segundo. Isto faz com que a janela tenha de ser pequena, o que cria outros problemas com os pedidos legítimos. Temos como exemplo deste problema a existência de um atraso na receção de pedido (como por latência, atrasos no servidor de receção, entre outros). Neste caso, o pedido legítimo será considerado inválido. Outra solução, mais eficaz mas mais complexa, é a de idempotência. Idempotência é uma propriedade de algumas operações, em que caso estas sejam idempotentes, é possível serem aplicadas repetidamente sem que o resultado seja alterado [25]. Podemos introduzir impotência em operações que não as têm de uma forma simples, criando um identificador de chamada (também chamado de *nonce*). Isto resulta em que a operação, quando recebe dois pedidos com o mesmo *nonce*, pode ignorar o segundo e apenas repetir o resultado do primeiro ou simplesmente emitir um erro. Desta forma, quando existe um ataque de repetição, o recetor pode apenas receber o primeiro pedido e ignorar os restantes, o que resolve o problema.

Para além da segurança na comunicação, algo que também precisa de ser assegurado é a privacidade dos dados. Sendo a parte dos tenants gerido pela base de dados, não nos temos de preocupar com este ponto. No entanto, o backend contém uma cache "inteligente", isto é, uma cache que automaticamente atualiza quando deteta alguma alteração nos dados. Com isto, vão ser precisos vários testes e uma análise sobre o funcionamento desta cache para podermos garantir que os dados de um tenant não são enviados para o edocHook de outro tenant.

7.2 Páginas customizadas

Páginas customizadas são um conjunto de páginas, renderizadas dentro do eDoc e criadas pelo cliente. Desta forma, um cliente pode adicionar funcionalidades mais avançadas que requerem uma interface gráfica para utilização por utilizadores.

7.2.1 Arquitetura da solução

Para o frontend poder renderizar as páginas customizadas feitas pelo cliente, estas terão de ser hospedadas. A forma como são hospedadas e como serão obtidas serão vistas na secção 7.2.2. Depois do frontend obter os ficheiros necessários, irá atualizar o menu de modo a mostrar as entradas necessárias para o uso das páginas (que serão também configuráveis).

Para configurar o menu, será necessário que também esteja hospedado um ficheiro de configuração, onde se encontram as entradas e o *URL* respetivo.

Para este tipo de extensão vamos utilizar um conceito de modularidade de software chamado *ModuleFederation* [17], que é um empacotador de módulos. Por outras palavras, o *ModuleFederation* empacota (minifica e adiciona pacotes necessários à sua correta leitura) e permite a leitura de dado pacote por um projeto global. Temos como exemplo o próprio frontend do eDoc. Existe um projeto chamado de shell, onde ficam componentes gerais para todas as páginas. A shell também tem, na sua página, uma parte reservada aos micro frontends (utilizando um *router-outlet*). Quando um URL é aberto, a shell tenta encontrar o micro-frontend correspondente e renderiza-o no seu componente principal.

Estas páginas customizadas também poderão utilizar painéis laterais. Estes são usualmente utilizados para mostrar dados adicionais à página, edição das páginas ou subpáginas (como detalhes de um fluxo), porém podem ser utilizados quaisquer componentes. Temos como exemplos de uso de painéis laterais a edição da página inicial, vista de detalhes de um fluxo e detalhes de uma etapa. Isto cria uma necessidade de criar uma interface global simples de modo a que os clientes possam criar os seus próprios painéis laterais. Esta ferramenta será vista mais em detalhe no sub-módulo 7.2.3.

7.2.2 Hospedagem da solução da extensão

Uma das decisões mais importantes a escolher para a realização deste tipo de extensão é o local da hospedagem da solução cliente. Para este problema podemos optar por duas soluções, sendo estas a hospedagem por parte do cliente e a hospedagem por parte do eDoc. Cada uma destas soluções têm as suas vantagens e desvantagens, no entanto, ao contrário dos edocHooks, ficou decidido que apenas iríamos disponibilizar uma opção aos clientes. Assim sendo, é necessário escolher a melhor solução que facilite o uso desta ao cliente e, simultaneamente, não seja muito dispendioso para o eDoc.

7.2.2.1 Hospedagem da solução por parte do cliente

Nesta solução, supõe-se que seja o cliente que hospede a solução. Este tipo de hospedagem é mais simples na implementação e menos dispendiosa para a nossa equipa. No entanto, esta solução também cria uma nova camada de complexidade, segurança e custos para o cliente:

- **Complexidade** - Sendo o cliente a hospedar a solução, este terá de encontrar um serviço para a disponibilizar, configurar este serviço e configurar a integração deste com o serviço do eDoc. Caso o cliente não encontre nenhum serviço que disponha da funcionalidade de hospedagem de ficheiros, este terá de criar um de raiz. Isto pode ser feito a partir de um servidor privado (virtual ou não) ou de um serviço de *containers* em cloud. No entanto, o cliente também terá de saber como configurar este tipo de serviços de raiz. Esta complexidade pode ser um pouco reduzida com alguma documentação e apoios da parte da equipa do eDoc, no entanto continua sempre a ser um processo moroso e com alguma probabilidade para o cliente encontrar algum tipo de problema.
- **Segurança** - Existem duas falhas de segurança criadas por alguma configuração incorreta por parte do cliente. A primeira destas é a possibilidade de um atacante alterar os dados da solução hospedada, o que permite correr código potencialmente malicioso. Como a solução apenas fica hospedada no serviço e corrida no cliente, não existe possibilidade de correr código do lado do servidor. No entanto, a possibilidade de correr este código malicioso do lado do utilizador continua a ser uma possibilidade, sendo que o atacante pode intercetar dados fulcrais à segurança dos clientes. Estes dados podem passar desde dados presentes nas páginas, dados inseridos pelos utilizadores e até a interseção dos tokens de autenticação, o que permite ao atacante chamar pedidos da API representando o utilizador em questão.
  Outro problema grave com a segurança deste tipo de hospedagem é a leitura do código da solução sem permissões. Esta é uma falha grave, visto que o atacante pode obter alguns dados confidenciais necessários para o bom funcionamento da aplicação. Entre estes dados, encontram-se alguns tokens de autenticação, como por exemplo o token para uso de uma API externa ao eDoc, mas essencial para o funcionamento da aplicação. Estas APIs podem conter alguns dados vitais, que podem incluir, por exemplo, alguns dados fiscais da empresa cliente.
- **Custos** - Visto toda a gestão da hospedagem passar para o cliente, este terá de suportar os custos dos gestores destes serviços que, em caso de baixo uso, podem-se tornar algo com uma baixa relação custo/benefício. Este fator pode-se tornar algo pouco atraente para os clientes e que, em alguns casos, pode tornar esta ferramenta muito pouco importante para a escolha da plataforma gestora de documentos por parte do cliente.

Por outro lado, esta solução tem algumas vantagens que podem ser ponderadas antes de se tomar uma decisão. Entre estas, encontra-se um menor custo para o eDoc (o que se pode transmitir num menor preço da ferramenta), uma maior flexibilidade na escolha do provedor de serviços por parte do cliente e um possível aumento da velocidade da leitura de páginas customizadas.

- **Menor custo para o eDoc** - Sendo a hospedagem gerida pelo cliente, esse custo passa para o cliente que use este tipo de extensões. Assim, o custo que o eDoc teria para hospedar os ficheiros passa apenas para os clientes que necessitam, diminuindo

assim o custo de entrada para todos os clientes.

- **Maior flexibilidade na escolha do provedor de serviços** - Passando esta escolha para o cliente, este pode escolher o provedor do serviço que mais se adequa às suas necessidades, quer sejam estas desempenho, geográficas, ideológicas ou simplesmente monetárias. Outra vantagem com uma maior flexibilidade na escolha do provedor de serviços passa pela maior tranquilidade que alguns cliente podem ter com questões de segurança, em que podem ser estes que configuram toda a gestão de segurança à sua medida. Isto retira algum peso da equipa do eDoc, ao mesmo tempo que permite aos clientes gerirem as suas soluções como acharem melhor para o seu uso.
- **Possível aumento na velocidade de leitura de páginas customizadas** - Estando todos os clientes hospedados em serviços diferentes, a utilização dos serviços é espalhada entre estes, ao invés de estar focada apenas numa instância ou num provedor. Assim, a velocidade na leitura das páginas customizadas pode ser aumentada, dependendo também do provedor escolhido pelo cliente.

7.2.2.2 Hospedagem da solução por parte do eDoc

Nesta solução, a hospedagem e distribuição das soluções cliente ficam a cargo da equipa do eDoc. Este tipo de hospedagem é mais simples para os clientes, no entanto, é mais restrita e mais complexa para a equipa do eDoc. Esta complexidade dá-se tanto a nível de segurança como a nível de configuração.

- **Hospedagem mais restrita** - Ficando a hospedagem a cargo da equipa do eDoc, esta irá residir apenas num servidor/serviço, neste caso ficaria em cloud. Com isto, todos os clientes iriam, obrigatoriamente, utilizar o serviço escolhido pelo eDoc, o que pode não ser o ideal em alguns casos. Alguns clientes também podem ter alguns problemas com algum provedor em específico, como, por exemplo, uma alta latência devido a questões geográficas.
- **Complexidade a nível de segurança** - Estando todas as soluções de todos os clientes hospedados no mesmo serviço, qualquer falha de segurança implica que todas as soluções de clientes sejam consideradas comprometidas. Mesmo sendo possível uma maior segurança, visto que a integração com o sistema do eDoc pode ser mais direta e restrita, nenhum sistema é completamente seguro. Isto aumenta os problemas da equipa do eDoc caso se descubra alguma vulnerabilidade no sistema.
- **Complexidade a nível de configuração** - Sendo que todas as soluções ficariam no mesmo sistema, este teria de ser altamente otimizado e iria ter um alto nível de esforço para permitir uma utilização fluida por parte dos utilizadores. Com isto, qualquer falha na configuração teria impacto em todos os clientes, o que atrasaria qualquer alteração no serviço. Estando também todos os clientes no mesmo serviço, caso haja alguma falha, o tempo de baixa seria sentido por todos os clientes, podendo ser algo bastante prejudicial para a reputação do eDoc.

- **Aumento do preço do sistema para todos os clientes** - Sendo que, neste caso, os clientes não pagariam este serviço, todo o custo da hospedagem teria de ser dividido por todos os clientes e adicionado ao preço base do eDoc. Mesmo que o preço que cada cliente pagaria iria ser menor que se estes hospedassem as suas soluções, isto iria criar casos em que um cliente iria estar a pagar algo que não utilizaria.
  A importância deste ponto pode ser reduzida caso este valor fosse pago numa taxa, sendo que esta só existiria para clientes que utilizassem o serviço. Neste caso, esta taxa iria ser algo superior caso fosse dividido entre todos os clientes, e continuaria a ter o problema do uso que os clientes dariam a este tipo de extensão. Por exemplo, poderia existir um cliente que apenas necessitasse uma página customizada que pagaria o mesmo que um grande cliente que utilize centenas. Neste caso, o cliente com menos necessidade para o uso deste tipo de extensões poderia tentar arranjar outra solução, visto uma má relação preço-benefício.

No entanto, este tipo de hospedagem também tem alguns pontos positivos, como uma agilização no uso pelos clientes, um menor custo para alguns clientes e uma possibilidade de uma maior segurança, devido a uma melhor integração com o sistema do eDoc.

- **Diminuição de custos para alguns clientes** - Ao mesmo tempo que aumenta o preço para todos os clientes, os custos para os clientes que utilizam bastante esta extensão irão ser significativamente reduzidos. Neste caso, como o eDoc hospedaria todas as soluções de todos os clientes, o preço por solução iria ser bastante reduzido em comparação a hospedar soluções individualmente. Esta redução de custo iria passar diretamente para os clientes que utilizam bastante este tipo de extensão.
- **Complexidade** - Sendo o eDoc a hospedar todas as soluções, apenas será necessário configurar o serviço uma única vez. Isto reduz a complexidade para os clientes, sendo que estes apenas terão de nos enviar a extensão para a hospedarmos. Simultaneamente, esta alteração também terá uma diminuição na complexidade para o eDoc, visto que bastantes clientes iriam pedir para a nossa equipa os ajudar a configurar tudo para o bom funcionamento desta extensão.
- **Segurança** - Sendo o eDoc a configurar o sistema, a probabilidade de existirem falhas de segurança é bastante mais baixa. Sendo uma única entidade a hospedar todas as extensões, é possível investir mais na sua segurança do que em serviços individuais, o que reduz o risco para alguma falha.

7.2.2.3 Conclusão

No final, ficou decidido que iríamos passar a hospedagem para o cliente. Desta forma, permitimos que o cliente escolha a melhor solução para a hospedagem das suas soluções, ao mesmo tempo que aumentamos a segurança de todos os clientes caso um cliente necessite de utilizar mais recursos do que o sistema aguenta (vários pedidos simultâneos de uma solução complexa, isto é, um ficheiro com um tamanho bastante elevado).

7.2.3 Painel Lateral

No universo das aplicações web, as janelas modais têm um papel fundamental. Estas servem como dispositivos de apresentação de informações contextuais ou detalhadas que melhoram a compreensão do utilizador sobre um determinado elemento da página sem sobrecarregar a interface principal. Diferentemente da forma comum de se colocar estas informações na página, ficou decidido que no eDoc estas páginas teriam o formato de um painel lateral. Este painel lateral é um painel que aparece do lado direito da página principal. Estes painéis não têm navegação, sendo o seu uso mais próximo das tais janelas modais.

Outro ponto importante sobre os painéis laterais é a possibilidade de haverem vários abertos simultaneamente. Esta função já está implementada, no entanto a sua interface ainda está a ser pensada. Atualmente estão desenhadas duas soluções possíveis, sendo estas a apresentação múltipla e a apresentação única. A apresentação múltipla, tal como o nome sugere, apresenta vários painéis laterais lado a lado. Utilizando este tipo de apresentação, fica mais simples a leitura e escrita de dados em vários painéis, no entanto fica um pouco mais confuso e, em certos casos, pode ficar complicado de utilizar. No caso de haver bastantes painéis, pode ser necessário apresentar uma página de *scroll* horizontal, algo pouco utilizado em páginas *web*.

O outro modo de apresentação é a apresentação única. Neste modo, apenas um painel é visível em qualquer momento. No entanto, todos os painéis ficam guardados como uma "lista ligada", ou seja, é possível alterar o painel visível utilizando dois botões que avançam ou recuam na hierarquia. Isto reduz bastante a quantidade de informação a qualquer momento, o que é ao mesmo tempo positivo e negativo. Por um lado, uma menor quantidade de informação no ecrã aumenta o foco do utilizador e torna a aplicação menos complicada de se utilizar. Ao mesmo tempo, este modo complica a comparação de dados entre dois painéis e a cópia de dados entre painéis. Outra desvantagem deste modo é um aumento na complexidade de leitura de dados de dois painéis, algo que por vezes acontece.

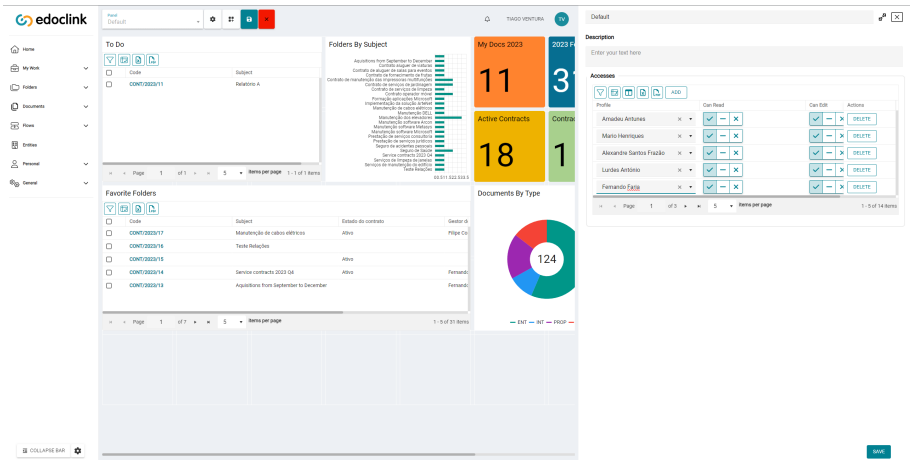

Figura 7.5: Exemplo de uma página com um painel lateral

Dada a importância central deste módulo no contexto mais amplo da aplicação, torna-se imprescindível oferecer aos clientes a capacidade de criar os seus próprios painéis laterais. Diante desta necessidade, a implementação de uma interface que permita a criação de tais painéis é fundamental.

No entanto, tal implementação tem de ser feita com dois objetivos em mente: simplicidade e funcionalidade. A simplicidade é necessária para garantir que os clientes possam, de maneira intuitiva e sem obstáculos técnicos significativos, utilizar tal interface para criar os seus painéis laterais. Ao mesmo tempo, esta interface deve ser poderosa o suficiente para permitir aos clientes aceder a todas as funcionalidades permitidas pelo eDoc. Este é um equilíbrio delicado a ser alcançado e a arquitetura da interface teve de ser pensada cuidadosamente para atingir este objetivo.

Com isto, ficou decidido criar uma classe abstrata com todos os métodos cuja implementação pode ser generalizada para a maioria dos casos de uso ficariam implementados de forma a facilitar o uso desta ferramenta por parte do cliente, ao mesmo tempo que os cliente poderiam alterá-los para criar funcionalidades que de outra forma seriam impossíveis.

Esta abordagem tem várias vantagens. Em primeiro lugar, ao fornecer métodos predefinidos que abrangem a maioria de casos de uso, a interface torna-se mais simples de se usar por parte dos clientes. Eles podem simplesmente implementar os métodos requeridos para realizar muitas das tarefas necessárias, reduzindo muito a complexidade do processo e a curva de aprendizagem associada.

Além disso, ao utilizar uma classe abstrata, ainda mantém-se a flexibilidade para os clientes personalizarem a funcionalidade de acordo com as suas necessidades específicas. Embora esta classe abstrata forneça uma base sólida sobre o qual os clientes podem construir, eles têm a possibilidade de alterar os métodos fornecidos para implementar funcionalidades que de outra forma seriam impossíveis de realizar.

Temos como exemplos de alguns métodos padrão desta classe os parâmetros de entrada e nível do painel. Para abrir um painel lateral, a solução encontrada foi a criação de uma estrutura no URL que define os painéis abertos. Estes têm o aspeto de https://URL/(nomePainel:arg1,arg2,...). Desta forma, não só é simples a abertura de qualquer painel a partir de qualquer página, também é possível os utilizadores partilharem páginas com todos os painéis abertos por padrão. Um exemplo pode ser um utilizador que quer partilhar uma etapa com outro utilizador, que poderá enviar https://URL/(etapa:CódigoEtapa), sendo que este URL automaticamente abrirá os detalhes da etapa.

7.2.4 Biblioteca de componentes e módulo "core"

Para utilizar a biblioteca de componentes ou o módulo "core"numa solução cliente, teremos de instalar estes a partir de um repositório NPM. Estas bibliotecas serão publicadas num repositório local, completamente controlado pela equipa do eDoc. Deste modo, os clientes

poderão utilizar as bibliotecas do eDoc sem existir a necessidade de instalar manualmente estes dois projetos.

Com isto, para utilizar ambas as bibliotecas é necessário correr os seguintes passos:

1. **Instalar as bibliotecas** - Para este passo, apenas é necessário correr os seguintes comandos (os URLs de registo são apenas ilustrativos e não estão ativos):

```
sysadmin@localhost:~$ npm install @edoclink-frontend/ui-components
↪ --registry=https://npm.edoclink.com/
sysadmin@localhost:~$ npm install @edoclink-frontend/core
↪ --registry=https://npm.edoclink.com/
```

Listagem 7.1: Comandos para importar módulos NPM

2. **Importar as bibliotecas dentro de um módulo** - Para utilizar as bibliotecas, é necessário importá-las num determinado módulo, sendo que apenas estarão disponíveis para o módulo em questão e módulos que o importem. O exemplo a seguir mostra como utilizar os componentes do eDoc:

```
...
import { UiComponentsModule } from '@edoclink-frontend/ui-components';

@NgModule({
  imports:      [ ..., UiComponentsModule ],
  providers:    [ ... ],
  declarations: [ ... ],
  exports:      [ ... ],
  bootstrap:    [ ... ]
})
export class SomeModule { }
```

Listagem 7.2: Código para a importação da biblioteca de componentes

Sendo a biblioteca de componentes e o módulo "core"criado utilizando a framework "angular", estes só funcionarão neste ambiente. No entanto, é possível criar páginas em qualquer framework suportada pelo *module federation*.

7.2.5 Segurança de páginas customizadas

Como dito anteriormente, a possibilidade de correr código no frontend tem bastantes riscos a nível de segurança. Porém, tomando as precauções necessárias, podemos diminuir bastante o risco ao ponto de ser considerado seguro.

A principal preocupação quando utilizamos esta tecnologia é a possibilidade de introduzir uma vulnerabilidade em que o atacante pode alterar o código da extensão, que, a partir deste ponto, irei de chamar de injeção de código fonte. Esta vulnerabilidade é bastante preocupante pois permite correr código malicioso diretamente no browser do utilizador. Com isto, um atacante consegue obter dados privados, como *tokens* e *cookies*,

tal como pode executar um ataque do tipo de homem no meio mesmo quando utilizado https. Por exemplo, um atacante pode obter o token de acesso ao eDoc quando um cliente entra na página infetada, o que permite ao atacante personificar o cliente sem obter os dados de autenticação.

Para prevenir este tipo de ataque, é necessário certificar que o ficheiro onde está compilada a solução do cliente não é alterado. Com esta finalidade, foram encontradas duas soluções:

- Assinatura do ficheiro com uma chave privada do eDoc, que ficará guardada com o ficheiro e certifica que a aplicação não foi alterada. Com esta assinatura, a qual vamos chamar de certificado de solução a partir deste ponto, podemos não só certificar que a solução não foi alterada, mas também ter algum tipo de certificação local como o código não tem nenhum tipo de vulnerabilidade que comprometa o eDoc. Podemos chegar a isso com a leitura do código por parte da equipa interna do eDoc, tal como algum tipo de validação com testes local. Desta forma, protegemos não só o nosso serviço, visto que pode haver algum tipo de falha na programação que faça vários pedidos em rápida sequência e sem parar ao nosso servidor, mas também o cliente, tendo a certeza que o código que ele escreveu será o enviado para os clientes e protegemos os seus dados.
- Validação do ficheiro através de uma assinatura utilizando um algoritmo *hash*. Esta assinatura, gerada por algoritmo *hash*, será enviada para o eDoc de alguma forma segura e negará a leitura das páginas customizadas caso a assinatura que temos não coincida com a assinatura da solução. Como utilizando um algoritmo *hash* temos a garantia que qualquer alteração ao ficheiro irá alterar a sua assinatura, podemos validar que se a duas assinaturas são iguais implica que o ficheiro não tenha sido alterado de nenhuma forma. Esta solução é mais simples que a assinatura com uma chave privada do eDoc, no entanto temos o problema de assegurar que a introdução da assinatura no sistema é fidedigna. Este problema pode ser minimizado se for alguém da equipa do eDoc a inserir a assinatura, mas continua a existir a possibilidade de haver alguma vulnerabilidade entre o cliente e a equipa do eDoc.

Outra preocupação é a possibilidade de leitura do código de um cliente por parte de outro tenant. Aqui entramos mais a fundo na proteção dos dados, mas também é uma vulnerabilidade importante de se considerar. Existindo esta vulnerabilidade, um tenant pode aceder ao código fonte de uma página de outro tenant, que pode conter alguns dados confidenciais como tokens de APIs externas e internas. Além disso, o código pode conter alguma vulnerabilidade escondida que pode ser mais facilmente detetada e explorada caso algum atacante tenha acesso ao código-fonte.

Para resolver este problema, a melhor solução será proteger o código por detrás de uma autenticação que contenha a informação com o tenant atual. Desta forma, um atacante apenas terá acesso ao código se já tiver acesso ao sistema, algo que torna este problema muito pouco valioso nos olhos de um atacante, visto que a principal razão para obter o código seria para obter vulnerabilidades que dariam acesso ao sistema.

Contudo, mesmo já tendo acesso ao sistema, um atacante pode tentar encontrar uma solução para escalar os privilégios. Face a este problema, o atacante terá duas soluções, sendo que apenas uma delas tem solução. A primeira é alterar o código para obter os tokens de todos os utilizadores, conseguindo assim credenciais com mais privilégios. Este problema já se encontra explicado e resolvido no ponto anterior. A outra solução passa pelo atacante tentar encontrar alguma vulnerabilidade na solução do cliente, algo que, dada a forma como este tipo de extensão funciona, não é possível resolver.

Por último, existe a preocupação de leitura de dados privados dentro do código da solução. Caso o atacante já esteja autenticado dentro do tenant correto, pouco se pode fazer. Dado o funcionamento desta extensão, caso um browser faça o pedido da solução para a disponibilizar ao cliente, esta tem de ser completamente disponibilizada para o utilizador. A única solução para este problema está numa comunicação aos clientes que este problema é algo que existe e que não pode ser resolvido, dando a solução aos clientes de estes protegerem estas chaves através de uma API interna intermédia. Esta API apenas faria a ligação do browser às APIs de destino, guardando as credenciais fora da solução dada aos utilizadores.

7.3 FormBuilder

Sendo o FormBuilder uma aplicação já existente e desenvolvida em Angular, ficou decidido que apenas será transferida para o novo eDoc. Com isto, será necessário fazer algumas alterações para funcionar como um micro-frontend dentro do eDoc. Entre estas, as mais importantes são a comunicação com o backend e a alteração do visual. Para a comunicação com o backend, será utilizado o módulo API já utilizado no resto da aplicação do eDoc. A interface de comunicação do eDoc terá, portanto, de ser alterada de modo a utilizar esse módulo.

A outra alteração será a alteração da interface gráfica. Para manter um estilo mais constante com o restante da aplicação, será alterada a biblioteca de componentes atualmente utilizada no FormBuilder pela biblioteca de componentes do eDoc. Isto faz com que haja um visual que não diste do restante da aplicação, ficando então uma ideia de uma aplicação global e concisa. Para além dos componentes, também será necessário o uso de painéis laterais em vez de janelas modais, tarefa que se pode mostrar bastante delicada.

7.4 Parâmetros Customizados

Devido à pequena complexidade desta tarefa, a estrutura desta não tem muito que justifique uma análise com algum detalhe. Mesmo assim, algo que teve de ser definido antes da sua implementação foi a divisão desta tarefa em várias sub-tarefas, de modo a apoiar tanto a validação dos resultados como a certificação que todos os passos são executados.

Com isto, a realização desta tarefa ficou dividida em 5 partes.

1. **Criar tabela na base de dados e procedimentos necessários** - Tal como o nome indica, será necessário criar a tabela dos parâmetros customizados na base de dados. Irá também ser necessário criar todos os procedimentos necessários para o uso desta extensão. Estes são criar, listar, ler, atualizar e remover.
2. **Criar métodos no SDK** - É necessário a criação dos métodos no SDK para a utilização desta extensão. Estes métodos são os mesmos que os procedimentos existentes na base de dados.
3. **Criação de uma interface REST** - Para a utilização desta extensão por parte do frontend, será necessário criar todos os métodos para a interface REST. Estes métodos também serão os mesmos que os procedimentos na base de dados.
4. **Criação das páginas e componentes no frontend** - Será necessário criar todas as páginas necessárias ao uso desta extensão por parte dos clientes. Com isto, será necessário criar duas páginas (sendo esta a listagem de parâmetros e a edição/leitura) e vários componentes, como por exemplo o componente para usar estes dentro das regras.
5. **Introdução dos parâmetros nas regras** - É necessário criar o mecanismo que permite às regras ler e escrever os dados dos parâmetros personalizados. Para isto, é necessário introduzir os componentes frontend nos locais de criação de regras e introduzir a leitura e atualização dos parâmetros na execução das regras no backend.

8

Implementação da solução

Neste capítulo irei apresentar todos os passos necessários para a realização da implementação da solução, ao mesmo tempo que apresentarei alguns problemas encontrados durante a implementação. No final, irei apresentar a forma como foram validados os resultados desta implementação.

Em relação ao desenvolvimento, é importante notar que dado o projeto ainda se encontrar numa fase muito inicial, não foi possível implementar alguns dos projetos. Nestes casos, uma de duas soluções foram utilizadas. A primeira destas foi a criação de um projeto externo com a funcionalidade pretendida implementada, no entanto sem nenhuma ligação com o sistema do eDoc. A outra solução foi a criação de um ramo na solução, onde foram implementados os requisitos necessários (mesmo que algumas partes seja algo hard-coded, sendo o importante que fosse algo funcional).

8.1 edocHooks

A implementação dos edocHooks foi faseada, alterando uma solução de cada vez. Este método permitiu testar bastante antes de avançar para outra tarefa. No entanto, utilizando este método implicou a criação de alguns métodos estáticos que simulam a utilização de todos os métodos necessários.

A solução encontrada para a implementação dos diferentes modos de execução foi a criação de uma função assíncrona no backend que faz o pedido para o servidor cliente. Assim, caso o edocHook seja síncrono, esperamos pela execução desta função, e caso não seja corremos em paralelo. Esta foi uma solução bastante simples, no entanto é o suficiente para conseguir obter todas as funcionalidades que necessitamos para esta extensão.

Durante a fase anterior ficou decidido que, por razões de segurança, o uso de HTTPS seria obrigatório. No entanto, a criação e manutenção de um certificado reconhecido por uma autoridade de certificados pode-se tornar uma tarefa complexa para quem a está a realizar pela primeira vez. Por esta razão, também ficou decidido que o eDoc também iria aceitar servidores com certificados auto-assinados. No entanto, caso não tivéssemos nenhum tipo de validação, qualquer atacante poderia criar um certificado

auto-assinado para um ataque homem no meio. Para resolver este problema, criei uma forma de autenticação de certificados. Caso o cliente queira utilizar um destes certificados, terá de comunicar a assinatura deste à equipa do eDoc, ficando então o certificado ligado ao servidor cliente. Esta assinatura fica guardada na base de dados e o sistema apenas aceitará pedidos quando a assinatura do certificado seja igual à enviada pelo servidor do cliente.

No final, a implementação desta tarefa ficou algo simples, visto que um WebHook consiste num simples pedido HTTP. Os únicos pontos que necessitaram de mais pesquisa e análise foram os pontos vistos em cima, os certificados auto-assinados e os modos de execução.

8.2 Páginas Customizadas

No início desta extensão, comecei por fazer um teste sem backend. Nesta versão, a lista de páginas customizadas estava no frontend como uma variável constante. Desta forma foi possível mostrar uma versão básica, mas com a ideia principal a funcionar para demonstrar como funcionará a aplicação. Após aprovada, esta versão foi apresentada à equipa de suporte para se ambientarem com as ferramentas de modo a conseguir ajudar o cliente quando este precisar de ajuda.

De seguida, o próximo passo foi ligar esta ferramenta ao backend. Durante esta fase ainda não estava pensado utilizar a base de dados, sendo apenas necessário passar tal constante para o backend. Com isto, conseguimos acabar a parte da leitura de páginas customizadas por parte do frontend, ficando apenas o backend e a edição restantes. Esta fase teve bastantes problemas, algo que será visto com mais detalhes na subsecção 8.2.1.

Visto o eDoc ainda não ter nenhum local para hospedar as soluções dos clientes, os ficheiros contendo as páginas customizadas foram hospedados localmente. O URL destes teve, portanto, de ser hard-coded.

8.2.1 Problemas encontrados e soluções implementadas

Na primeira versão desta extensão, a lista de páginas customizadas estavam escritas no código do frontend para não ser necessário realizar pedidos ao backend. Isto foi bastante útil na fase inicial, visto que facilitou o teste de funcionalidade desta implementação sem ser necessária a preocupação com problemas com as ligações.

Mesmo sendo esta fase muito simples em questão de complexidade, os problemas começaram logo no início. O principal problema foi a funcionalidade de leitura de páginas customizadas a partir de uma lista de URLs que não seria compilada pelo gestor de micro-frontends da que era utilizada. Sendo que todo o projeto estava pensado e implementado com esta framework em mente, alterá-la seria algo demasiado complexo e que rapidamente foi descartado. Depois de alguma pesquisa, foi encontrada uma extensão que permitia a

leitura de micro-frontends a partir de um URL cuja declaração seria feita em tempo de execução. Esta extensão foi, portanto, essencial à execução deste módulo.

Com este problema resolvido e com a aplicação a abrir páginas customizadas a partir de uma lista, o passo seguinte seria a leitura desta lista a partir de um endpoint vindo da API no backend. Esta fase foi mais complicada do que inicialmente pensado, apresentando vários problemas distintos. O primeiro destes foi a questão da autenticação.

Para a autenticação do eDoc é utilizado um serviço chamado Keycloak [16]. Este permite adicionar autenticação a aplicações de uma forma simples, visto que este é um serviço de *SSO* (serviço que permite *logon* único) e toda a lógica de autenticação fica do lado do Keycloak. Assim, a sua utilização fica bastante simples em qualquer aplicação, podendo também os utilizadores e tokens partilhados por múltiplas aplicações. Esta é uma funcionalidade importante, visto que está previsto que sejam desenvolvidas algumas aplicações separadas do eDoc, mas que partilham a mesma base de clientes. Desta forma, os utilizadores não terão de voltar a iniciar sessão quando alterarem de aplicação. Como podemos constatar na imagem 8.1, o login da aplicação é completamente gerido pelo Keycloak.

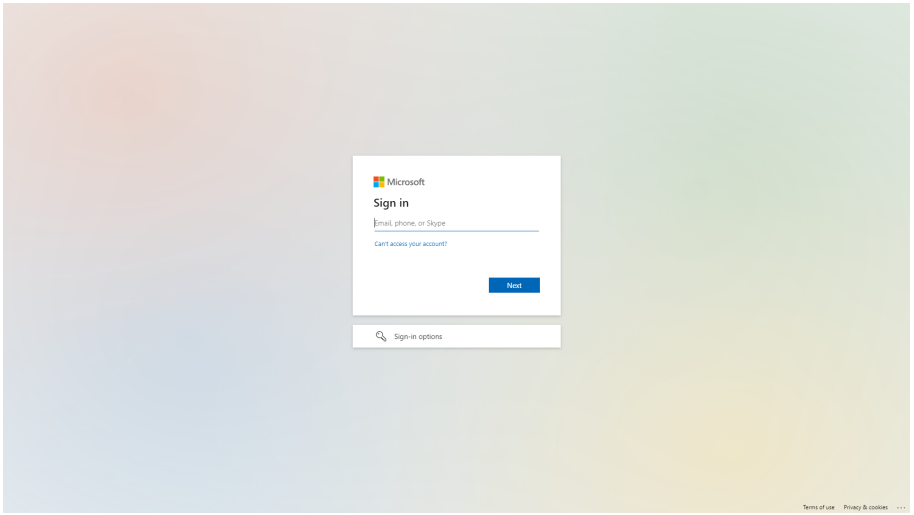

Figura 8.1: Exemplo de login do Keycloak utilizando SSO da Microsoft

Também podemos observar na figura 8.1, que apresenta a página de login para o ambiente de desenvolvimento do eDoc, que o Keycloak permite uma autenticação utilizando um serviço *SSO* (*Single Sign-On*). Estes são serviços de autenticação que permite a autenticação de múltiplos serviços utilizando apenas um conjunto de credenciais. Tal como apresentado na imagem, o serviço *SSO* utilizado pela equipa do eDoc para o ambiente de desenvolvimento é o da Microsoft.

A autenticação feita dentro do eDoc utiliza um *provider* dentro do projeto angular que é executado quando a aplicação é inicializada. Neste método, o serviço de Keycloak é inicializado e a aplicação obtém o seu token ao serviço. A outra parte deste problema é como se pode atualizar o *router* (gestor dados onde os URLs são mapeados às páginas, ou

rotas) para manter o caminho quando abrimos diretamente a partir do URL. A solução encontrada foi atualizar as rotas na inicialização da aplicação. Com isto, foi rapidamente decidido que teríamos de ler a lista de páginas da API na logo a seguir a concluir a autenticação. Aqui apareceu o primeiro problema, visto que o angular não tem nenhuma lógica para executar dois métodos de inicialização sequencialmente. Após analisadas todas as opções possíveis, ficou decidido que a solução mais simples seria ler a lista de páginas customizadas no mesmo método que se fazia a inicialização do Keycloak.

O eDoc é criado a partir de um conjunto de módulos, cada um com as suas funções. Para esta extensão, apenas foram alterados 2 módulos, o módulo "core"e o módulo "AppRouter". O primeiro tem toda a lógica a ser utilizada por toda a aplicação, tal como toda a lógica de autenticação (desde a ligação com o Keycloak com a adição dos *headers* necessários para a comunicação com o backend), *cache* e recursos de tradução para vários idiomas. O módulo "AppRouter"tem como única função a gestão das rotas para todos os micro-frontends da aplicação. O módulo "core"irá ser partilhado com os clientes, visto que este é necessário para a criação de páginas customizadas.

Esta solução tem os seus problemas, visto que toda a lógica da autenticação está criada no módulo *core*, e a parte das rotas no módulo "AppRouter". Uma solução criada de modo a eliminar os erros de lógica quando executado o módulo "core"sem ser necessário utilizar rotas, como quando executado pelo cliente para criar páginas customizadas, foi criar uma variável partilhada no módulo das rotas e apenas inicializar as páginas customizadas se essa variável existir.

O problema final encontrado na implementação desta extensão no frontend foi que o *provider* de inicialização da aplicação corria antes de se ler as configurações para comunicação com o backend. Com isto, quando se lia os dados do backend, o caminho estava incorreto e não conseguia chegar ao backend. A solução para este problema foi bastante simples, sendo apenas necessário chamar o método de leitura de configuração antes de se chamar o backend. O problema com esta solução foi que, quando se lia as configurações mais tarde, este módulo dava erro, pois tentava alterar dados que estavam a ser utilizados por outros métodos. A solução foi a validação que os dados necessitam de atualização antes de atualizar os dados. Assim, caso a configuração seja igual, nada será alterado e não levantará nenhuma exceção.

8.2.1.1 Problemas encontrados na implementação do backend

Ao contrário do que acontece no frontend, parte do backend desta extensão é uma simples ligação com a base de dados e o suporte aos diferentes métodos para a gestão desses dados. Esta tarefa é algo bastante comum, pelo que o sistema já está completamente pronto a aceitar este tipo de funcionalidade. Por este motivo, esta tarefa foi relativamente simples e não foram encontrados nenhuns problemas.

À semelhança do que aconteceu com o restante desta tarefa no backend, a alteração do menu para aceitar dados dinâmicos também foi relativamente simples. Neste caso,

a alteração efetuada para resolver este problema foi a leitura de duas tabelas quando o pedido dos dados do menu por parte do frontend. Estas duas leituras são a leitura dos dados estáticos (estes encontram-se na base de dados devido ao elevado número de alterações que a equipa do eDoc faz nestes) e a leitura dos dados dinâmicos. Devido ao facto de os dados do menu já estarem relativamente dinâmicos, apenas foi necessário copiar o código do que já estava implementado com algumas alterações.

8.2.1.2 Problemas com a introdução dos módulos partilhados no NPM

Para ser possível proceder à avaliação deste módulo, foi necessário hospedar os dois módulos partilhados (módulo core e biblioteca de componentes) num repositório NPM. Esta fase trouxe alguns problemas, visto ser a primeira vez que esta tarefa foi executada pela equipa do eDoc.

Comecei por instalar um gestor de pacotes, chamado *Verdaccio* [29]. Este é um registro local gratuito com todas as funcionalidades que necessitamos para esta fase do projeto. A sua instalação foi bastante simples, apenas sendo necessário instalar o projeto com o uso do NPM e correr dois comandos na consola para se criar um utilizador.

Após instalado e configurado o *Verdaccio*, foram criados os dois registos dos módulos partilhados. Esta fase também foi simples, apenas diferindo da forma usual de criar dois registos no NPM pela adição de um argumento nos comandos a referir o registo local.

A última fase foi a de instalação e utilização dos módulos dentro de um projeto externo. Foi neste ponto que os maiores problemas apareceram. Inicialmente, não era possível importar o projeto dentro do módulo principal do projeto externo. Ao tentar perceber o que estava a acontecer, reparei que o IntelliSense do VSCode encontrava os módulos quando se estava a tentar importar, no entanto o angular dizia que estes não existiam. Quando visto mais a fundo, foi constatado que o problema era na definição dos módulos quando compilados. A solução foi simples, no entanto complexa de se encontrar. No final, foi necessário adicionar uns tokens na definição do módulo de modo a permitir uma compilação correta. Depois de resolvido este pequeno problema, o angular não reportou mais nenhum problema.

Quando executada a aplicação localmente, não apresentou nenhum problema visual nem funcional, passando então para o teste da integração deste projeto para o eDoc. Quando compilado e lido pelo eDoc, esta integração funcionou na íntegra, não havendo mais nenhuma complicação.

8.2.2 Resultados

Visualmente esta extensão é bastante simples, visto que esta é uma extensão em que o cliente criará a sua própria *interface*.

No final, foi criada 1 página para a configuração das páginas. Tal como podemos observar pela figura 8.2, esta página é bastante simples, contendo apenas uma lista com todas as páginas customizadas do cliente. Nesta lista é possível editar, clicando no botão

do lado direito da lista. Isto abrirá um painel lateral com os dados da entrada em questão, sendo possível ler e/ou editar os dados.

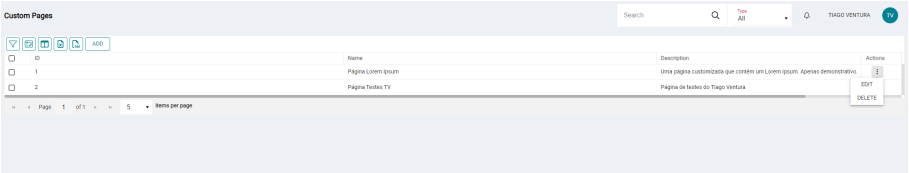

Figura 8.2: Página de listagem de páginas customizadas

Como exemplo de página customizada podemos observar na figura 8.3 em que se encontra uma página criada, compilada e hospedada de uma forma completamente isolada do ambiente do eDoc. Mesmo que este seja um exemplo simples, mostra a principal funcionalidade desta extensão. Na imagem também podemos observar o menu com as entradas necessárias para a utilização desta página.

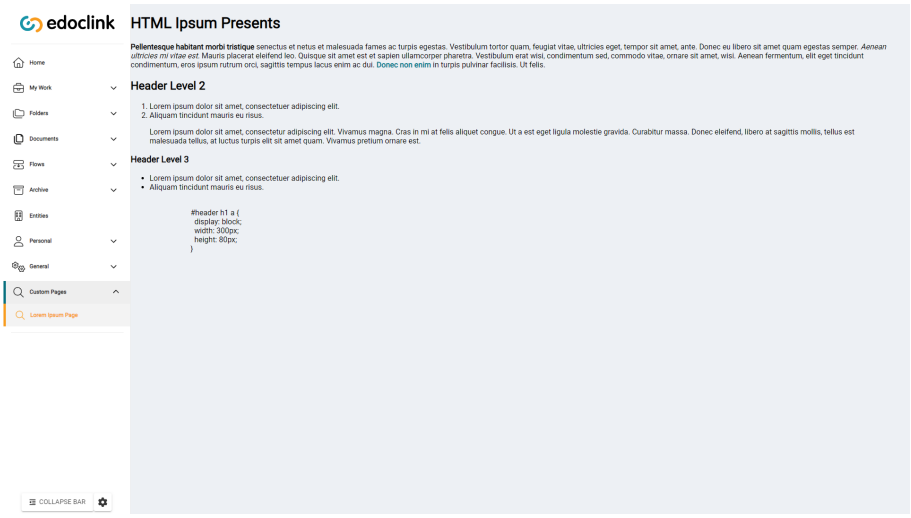

Figura 8.3: Exemplo de uma página customizada com entrada no menu

8.3 FormBuilder

Durante a implementação das páginas customizadas, ficou decidido internamente que o FormBuilder seria implementado externamente, de modo a poder ser utilizado por todas as equipas da LinkConsulting. Com isto, também será possível vender o produto de modo a ser utilizado em aplicações de clientes finais, no entanto, esta funcionalidade foi apenas mencionada e não está pensado seguir este caminho.

No entanto, a criação de uma nova versão do FormBuilder é um projeto de dissertação de outro colega da FCT, sendo que a sua conclusão está prevista para meados do ano de 2024. No entanto, como está previsto a primeira versão do eDoc ser disponibilizada para clientes no final do ano de 2023, ainda que numa fase básica, ficou decidido transferir temporariamente o FormBuilder da versão anterior do eDoc para a versão que está a ser

desenvolvida. Esta tarefa trouxe algumas dificuldades, mas o resultado ficou bastante coerente com o resto da aplicação.

Para além de ser necessário atualizar a comunicação do FormBuilder com o backend, foi também necessário atualizar o visual da aplicação. Para ficar o mais consistente com o aspeto do eDoc, foram atualizados todos os campos para utilizar a nova biblioteca de componentes. Isto trouxe algumas dificuldades visto que alguns componentes trabalham diferentemente dos que estavam anteriormente. Com isto, alguns campos tiveram de ser totalmente reimplementados, enquanto outros apenas foram necessárias algumas alterações. Isto é algo que será visto com mais detalhes na sub-secção 8.3.2.

8.3.1 Problemas encontrados e soluções implementadas

O primeiro problema encontrado com a transferência desta ferramenta foi a de passagem da comunicação com o backend do FormBuilder para um módulo partilhado do eDoc. Na sua versão anterior, o FormBuilder geria toda a comunicação com o backend. Alguns métodos tiveram de ser removidos (como a inserção dos tokens de autenticação nos pedidos ao backend), enquanto outros tiveram de ser significativamente alterados. Sendo os métodos dos tokens um simples *HttpInterceptor*, este foi facilmente removido. No entanto, este *provider* tinha bastante métodos adicionais que permitiam o bom funcionamento deste. Foi na remoção destes métodos adicionais que se mostrou um problema maior, visto que alguns alteravam a forma como o FormBuilder era inicializado e estavam bastante enraizados na aplicação.

Um destes métodos foi a leitura dos tokens do utilizador ao eDoc, utilizando uma "comunicação"entre os dois frontends a partir de alguns métodos na consola. Visto o FormBuilder e o eDoc estarem escritos em diferentes frameworks, a comunicação direta entre o frontend dos dois era impossível. Com isto, a solução encontrada para uma comunicação simples de alguns dados vitais ao bom funcionamento do FormBuilder foi criar alguns métodos globais no eDoc, sendo que o FormBuilder os corria para obter os dados que necessitava. Temos como exemplo a necessidade de o FormBuilder obter o token de autenticação. Com isto, o eDoc tem um método chamado "getEdocToken()"que quando executado retorna o token de acesso à API do eDoc. Para o FormBuilder obter o token era, portanto, apenas necessário fazer a chamada desse método.

É nestes métodos que está o maior problema, visto que algumas funções do FormBuilder não são inicializadas sem os dados destes. Com isto, foi preciso encontrar todos os locais onde esta funcionalidade era utilizada e criar uma solução para eliminar estas chamadas problemáticas.

Outra fonte de problemas foram as próprias chamadas à API, sendo que anteriormente esta lógica era gerida pelo FormBuilder. A alteração foi algo simples, visto que a aplicação já tinha um serviço que fazia a *interface* entre a aplicação e o backend. Assim sendo, apenas foi necessário alterar este serviço para utilizar a nova biblioteca. No entanto, alguns tipos de dados foram alterados entre as duas versões, logo foram precisos encontrar essas

diferenças e alterar no projeto onde fosse necessário. Isto tornou-se uma tarefa morosa e bastante complexa de se testar, visto que qualquer ponto falhado pode arranjar problemas nalguma funcionalidade escondida da aplicação.

Um local bastante problemático para o FormBuilder era a lógica de apresentação de campos no modo builder. Para melhorar o desempenho da aplicação e reduzir o número de problemas no futuro, toda esta lógica foi alterada. Visto que toda a lógica já estava separada da aplicação e esta era apenas chamada num ponto, foi bastante simples concluir esta tarefa. No final, toda a lógica de renderização dos campos foi transferida dos ficheiros *JavaScript* para o renderizador do angular, como já estava feito no modo render.

8.3.2 Alterações a campos do FormBuilder

Durante esta tarefa, todos os campos foram alterados. Antigamente era utilizado a biblioteca de componentes do *Bootstrap* [4], enquanto agora será utilizada a biblioteca de componentes do eDoc, sendo esta baseada em *KendoUI* [15]. Sendo a base das duas bibliotecas bastante diferentes, para além de alterar o código HTML dos campos, também foi necessário alterar a lógica de vários destes campos.

Temos como exemplos alguns campos em que foi necessário alterar bastante lógica:

- Na versão antiga, o campo booleano foi um campo criado de origem. Este foi desenvolvido utilizando três objetos <*div*> estilizados de modo a parecerem com 3 botões lado a lado. Para guardar e apresentar o valor escolhido, o campo tem uma variável onde o valor se encontra guardado e ao clicar num dos <*div*>, o conteúdo desta variável altera para corresponder com o valor introduzido. Para apresentar o valor selecionado ao utilizador, ao <*div*> cujo valor atribuído corresponde com o valor do campo é adicionado uma classe CSS que altera a cor com que este é renderizado.
- O campo "tabela"era apresentado ao utilizador utilizando uma biblioteca chamada *tabulator* [26]. O funcionamento desta foi bastante alterado de modo a apresentar subtabelas recursivamente. Com a biblioteca de componentes atual, este funcionamento já está presente, visto que este campo também teve de ser bastante alterado.
  Como o eDoc também utilizava este campo, toda a lógica de renderização e edição da tabela foi criada num ficheiro externo. Desta forma, a passagem do campo para o eDoc foi bastante simples e qualquer alteração no funcionamento da tabela era facilmente replicado pelos dois sistemas. Com isto, a alteração do campo tabela foi um pouco mais simples que no caso do campo booleano, no entanto, alguma lógica na ligação dos dados com o sistema do FormBuilder teve de ser refeita.

8.3.3 Resultado

Tal como dito anteriormente, o visual deste módulo foi alterado. Nas imagens seguintes conseguimos comparar o estilo anterior (imagem 8.5) e o atual (imagem 8.4).

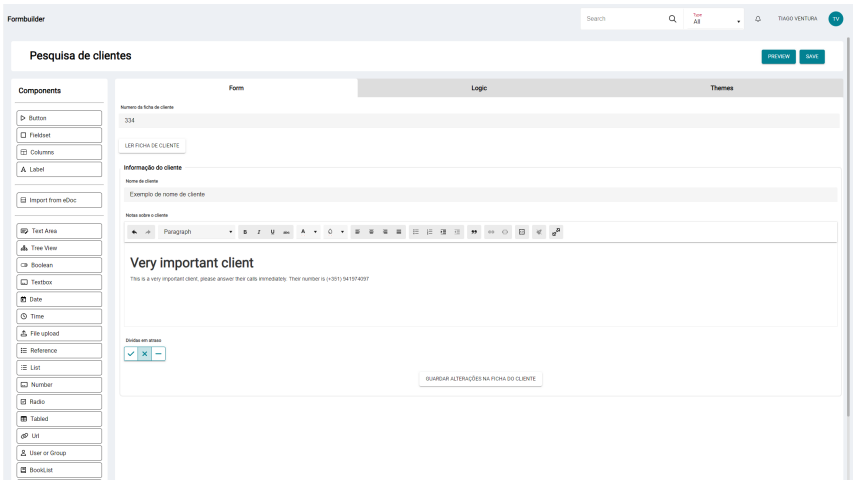

Figura 8.4: Estilo do novo FormBuilder

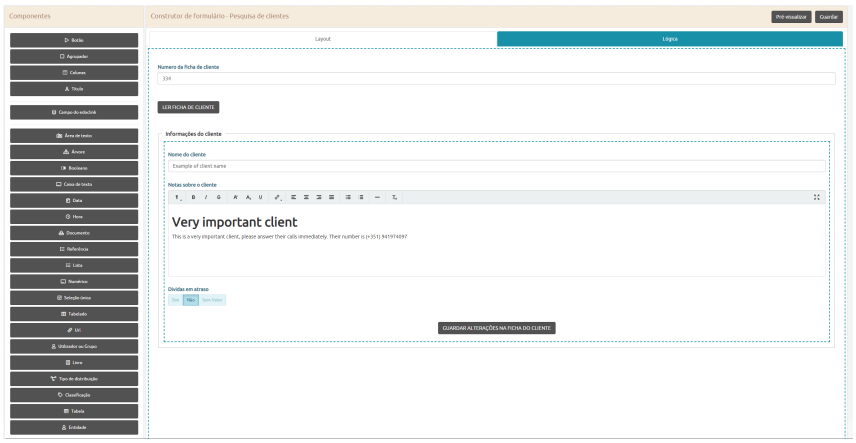

Figura 8.5: Estilo do FormBuilder antigo

A alteração visual mais percetível é a alteração do padrão de cores utilizado em toda a aplicação. Como podemos observar, o tema principal azul/bege passou para um tema bastante mais claro, com uma maior dominância da cor branca. Outra alteração visual bastante percetível é a alteração do estilo dos cabeçalhos. Como podemos observar, estes deixaram de estender toda a largura da aplicação, ficando portanto um pouco mais moderno.

Para além destas alterações, os campos foram algo onde o estilo também foi bastante alterado. No entanto, esta alteração deve-se à mudança de biblioteca, não estando relacionado com esta tarefa.

Como podemos constatar pelas imagens seguintes, as janelas modais foram substituídas por painéis laterais. Isto aumenta a coesão entre páginas no eDoc enquanto utiliza os componentes do eDoc, o que reduz a quantidade de erros que podem existir. Utilizando os componentes do eDoc também significa que qualquer problema que exista irá ser rapidamente detetado e, quando reparado, ficará funcional de uma forma geral.

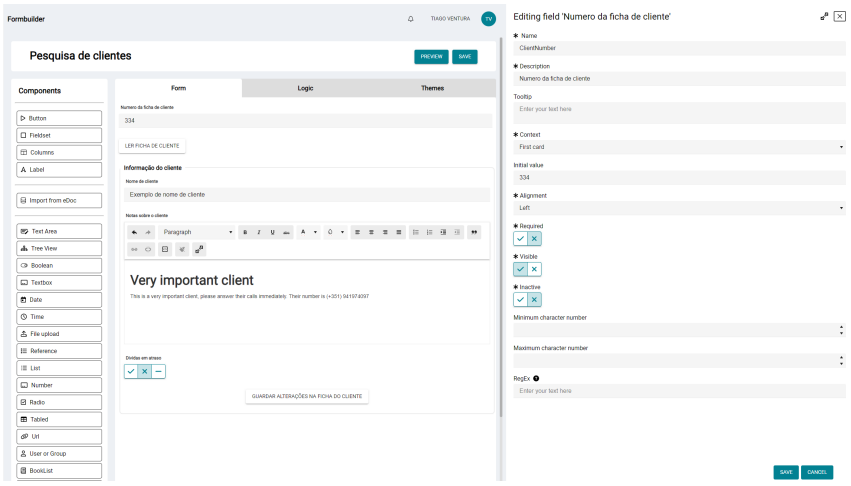

Figura 8.6: Novo painel lateral no FormBuilder

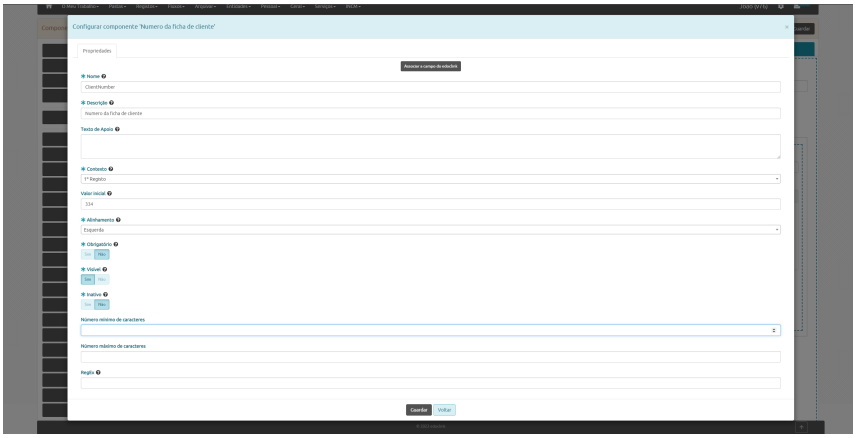

Figura 8.7: Modal antigo no FormBuilder

8.4 Parâmetros Customizados

A implementação desta extensão foi relativamente simples, no entanto, foram necessários vários passos para a sua implementação completa.

Como dito anteriormente, foi necessário a implementação de uma *interface* no backend que faz a gestão destes parâmetros na base de dados. Esta *interface* ficou no projeto eDocSDK. Após completo, foi criada uma *interface* REST para permitir a execução de todas as funções por parte do backend ou outros projetos que o cliente possa necessitar. Estas funções são: listar, ler, editar, criar e remover.

Após completar estas alterações, passei para o backend. Inicialmente foi criada uma página para a listagem de parâmetros, onde têm botões para editar, ler, criar e remover estes campos. Também foi criada uma página com os detalhes dos parâmetros para se poder ler, criar e editar. Após completa a fase da gestão de parâmetros, foram criados alguns campos para permitir o uso destes parâmetros nas regras. Isto permite que o utilizador possa utilizar estes campos tanto em condições como em valores de regras. Esta fase foi um pouco mais complexa, pois foi necessário encontrar todos os locais onde as

regras os liam e utilizavam os campos.

8.5 Testes unitários

Após completar estes passos, foi necessário implementar todos os testes unitários para este módulo. Tal como a implementação desta solução, a implementação destes testes também foi relativamente simples, visto que não foi necessário a implementação de nenhum teste visual. No entanto, foi preciso alterar alguns testes que já tinham sido elaborados, como, por exemplo, o das regras. Com isto, foram criados métodos para todas as funções dos parâmetros encontrados em cima.

De modo que a base de dados acabe com os mesmos dados com que começou, os testes começam por encontrar alguns códigos para os parâmetros que não existem. Após encontrados, testa a adição, listagem, leitura, edição e remoção dos componentes por esta ordem. Quando acabado de testar os métodos dos parâmetros, guarda o nome dos campos utilizados para poderem ser criados na fase das regras.

Durante o teste das regras, são criados os parâmetros no início dos testes e removidos no final. Desta forma, estes podem ser utilizados nas regras e, no final, a base de dados acaba com os mesmos dados com que começou.

9

Validação de resultados

Para a validação dos resultados, foram utilizados três métodos: testes unitários, testes visuais e testes manuais.

- **Teste unitários** - Como dito anteriormente, foram criados testes unitários para todos projetos realizados durante a realização desta dissertação. Estes testes apoiaram bastante no momento do desenvolvimento, visto que, quando foram necessárias alterações, foi facilmente validado se o resultado pretendido anteriormente continuava a funcionar. Simultaneamente, também era possível validar que todo o código desenvolvido funciona para todas as versões encontradas em clientes, sendo que os testes unitários apenas são alterados quando existe alguma funcionalidade que deixa de ser suportada.
- **Teste visuais** - À semelhança dos testes unitários, estes também foram criados para todos os projetos implementados. Neste caso, para além de apoiar na validação aquando da alteração de código, também apoia a validação que toda a aplicação está concisa. Dado que os testes visuais guardam todas as páginas numa pasta partilhada, é possível todos os elementos da equipa conseguirem validar que todas as páginas têm um estilo conciso. Outra vantagem dos testes unitários é o apoio que dá à equipa de apoio ao cliente, visto que estas imagens podem ser utilizadas para a criação de alguns fluxos de trabalho para oferecer aos clientes.
- **Teste manuais** - Estes testes são os mais complexos de se gerir, no entanto são os testes onde encontramos o maior número de problemas e pontos não otimizados da implementação. Destes, temos como exemplo o tempo de timeout, que estava demasiado curto para quando estávamos a trabalhar com latências normais fora da rede local. Para tentar englobar todos os tipos de utilizadores nestes testes, foram criados três grupos com diferentes experiências com eDoc. O primeiro grupo foi com uma equipa interna no eDoc, contendo alguns elementos da equipa de desenvolvimento, outros da equipa de testes e ainda alguns da equipa de apoio ao cliente. Este grupo permitiu encontrar algumas funcionalidades que não estavam a ser abrangidas pelos projetos desenvolvidos.

  O segundo grupo foi constituído por clientes já existentes do eDoc. Este grupo

permite ter uma ideia de como os utilizadores de antigas versões do eDoc utilizam a aplicação, principalmente onde se encontram os pontos onde existe uma maior dificuldade no uso da aplicação. Com este grupo, também é possível encontrar alguns pontos que estão mais confusos. Isto é algo difícil de se testar com a equipa interna, visto que estes estão mais acostumados com toda aplicação, algo que não permite encontrar algumas falhas.

O terceiro e último grupo consiste de pessoas que nunca utilizaram o eDoc ou que têm muito pouca experiência com o sistema. Este grupo permite encontrar algumas partes do programa que podem estar pouco concisas ou que poderiam ser arranjadas de outro modo. Não tendo nenhuma experiência com o sistema, este grupo permite que encontremos os pontos que mais aumentem a resistência de utilização do sistema por clientes novos. Um bom exemplo para este facto são as páginas de edição dos edocHooks. Alguns clientes que nunca tinham utilizado o eDoc estavam com dificuldades a encontrar os locais onde está a configuração dos eventos subscritos, algo que conseguimos resolver criando um botão que abre diretamente a página onde se pode configurar estes eventos.

Outra forma de validar os resultados foi utilizando um questionário após os testes manuais. Este formulário continha 7 perguntas e as respostas eram um valor numérico que começava com 1 (não/não concordo) e acabava com 10 (sim/concordo). As perguntas disponíveis e os os seus resultado médios (arredondados às unidades) foram os seguintes:

- **1 - O sistema dos edocHooks consegue substituir o uso atual dos triggers?** - Resultado médio: 5
- **2 - Os edocHooks têm vantagens sobre os triggers?** - Resultado médio: 7
- **3 - O sistema novos das páginas customizadas consegue substituir o uso antigo?** - Resultado médio: 9
- **4 - As novas páginas têm vantagens sobre o uso antigo?** - Resultado médio: 9
- **5 - A interface do FormBuilder está concisa?** - Resultado médio: 10
- **6 - O FormBuilder está mais fácil de utilizar que o mesmo do eDocv7** - Resultado médio: 6
- **7 - Os parâmetros customizados contêm todas as funções que existiam no eDocv7** - Resultado médio: 10

Como podemos observar, os pontos 1, 2 e 6 tiveram resultados baixos. Entre estes, o resultado que mais me espantou quando estudei os resultados foi o ponto 6. Para tentar compreender o baixo valor desta questão, tentei obter mais informações junto das pessoas que a responderam. A justificação dada é bastante lógica, sendo que o problema está na pergunta. Como o novo FormBuilder foi uma cópia do antigo (apenas alterando muito pouco a lógica dos campos e a interface), a facilidade de uso continuou igual. Após esta justificação, tentei perceber um pouco melhor o porquê de, neste caso, a média ser diferente de 5, sendo que o software é igual. No final, existiam alguns problemas com o campo booleano que foi resolvido nesta versão, o que facilita o seu uso.

Os outros dois resultados mais negativos foram os pontos 1 e 2. Os problemas presentes

nestes pontos já estão relacionados com a própria tecnologia, no entanto um feedback que recebi de praticamente todos os responderam negativamente a estas perguntas foi que existem algumas falhas na documentação deste módulo. Sendo este tipo de extensões em si bastante extensível, uma documentação completa é bastante complicada, no entanto faltam alguns pontos importantes para reduzir um pouco a perceção de falta de funcionalidades.

10

Conclusão

Como foi possível observar nesta dissertação, foi encontrada uma solução para os problemas definidos na motivação. Sendo um sistema SaaS multi-tenant, foi dada uma grande importância à segurança deste serviço. Este ponto foi bastante analisado durante esta dissertação e todas as falhas encontradas foram devidamente resolvidas.

Algo de notar é que, mesmo que a maioria das extensões tenham sido melhoradas de algum modo, a ideia geral foi que os triggers eram mais simples de se utilizar que os edocHooks. Para esta discrepância tenho duas teorias, sendo a primeira que apenas é um sistema diferente que demora algum tempo até se ficar confortável. A outra teoria será que, sendo este um sistema em cloud, algumas funções podem estar com menos funcionalidades e pode ser um pouco mais lento a executar todas as tarefas. Caso esta última se venha a mostrar verdadeira, não consegui, até agora, encontrar nenhuma solução simples e segura para a resolver.

10.1 Trabalho futuro

Penso que esta dissertação ainda tem alguns pontos em que pode ser melhorada. Dentro deste pontos encontra-se um FormBuilder extensível com uso de *plugins*. Este novo FormBuilder poderia não só ser utilizado por qualquer equipa da Link Consulting, mas também poderia ser vendido para o uso de clientes da Link. Estes *plugins* adicionariam campos, ações, lógica, integração com os sistemas clientes e outras funcionalidades que os clientes necessitem sem ser preciso alterar o código fonte da aplicação principal. Este projeto poderia ser realizado utilizando a mesma tecnologia que as páginas customizadas, mas o *Module Federation* iria ler módulos com componentes e serviços em vez de ler e renderizar páginas.

Outro ponto onde este projeto poderia ser estendido poderia ser com uma loja de *plugins*, ou um portal *self-service* de integrações. Este módulo teria a mesma forma e finalidade da loja de extensões do *Visual Studio Code*, por exemplo. Com isto, os clientes poderiam utilizar integrações com sistemas conhecidos sem ser necessário criar a extensão de raiz. Estes *plugins* poderiam ser vendidos, sendo que os clientes teriam a opção de

comprar a extensão já completa ou implementar uma nova. Um exemplo para um *plugin* que poderia ser vendido é a integração com o *SAP* [23].

Para além dos pontos onde poderiam ser adicionadas novas funcionalidades, o projeto em si poderia ser melhorado. Mais especificamente, um ponto que poderia ser melhorado é a documentação de todas as funcionalidades dos edocHooks. Sendo este módulo bastante extensível, existem algumas funcionalidades que podem ser realizadas mas que não foram introduzidas intencionalmente, apenas existindo devido ao conjunto de outras funcionalidades do sistema. Estes pontos deveriam ser estudados de um modo mais profundo de modo a dar uma melhor ideia ao cliente do potencial dos edocHooks.

Outra funcionalidade em falta que foi descoberta depois da implementação deste projeto é a falta de transacionalidade nos edocHooks. Esta funcionalidade nos triggers era-me desconhecida, no entanto percebi que era bastante importante para alguns clientes. A alteração necessária para a implementação desta funcionalidade é bastante baixa visto que o cliente já responde ao edocHook com uma verificação de falhas. Com isto, apenas é necessário criar uma transação quando existe um edocHook e dar rollback no caso deste retornar algum erro.

Bibliografia

[1] R. Appelqvist e O. Örnmyr. *Performance comparison of XHR polling, Long polling, Server sent events and Websockets*. 2017 (ver p. 11).

[2] *Azure DevOps*. URL: https://azure.microsoft.com/en-us/products/devops (ver p. 8).

[3] M. Biehl. *Webhooks – Events for RESTful APIs*. API-University Press, 2017-12 (ver p. 11).

[4] *Bootstrap*. 2023. URL: https://getbootstrap.com/ (ver p. 64).

[5] S. Bui. «Micro frontend: Microservice implementation on Web Development». Em: (2021) (ver p. 29).

[6] *Camunda Form Builder*. 2023. URL: https://camunda.com/platform/modeler/#forms (ver p. 10).

[7] F. Chong, G. Carraro e R. Wolter. «Multi-tenant data architecture». Em: *MSDN Library, Microsoft Corporation* (2006), pp. 14–30 (ver p. 2).

[8] *Exemplo de uma página de documentação storybooks*. 2023. URL: https://5fb5253d9fbaa1002177207a-pfgnjdmtcy.chromatic.com/?path=/story/design-system-components-sbaccordion--default (ver p. 31).

[9] M. Geers. *Micro Frontends*. URL: https://micro-frontends.org/ (ver p. 13).

[10] *How signatures work*. URL: https://en.wikipedia.org/wiki/Digital_signature#/media/File:Private_key_signing.svg (ver p. 45).

[11] *How webhooks work*. URL: https://shopify.dev/assets/apps/how-webhooks-work-4241223c92b29784163b8f069e7e55fbd57147821a7664805b870fd0099eb51f.png (ver p. 12).

[12] K. Indrasiri. *Beginning WSO2 ESB*. Apress, 2016 (ver p. 18).

[13] *JotForm*. URL: https://www.jotform.com/ (ver pp. 10, 11).

[14] *Kaspersky- Defending Yourself from a Man in the Middle Attack*. 2023. URL: https://www.kaspersky.com/resource-center/threats/man-in-the-middle-attack (ver p. 44).

[15] *KendoUI Angular by Telerik*. 2023. URL: https://www.telerik.com/kendo-angular-ui (ver p. 64).

[16] *Keycloak*. 2023. URL: https://www.keycloak.org/ (ver p. 59).

[17] *Module Federation*. 2023. URL: https://webpack.js.org/concepts/module-federation/ (ver p. 47).

[18] *NginX*. 2023. URL: https://www.nginx.com/ (ver p. 29).

[19] *OneDev - Self-hosted Git Server with CI/CD and Kanban*. 2023. URL: https://onedev.io/ (ver p. 9).

[20] *Página incial do website da ferramenta StoryBook*. URL: https://storybook.js.org/ (ver p. 30).

[21] S. Peltonen, L. Mezzalira e D. Taibi. «Motivations, benefits, and issues for adopting micro-frontends: a multivocal literature review». Em: *Information and Software Technology* 136 (2021), p. 106571 (ver p. 13).

[22] C. M. Rouven Krebs e S. Kounev. «Architectural Concerns in Multi-Tenant SaaS Applications». Em: *MSDN Library, Microsoft Corporation* (2006), pp. 14–30 (ver p. 5).

[23] *SAP - Software de Gestão*. 2023. URL: https://www.sap.com/portugal/index.html (ver p. 72).

[24] S. Satyanarayana. «Cloud computing: SAAS». Em: *Computer Sciences and Telecommunications* 4 (2012), pp. 76–79 (ver p. 5).

[25] L. Schwenke. *O que é idempotência e porque devemos utilizar*. 2019-02. URL: https://xuenqui.medium.com/idempot%C3%AAncia-uma-boa-pr%C3%A1tica-a-se-utilizar-em-servi%C3%A7os-rest-633c38f4d7c0 (ver p. 46).

[26] *Tabulator*. 2023. URL: https://tabulator.info/ (ver p. 64).

[27] *Three Teams Microfrontend*. URL: https://micro-frontends.org/ressources/screen/three-teams.png (ver p. 14).

[28] *TRUMBOWYG: A lightweight WYSIWYG editor*. 2023. URL: https://alex-d.github.io/Trumbowyg/ (ver p. 35).

[29] *Verdaccio*. 2023. URL: https://verdaccio.org/ (ver p. 61).

[30] V. Wang, F. Salim e P. Moskovits. *The WebSocket Protocol*. Apress, 2013 (ver p. 11).

[31] *Webhook Security Vulnerabilities Guide*. URL: https://hookdeck.com/webhooks/guides/webhook-security-vulnerabilities-guide (ver p. 44).

[32] *Webhooks documentation*. 2023. URL: https://docs.github.com/en/webhooks (ver p. 12).

[33] *When to use Webhooks, websocket, pub/sub, and polling*. URL: https://hookdeck.com/webhooks/guides/when-to-use-webhooks (ver p. 12).

[34] *YouTrack - Powerful project management for all your teams*. 2023. URL: https://www.jetbrains.com/youtrack/ (ver p. 9).