INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

PipelineManager

Rui Pedro Dias de Sousa Faria

Relatório de Projeto apresentado à Escola Superior de Tecnologia e Gestão para obtenção do Grau de Mestre em Informática

Trabalho realizado sob a orientação de:

João Paulo Pereira

Bragança
Outubro de 2023

Resumo

A gestão de tarefas e a colaboração entre as equipas de trabalho numa empresa de forma eficaz desempenham um papel fundamental para o seu sucesso. E, por isso, na constante evolução da empresa, existe necessidade de manter e melhorar esse papel a cada dia.

Assim sendo, o objetivo deste projeto é desenvolver uma solução que permita aos colaboradores de uma empresa a gestão do seu pipeline de trabalho de forma eficiente e aos team leaders a gestão da equipa, facilitando a tomada de decisão. Desta forma, existirá uma otimização do fluxo de trabalho e a empresa terá uma melhor monitorização e controlo no progresso das suas atividades.

A implementação deste projeto envolveu a utilização da tecnologia SAP Low Code No Code para o desenvolvimento da aplicação, permitindo a integração com o ambiente SAP e facilitando a sua criação.

A solução implementada utilizou uma variedade de dados não oficiais, sendo estes guardados e organizados em tabelas criadas no ambiente SAP.

Em suma, a solução final resultou numa aplicação altamente funcional e eficaz que simplifica a gestão dos pedidos dos colaboradores. A automação de notificações por emails em caso de inatividade ou atualização dos pedidos é uma funcionalidade que veio somar no que toca à otimização do fluxo de trabalho. Além disso, o envio de um email periódico para o team leader de um relatório com os pedidos de todos os colaboradores fornece uma visão abrangente do estado geral do trabalho.

Palavras Chave: SAP, SAP Low Code No Code, Gestão

Abstract

The effective management of tasks and collaboration between work teams in a company plays a fundamental role in its success. And so, in the constant evolution of the company, there is a need to maintain and improve this role every day.

Therefore, the aim of this project is to develop a solution that allows a company's employees to manage their work pipeline efficiently and team leaders to manage the team, facilitating decision-making. In this way, workflow will be optimised and the company will have better monitoring and control over the progress of its activities.

The implementation of this project involved using SAP Low Code No Code technology to develop the application, enabling integration with the SAP environment and facilitating its creation.

The implemented solution used a variety of unofficial data, which was stored and organised in tables created in the SAP environment.

In short, the final solution resulted in a highly functional and effective application that simplifies the management of employee requests. The automation of email notifications in the event of inactivity or order updates is a feature that has added to the optimisation of the workflow. In addition, sending a periodic email to the team leader with a report of all employees' requests provides a comprehensive overview of the overall status of work.

Índice Geral

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Capítulo 1 Introdução

O presente relatório retrata o trabalho desenvolvido no âmbito da Unidade Curricular de Projeto, do Mestrado em informática do Instituto Politécnico de Bragança (IPB).

Atualmente, a eficiência e gestão das tarefas em processos empresariais apresentam um papel crucial no sucesso de qualquer organização. Para atender a essas necessidades, este projeto aborda a criação de uma aplicação no ambiente SAP Low Code No Code (SAP, Low Code No Code) e o desenvolvimento de automatizações que enviam emails periódicos aos colaboradores para atualizarem o pipeline ou alertarem para a inatividade na aplicação. A finalidade principal deste projeto é permitir aos colaboradores de uma empresa criarem e monitorizarem o progresso do seu pipeline de trabalho de forma eficaz. Também permite aos team leaders, como administradores, adicionarem novos utilizadores à aplicação e terem uma visão geral de tudo, no que toca a pedidos, utilizadores e estatísticas da aplicação.

Desta forma, as empresas conseguirão otimizar e controlar melhor o seu fluxo de trabalho.

1.1. Enquadramento

Num ambiente empresarial em constante evolução, a necessidade de acompanhar e otimizar o fluxo de trabalho torna-se cada vez mais presente a cada dia. De forma a confrontar esses desafios e tirar proveito das tecnologias emergentes, as empresas tentam

obter soluções inovadoras que simplifiquem o processo de gestão de trabalho e, ao mesmo tempo, proporcionem maior visibilidade e controlo sobre o progresso das suas atividades.

Desta forma, na implementação deste projeto adotou-se a tecnologia SAP Low Code No Code para a criação de uma aplicação que permita não só aos colaboradores de uma empresa a gestão do seu trabalho de forma eficaz e otimizada, mas também aos teams leaders uma melhor monitorização dos pedidos aprimorando, assim, a tomada de decisão.

Como resultado final espera-se uma solução capaz de responder às necessidades das empresas que pretendem simplificar e otimizar o progresso das suas atividades.

1.2. Objetivos

O objetivo principal deste projeto é a criação de uma solução que permita otimizar a gestão do pipeline de trabalho dos colaboradores de uma empresa, proporcionando uma visão clara e eficiente das atividades realizadas.

Além disso, de forma a auxiliar as tomadas de decisões dos team leaders que visem diminuir perdas de tempo e aumentar visibilidade na monitorização dos pedidos, a solução deverá ainda permitir o envio de emails com um relatório de todos os pedidos de forma automatizada.

1.3. Estrutura do documento

Este relatório encontra-se organizado da seguinte forma:

No capítulo 1 é explicado de uma forma geral os objetivos deste trabalho e a sua

estrutura.

No capítulo 2 descreve-se o estado de arte.

No capítulo 3 descrevem-se as ferramentas utilizadas para o desenvolvimento da aplicação;

No capítulo 4 são apresentados os requisitos funcionais e não funcionais, diagramas de casos de uso, mockups e modelo de base de dados.

No capítulo 5 são descritas as etapas de desenvolvimento, desde a implementação do necessário em SAP até à construção da aplicação na plataforma SAP Build Apps.

No capítulo 6 apresenta-se uma breve demonstração da aplicação.

No último capítulo é apresentada uma breve conclusão do trabalho com referências a documentos que serviram de ajuda para o auto-conhecimento.

Capítulo 2 Estado de Arte

Neste capítulo, abordar-se-á os principais estudos e diversas pesquisas feitas para o desenvolvimento deste projeto que moldam a idealização do mesmo.

2.1. Sistema ERP

Um ERP, *Enterprise Resource Planning*, é um sistema de gestão empresarial completo e integrado que permite a otimização do fluxo de informação entre as diversas áreas de uma empresa como a gestão financeira, recursos humanos, gestão da cadeia de fornecimento e produção, entre outros. Este sistema, on-premise ou cloud, permite automatizar a comunicação entre os diferentes departamentos de forma eficiente, diminui custos e ajudam na tomada de decisões. (Infowester)

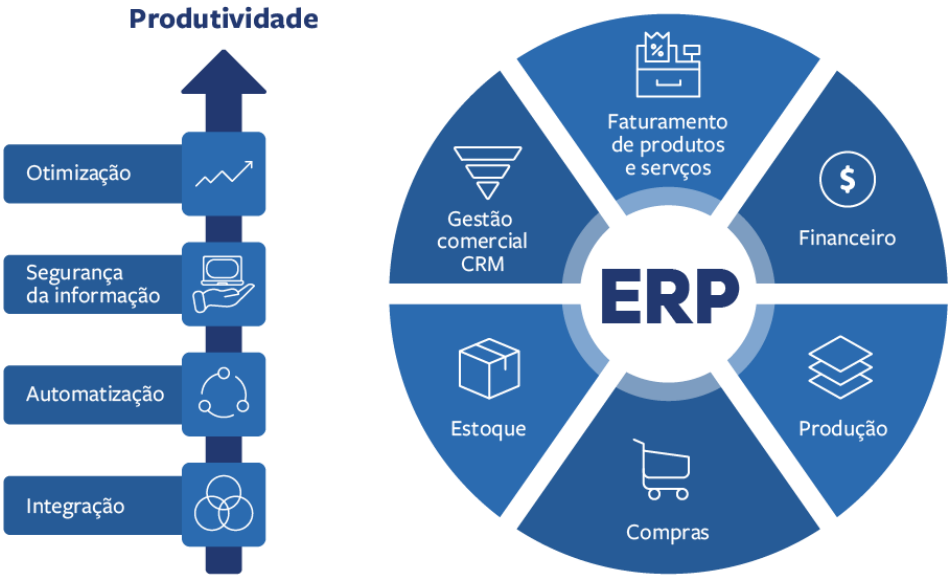


Figura 1 - Sistema ERP (Omie)

Existem várias ferramentas ERP no mercado: PRIMAVERA Business Software Solutions, AS400 ERP da IBM, que podem ser vistos como os principais concorrentes ao software da SAP, o WebERP, o Microsoft Dynamics ou o FreedomERP. No âmbito deste projeto foi utilizado o SAP ERP fornecido pela empresa SAP.

2.2. SAP Solution Manager

O SAP Solution Manager permite aos clientes gerir melhor as suas aplicações SAP e não-SAP. Permite centralizar, melhorar, automatizar e aperfeiçoar a gestão de todo o panorama do sistema, reduzindo assim o custo total de propriedade. O SAP Solution Manager também apoia os clientes na adaptação do ambiente a novos requisitos, por exemplo, na implementação de novos processos empresariais (SAP, SAP Solution Manager).

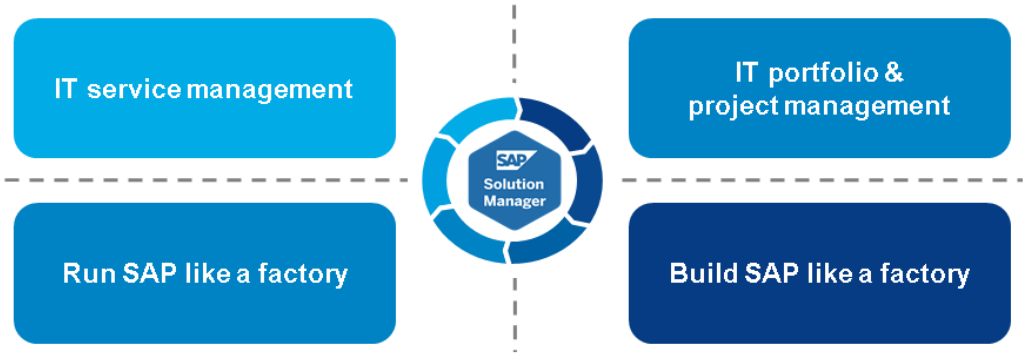


Figura 2 - SAP Solution Manager's key functional building blocks

O bloco project management é o que assemelha realmente à idealização do deste projeto, uma vez que permite toda uma gestão de atividades e monitorização do fluxo de trabalho.

Capítulo 3 Tecnologias e Ferramentas utilizadas

Neste capítulo indicar-se-ão as ferramentas e tecnologias essenciais para a realização deste projeto.

3.1. SAP

A SAP, ou Sistema de Aplicação e Produtos em Processamento de Dados, é uma plataforma de software amplamente utilizada por empresas para otimizar várias áreas de gestão, como finanças, recursos humanos, logística e produção. Essa ferramenta permite que as empresas melhorem a integração de informações e processos, o que facilita a tomada de decisões e a eficiência operacional. O SAP é uma solução valiosa para empresas que desejam melhorar o controlo e a gestão das suas atividades.

Tudo isto começou em 1972 com a empresa SAP SE cujo o nome completo na altura era "Desenvolvimento de Programas de Análise de Sistemas" e mais tarde abreviado para SAP. Inicialmente, era uma pequena empresa com cinco pessoas, mas evoluiu para uma multinacional em Walldorf, na Alemanha, com mais de 105.000 funcionários em todo o mundo. A SAP estabeleceu o padrão global para software de planeamento de recursos empresariais (ERP) com o lançamento de seu software original SAP R/2 e SAP R/3. Atualmente, o SAP S/4HANA leva o ERP a um novo patamar, aproveitando a computação in-memory para processar grandes volumes de dados e suportar tecnologias avançadas, como inteligência artificial (IA) e machine learning. (SAP, Plataform SAP)

3.2. SAP Abap

O SAP Abap, Advanced Business Application Programming (Press, ABAP Programming for SAP), é uma linguagem de programação criada pela SAP para o desenvolvimento de soluções personalizadas dentro do ambiente SAP.

É uma linguagem versátil e robusta que desempenha um papel fundamental no SAP, uma vez que permite responder de forma eficaz às necessidades de cada empresa. Usada para criar extensões das soluções SAP, relatórios personalizados, interfaces de utilizadores, integrações de tecnologias modernas, manipulação de dados, entre outras soluções que permitem melhorar a experiência dos utilizadores finais aquando das suas operações nas variadas transações do SAP.

3.3. Adobe Forms, Smart Forms e SAPscript

Os Adobe forms, Smart Forms e SAPscripts são tecnologias usadas no ambiente SAP para criar formulários personalizados, relatórios e documentos empresariais que podem ser impressos em determinadas transações SAP. Cada uma dessas tecnologias tem suas próprias características e utilizações específicas. (SAP, Smartforms), (SAP, Adobe forms), (TutorialsPoint)

3.4. Rest WebService

REST ou RESTful, Representational State Transfer, é uma solução arquitetural para construir sistemas de comunicação de rede, sistematicamente usado para criar WebServices.

Desta forma, os WebServices RESTful são uma forma de permitir que diferentes sistemas se comuniquem pela Internet, compartilhando dados entre si.

WebServices REST suportam dados em formato de JSON ou XML.
Cada função no REST WebService utiliza um identificador de recurso uniforme (URI) de *endpoint* único. O serviços de dados exige estritamente que cada função tenha um esquema fixo definido por XSD (XML Schema Definition). Com base nos métodos

HTTP, o serviço de dados pode então obter ou modificar recursos. Alguns dos métodos HTTP mais comuns incluem GET (obter dados), PUT (atualiza dados), POST (cria novos dados) e DELETE (elimina dados) (SAP, REST WebService).

3.5. SAP Cloud Connector

SAP Cloud Connector é uma aplicação que permite a criação de uma conexão segura à SAP Cloud, de forma que todos os serviços em Cloud possam comunicar de forma segura com os sistemas de um cliente *on-premise/*privado *Cloud landscape* (SAP A. M.).

De forma clara, esta tecnologia permite expor os serviços privados do cliente de forma publica utilizando outro serviço "por cima" que se intitula como serviço externo.

3.6. SAP Cloud Platform Integration

SAP Cloud Platform Integration é um serviço intermediário baseado em cloud que permite a integração entre sistemas cloud e aplicações on-premise com serviços externos SAP ou não. Trata-se de uma plataforma como serviço (Hanfi).

3.7. Low Code No Code

Low Code No Code são métodos de conceção e desenvolvimento de aplicações utilizando ferramentas intuitivas de "drag and drop" que reduzem exponencialmente ou eliminam a necessidade de programadores tracidionais (SAP, Low-code/no-code application development).

A escolha da tecnologia SAP Low Code No Code para o desenvolvimento dessas soluções é motivada por diversos fatores, como agilidade no de

- Agilidade no Desenvolvimento;

- Maior autonomia;
- Integração com SAP;
- Facilidade de Uso.

3.8. SAP BTP

A plataforma aberta da SAP funciona como um serviço para o desenvolvimento de aplicações empresariais na cloud num ambiente totalmente provisionado.

Utilizando um conjunto de serviços, capacidades e ferramentas de ponto a ponto, os programadores podem criar, alargar e integrar aplicações empresariais na cloud (SAP, SAP BTP).

3.8.1. SAP BTP Cockpit

O SAP BTP Cockpit é uma interface de utilizador central baseada na Web para administradores, que fornece acesso a uma série de funções para configurar e gerir aplicações e ligá-las a serviços no SAP BTP.

A utilização do cockpit permite gerir recursos, serviços, segurança, monitorizar métricas de aplicações e executar aplicações na nuvem (SAP, SAP BTP).

3.9. SAP Build Apps

O SAP Build Apps é uma solução de desenvolvimento de aplicações profissionais especificamente desenhada para que qualquer pessoa possa rapidamente criar aplicações multiplataforma web ou mobile sem código, independentemente do nível de competências.

Esta solução contém centenas de modelos e componentes de lógica empresarial pré-construídos, de forma a permitir impulsionar os projetos de aplicações e reduzir o tempo de desenvolvimento (SAP, SAP Build Apps).

3.10. JSON

O JSON (JavaScript Object Notation) é um formato leve de troca de dados. É fácil para os humanos lerem e escreverem. É fácil para as máquinas analisarem e gerarem. Baseia-se num subconjunto da norma de linguagem de programação JavaScript ECMA-262 3rd Edition - dezembro de 1999. O JSON é um formato de texto que é completamente independente da linguagem, mas utiliza convenções que são familiares aos programadores da família de linguagens C, incluindo C, C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem do JSON uma linguagem de troca de dados ideal (JSON)

3.11. Figma

O Figma é uma plataforma colaborativa para a criação de designs de interfaces e protótipos, da empresa Figma, Inc., lançada em 2016 por Dylan Field e Evan Wallace. O objetivo era de criar uma ferramenta gratuita que trouxesse colaboração entre pessoas e equipas, permitindo criar um produto para as mais diversas plataformas, mantendo a acessibilidade do sistema. (Alura)

3.12. Visual Paradigm

O Visual Paradigm é uma ferramenta de conceção e gestão de sistemas de TI, multiplataforma e fácil de utilizar. Este fornece aos programadores de software uma plataforma de desenvolvimento para criar aplicações de qualidade de forma mais rápida. Facilita uma excelente interoperabilidade com outras ferramentas CASE e com a maioria dos IDEs líderes, o que torna excelente todo o seu processo de desenvolvimento Modelo-Código-Desenvolvimento. (Visual Paradigm)

Capítulo 4 Arquitetura e Modelação do sistema

Este capítulo apresenta toda a arquitetura e modelação desenvolvidas para o processo de implementação do projeto.

Assim sendo, demonstra-se os requisitos funcionais e não funcionais, diagrama de casos de uso, modelo de base de dados e mockups, com o objetivo de ilustrar com mais clareza e detalhe o funcionamento da aplicação e as atividades que podem ser realizadas pelos utilizadores.

4.1. Requisitos

4.1.1. Funcionais

4.1.1.1. Ver pedidos

- **Ator**: qualquer utilizador;
- **Descrição**: A aplicação deve permitir a qualquer utilizador ver os seus pedidos.

4.1.1.2. Ver detalhes de um pedido

- **Ator**: qualquer utilizador;
- **Descrição**: A aplicação deve permitir a qualquer utilizador ver os detalhes de cada pedido.

4.1.1.3. Eliminar pedidos

- **Ator**: qualquer utilizador;
- **Descrição**: A aplicação deve permitir a qualquer utilizador eliminar os pedidos da sua lista.

4.1.1.4. Editar pedido

- **Ator**: qualquer utilizador;
- **Descrição**: A aplicação deve permitir a qualquer utilizador editar os pedidos da sua lista. Na edição deve permitir adicinonar ficheiros.

4.1.1.5. Ver Definições

- **Ator**: qualquer utilizador;
- **Descrição**: A aplicação deve permitir a qualquer utilizador ver as definições da aplicação.

4.1.1.6. Alterar Definições

- **Ator**: qualquer utilizador;
- **Descrição**: A aplicação deve permitir a qualquer utilizador alterar as definições da aplicação;

4.1.1.7. Ver estatísticas

- **Ator**: admin;
- **Descrição**: A aplicação deve permitir aos utilizadores admin ver as estatísticas da aplicação.

4.1.1.8. Ver utilizadores existentes

- **Ator**: admin;
- **Descrição**: A aplicação deve permitir aos utilizadores admin ver os utilizadores existentes na mesma.

4.1.1.9. Eliminar utilizadores

- **Ator**: admin;

- **Descrição**: A aplicação deve permitir aos utilizadores admin eliminar utilizadores da mesma.

4.1.1.10. Adicionar utilizadores

- **Ator**: admin;
- **Descrição**: A aplicação deve permitir aos utilizadores admin adicionar novos utilizadores.

4.1.1.11. Enviar email automático

- **Ator**: admin;
- **Descrição**: A aplicação deve permitir aos utilizadores admin enviar um email automático através de um botão.

4.1.1.12. Envio de email automático com relatório de pedidos

- **Ator**: SAP;
- **Descrição**: O sistema deve permitir o envio de emails periódico com o relatório dos pedidos.

4.1.1.13. Envio de email automático de inatividade

- **Ator**: SAP;
- **Decrição**: O sistema deve efeturar verificações periódicas de inativadade dos utilziadores e permitir o envio de emails por inativadade do utilizador.

4.1.2. Não Funcionais

4.1.2.1. Desempenho da aplicação

A aplicação deve apresentar informação em tempo real e ter um bom desempenho ao ter ações do utilizador.

4.1.2.2. Fiabilidade da aplicação

Os dados principais devem estar presentes na base de dados da aplicação e numa base de dados local, em vez de uma tabela interna. O envio de emails deve ser feito de forma segura.

4.1.2.3. Usabilidade

O Sistema deve garantir que a aplicação apresenta uma boa UX (*User Experience*).

4.1.2.4. Interface simples e intuitiva

A aplicação deve ter uma interface graficamente simples, agradável e fácil de compreender.

4.2. Diagrama de Casos de Uso

Os Diagramas de Casos de Uso são uma ferramenta essencial na modelagem de sistemas e no desenvolvimento de aplicações. Permitem ter uma representação visual das interações entre um sistema e os seus atores, ajudando a identificar as principais funcionalidades do sistema e como os utilizadores podem interagir com ele. Destacam as ações que um sistema pode executar em resposta às solicitações dos atores. São fundamentais para facilitar a compreensão e a passagem de conhecimento dos requisitos do sistema de forma clara e concisa. Assim, nesta secção apresentar-se-á os diagramas de casos de uso usados para representar os requisitos e interações entre atores e sistemas.

O diagrama abaixo, apresenta todas as funcionalidades que o utilizador com autorização de user pode usar na aplicação.

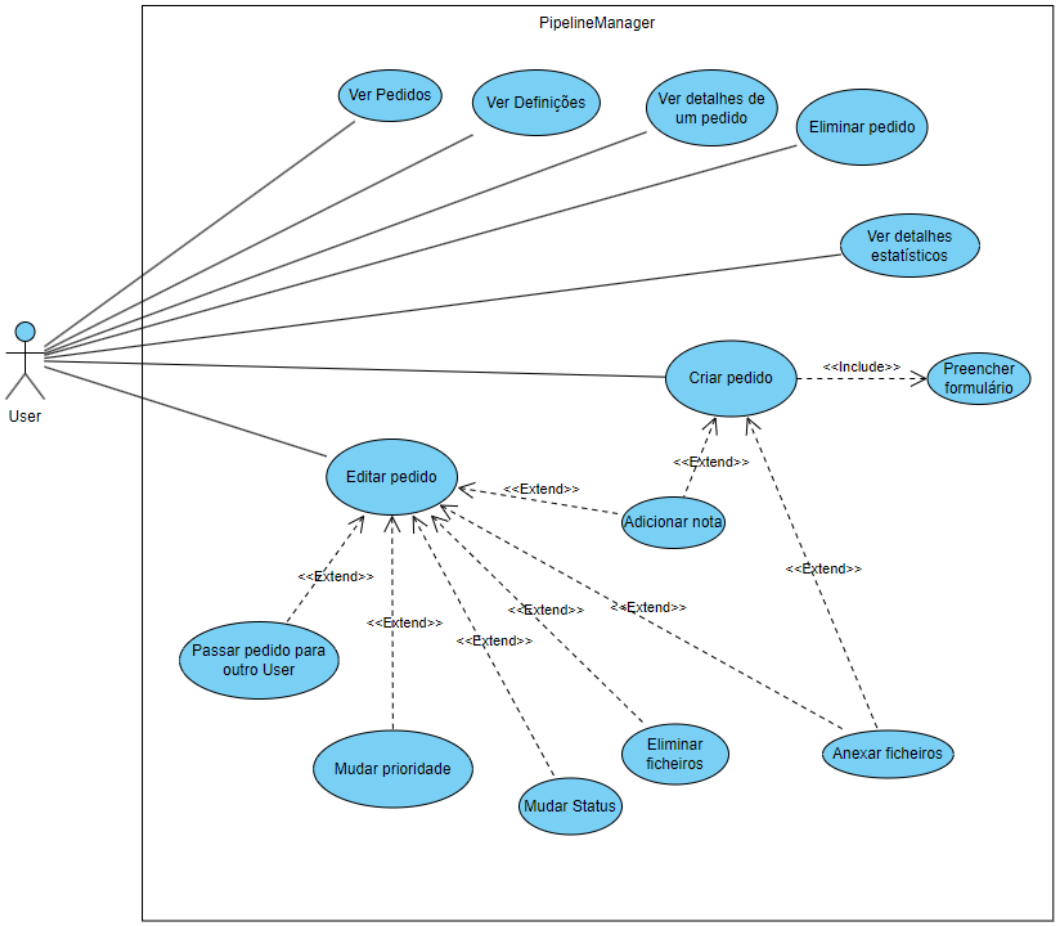


Figura 3 - Diagrama - Interação entre User e APP

O próximo diagrama é uma extensão do diagrama em cima, uma vez que o autor trata-se de uma utilizador com autorização de admin. Sendo assim, existem as mesmas funcionalidades que no diagrama anterior e mais algumas que permitem o controlo do utilizador com autorizações de user. Controlo esse que compreende, a eliminação e adição de utilizadores.

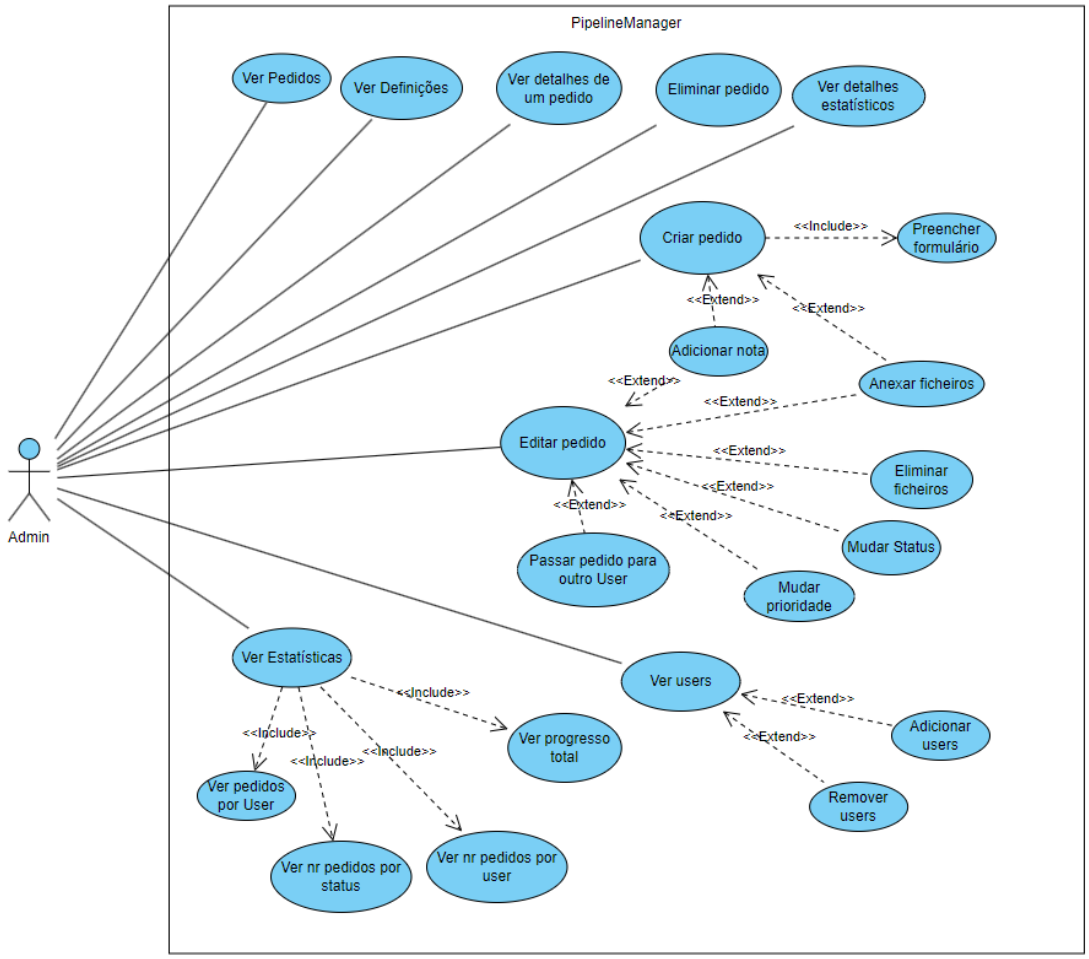


Figura 4 - Diagrama - Interação de Admin e App

O Diagrama abaixo, representa a comunicação entre os atores aplicação e base de dados SAP por meio do WebService REST, através do métodos implementados na classe *handler* do WebService REST.

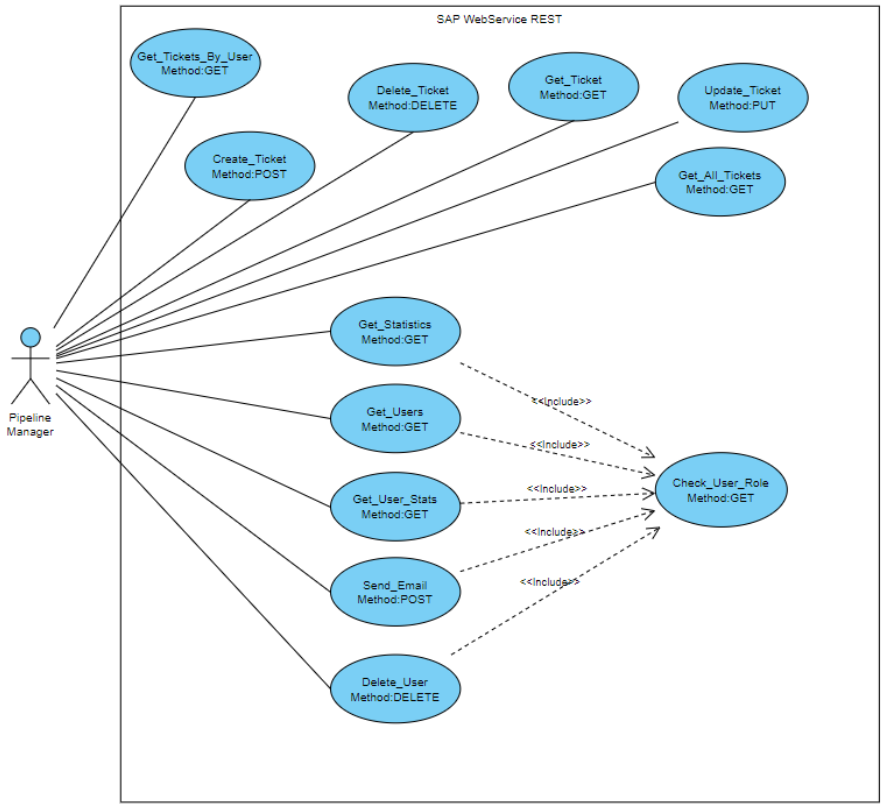


Figura 5 - Diagrama - Interação entre App, Webservice e Base de dados

O Diagrama abaixo representa a comunicação de um utilizador interno da SAP com os autores base de dados e Serviço SMTP através do programa desenvolvido em SAP.

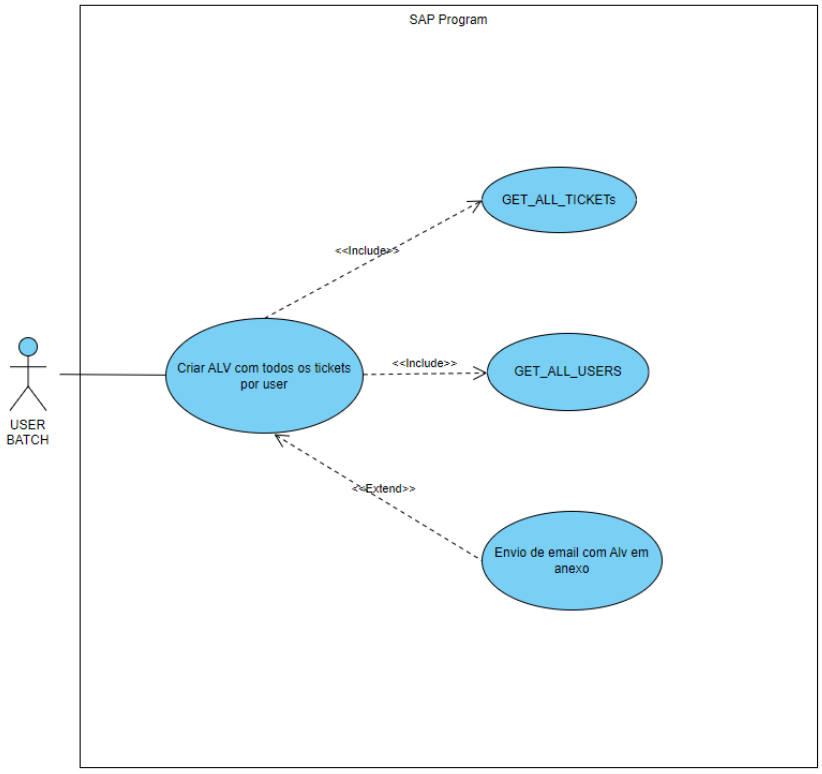


Figura 6 - Diagrama - Programa de PDS

O Diagrama abaixo representa a comunicação de um utilizador interno da SAP com os autores base de dados e Serviço SMTP através do programa desenvolvido em SAP.

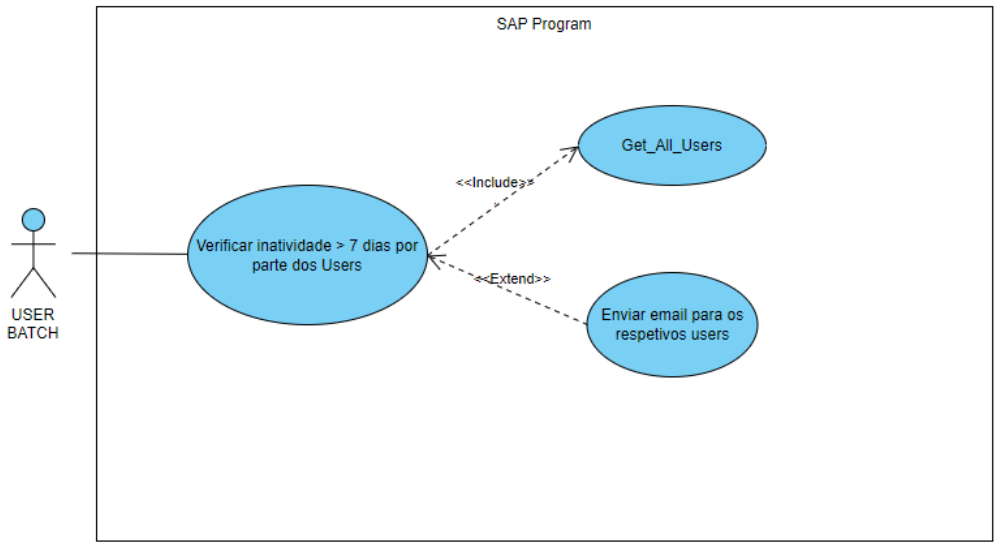
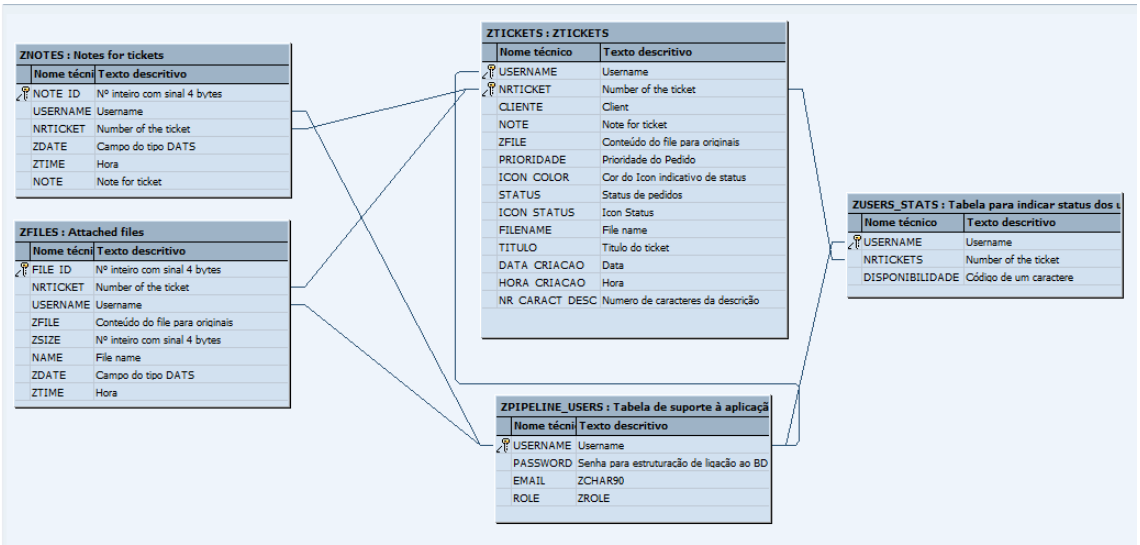


Figura 7 - Diagrama - Programa de Inatividade de Users

4.3. Modelo de base de dados

No que toca ao Modelo de base de dados, este tem como objetivo a representação estruturada e organizada dos dados que um sistema necessita para armazenar e gerir. A criação da base de dados e representação foi feita em SAP, sendo que as tabelas para suportar a base de dados foram criadas na transação SE11. Relativamente à representação do modelo de base de dados foi feita com o auxílio da transação SQVI (Press, How SAP Consultants Create Queries With These Three T-Codes) que estabelece a relação entre as tabelas criadas.

Neste modelo é possível verificar que a tabela principal é a ZTICKETS, uma vez que é nela que vai ser guardado o conteúdo principal da aplicação e que vai servir de base para os dados das outras tabelas, nomeadamente da ZNOTES, ZFILES e ZUSERS_STATS. Por fim, a tabela que vai conter os dados de acesso dos utilizadores e define todas as autorizações referentes à aplicação é a ZPIPELINE_USERS.



4.4. Mockups

As Mockups são uma forma visual de apresentar o design de uma aplicação, geralmente criadas de forma rápida e esquemática. São representações gráficas que têm como objetivo transmitir a aparência e a estrutura do projeto, permitindo á partida a compreensão daquilo que será o projeto final.

Assim sendo, nesta secção apresentar-se-á as mockups referentes a este projeto de todas as páginas desenvolvidas, Login, Pipeline, Create Ticket, Ticket Details, Settings, Statistics e Users.

Na página de Login, como o nome indica, permite ao utilizador entrar na aplicação com os seus dados de acesso dados pela organização em que se encontra.

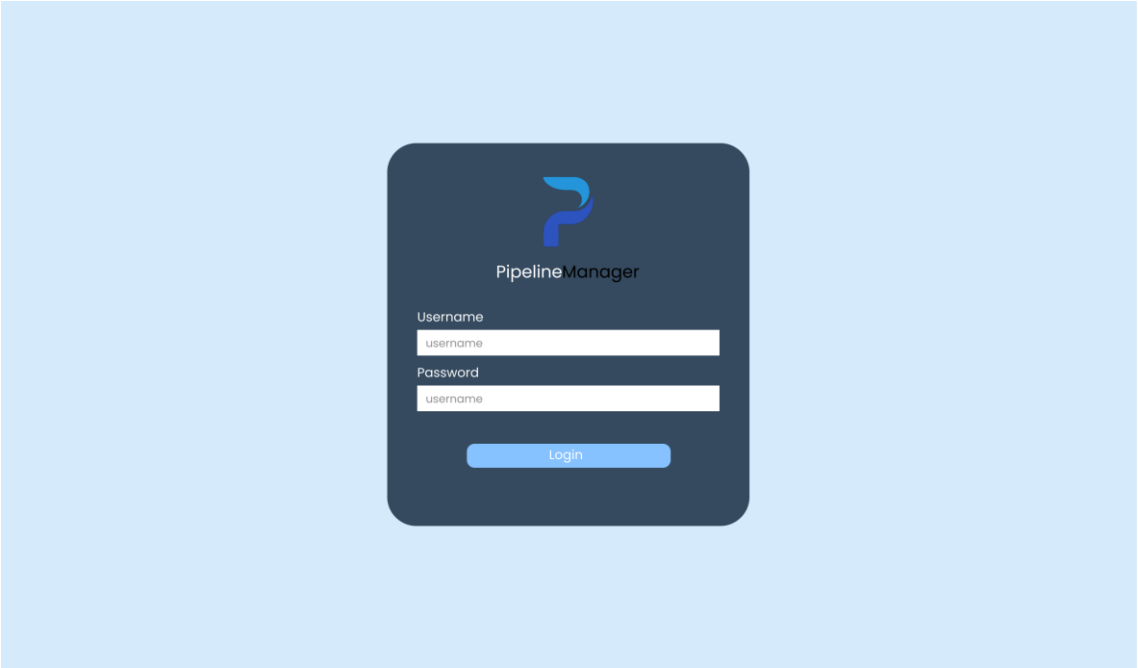


Figura 8 - Página de Login

Relativamente à página de Pipeline, é possível ver e eliminar todos os pedidos, assim como clicar num pedido para verificar os seus detalhes. Além disso, o utilizador pode verificar também estatísticas relativas ao seu progresso, como o número de pedidos, pedidos completos tendo em conta o total existente na lista, pedidos por fazer e pedidos por estado.

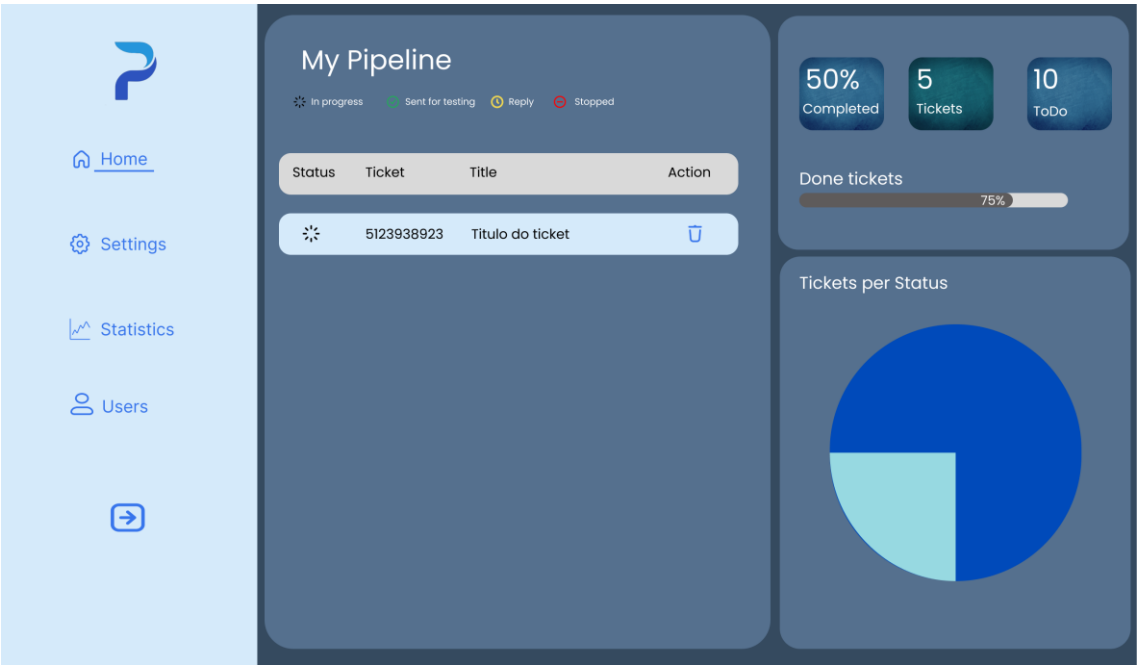


Figura 9 - Página do Pipeline

No que toca à página de adicionar um pedido, o utilizador pode preencher o formulário e adicionar ficheiros.

Figura 10 - Página da Criação do Pedido

Na página de detalhes de um pedido, o utilizador consegue verificar detalhes como o número do pedido e título, data e hora de criação, utilizador com o pedido, o estado, a prioridade, as notas do pedido e ficheiros anexados. Além disso, a página permite ao utilizador alternar para o modo de edição, podendo alterar o utilizador, estado, prioridade, adicionar notas e ficheiros.

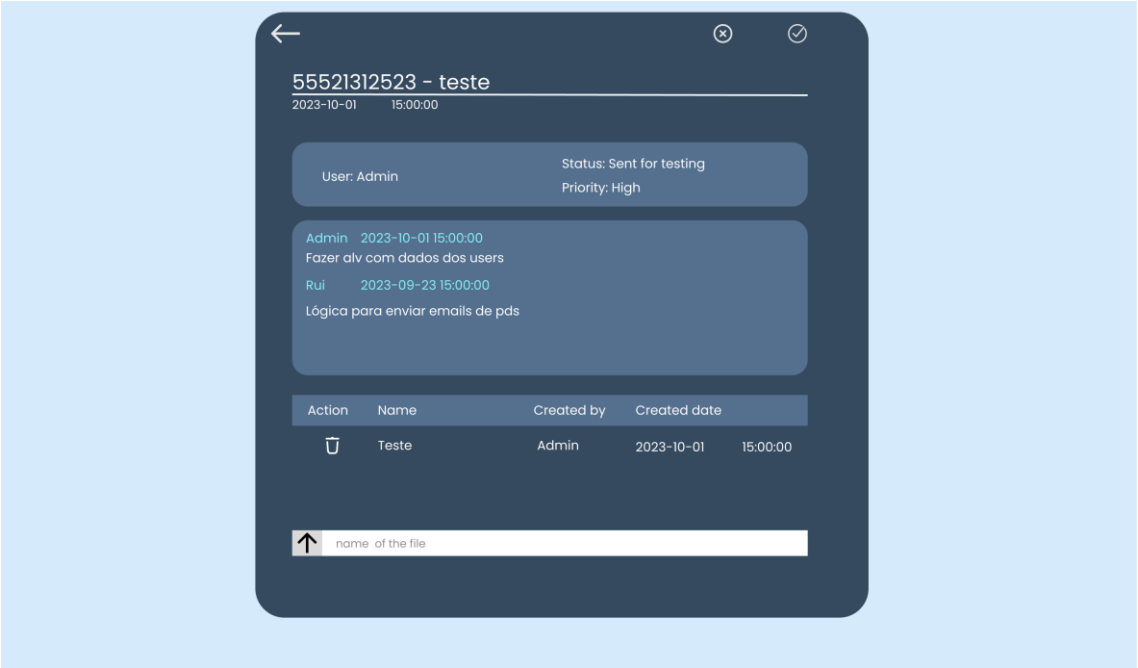


Figura 11 - Páginas de detalhes de um pedido

Nas definições, o utilizador tem a possibilidade de traduzir a aplicação entre os idiomas EN (Inglês) e PT (Português).

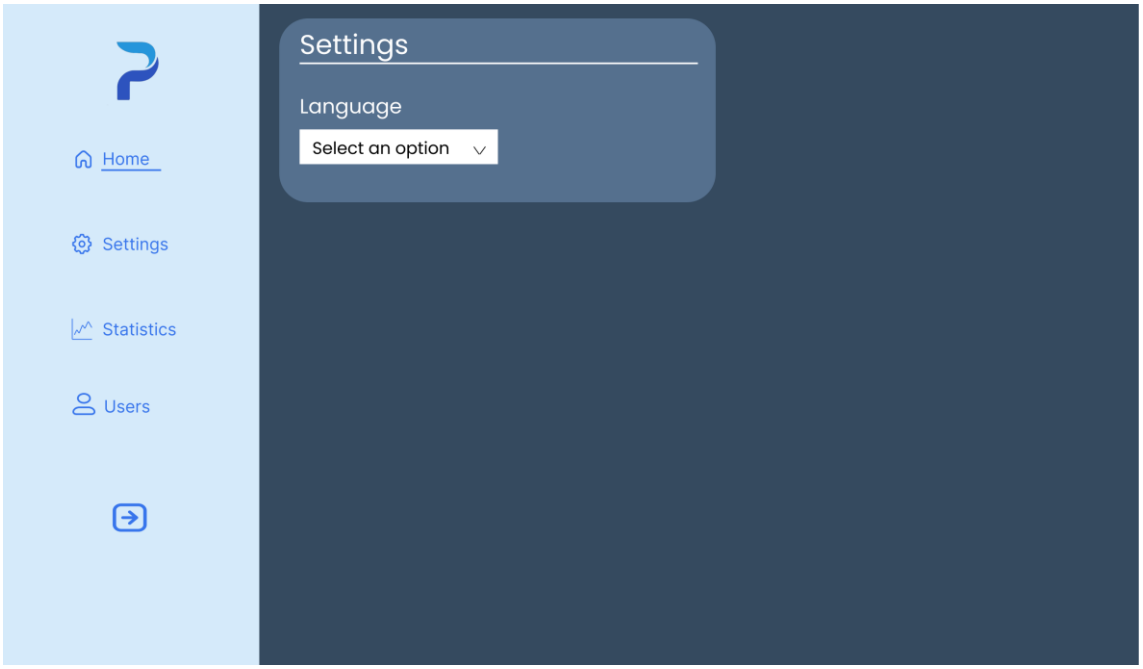


Figura 12 - Página de Definições

Na página de Estatísticas, o utilizador com autorização de admin consegue verificar todo o tipo de estatísticas relativas à aplicação. Isto é, progresso dos pedidos em percentagem, número de pedidos por estado, pedidos por utilizador em percentagem e detalhes de todos pedidos de todos os utilizadores.

Figura 13 - Página de Estatísticas

Finalmente, na página Users, o utilizador com autorização de admin pode ver e eliminar todos os utilizadores que existem na aplicação e adicionar novos. Além disso, se desejar pode enviar uma email de alerta para todos os utilizadores atualizarem o ponto de situação, ou seja, a sua lista de pedidos

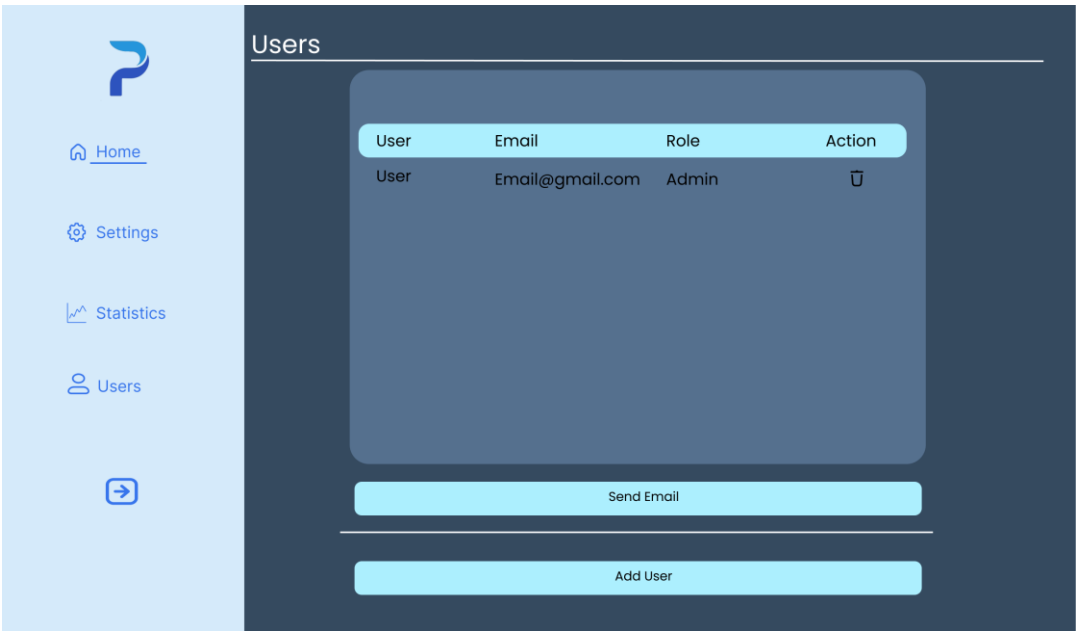


Figura 14 - Página de Administração de utilizadores

Capítulo 5 Proposta/Desenvolvimento/ Trabalho efetuado

5.1. Introdução

No decorrer deste capítulo apresentar-se-á o processo de criação da aplicação PipelineManager. Até este ponto, foram explicados conceitos, objetivos e todo um planejamento para o desenvolvimento. Agora, será o ponto onde dar-se-á vida à idealização, transformando linhas de código e ideias numa aplicação funcional e eficaz, desde a criação do necessário para a comunicação do SAP com a APP até à respetiva construção da APP.

O desenvolvimento é uma fase crucial que transforma o idealizado em realidade digital. Neste capítulo, abordar-se-á os aspetos técnicos e práticos do processo, desde a escolha das tecnologias até a codificação, lógica da app e testes. Será uma fase que exigirá criatividade, habilidade técnica e dedicação para concretizar a idealização feita.

Aqui, serão descritas as etapas de desenvolvimento, discutir os desafios encontrados e destacar as soluções criativas que permitiram criar uma aplicação que não atenda apenas às expectativas, mas que as supere.

Todo o código referido neste capítulo pode ser consultado em anexo deste documento.

5.2. Etapas de Desenvolvimento

De forma a fornecer uma compreensão detalhada do desenvolvimento da aplicação, nesta secção apresentar-se-ão as etapas que levaram uma fase inicial ao produto final. Entender

estas etapas é importante para perceber a complexidade e a meticulosidade necessárias para alcançar a aplicação final.

Assim sendo, seguem abaixo as etapas do desenvolvimento da aplicação:

- Comunicação SAP Build Apps (PipelineManager) – SAP
    1. Criação de conta SAP BTP – subscrição do plano Pay As You Go for SAP BTP;
    2. Ativação do booster SAP Build Apps;
    3. Iniciação do projeto no SAP Build Apps;
    4. Criação do Webservice REST em SAP;
    5. Testes e correção de problemas.
- Criação das tabelas da base de dados em SAP
- Criação das integrações REST na Aplicação
    1. Definição das entidades de dados na aplicação para permitir a obtenção de dados via pedidos HTTP;
    2. Criação dos métodos responsáveis pela lógica de obtenção de dados em SAP na classe que faz a gestão dos pedidos HTTP;
    3. Criação da Aplicação no SAP Build App.
- Testes
    1. Realização de testes de integração para verificar a interação entre os sistemas;
    2. Realização de testes nas diversas funcionalidades da aplicação;
    3. Identificação e correção de possíveis erros, falhas de segurança e problemas desempenho.

De seguida, apresentar-se-á o trabalho efetuado em cada uma das etapas.

5.3. Comunicação SAP Build Apps (PipelineManager) - SAP

Nesta secção serão explicados de forma detalhada todos os passos que foram necessários para a implementação do processo de comunicação entre a app PipelineManager na plataforma SAP Build Apps e o SAP, mais propriamente com o WebService REST.

Inicialmente, procedeu-se à criação do WebService REST em SAP,

Antes de se avançar com a explicação dos detalhes, creio que seja importante destacar que esta fase foi das mais complexas e desafiantes, uma vez que foi necessário estudar todas as tecnologias SAP que permitiam a comunicação entre diferentes sistemas, neste caso, Cloud/on-premise. para poder ter acesso ao serviço/plataforma SAP Build Apps, foi necessário subscrever o plano (SAP, Pay As You Go for SAP BTP) (Figura 15 - Pay as You Go for SAP BTP)

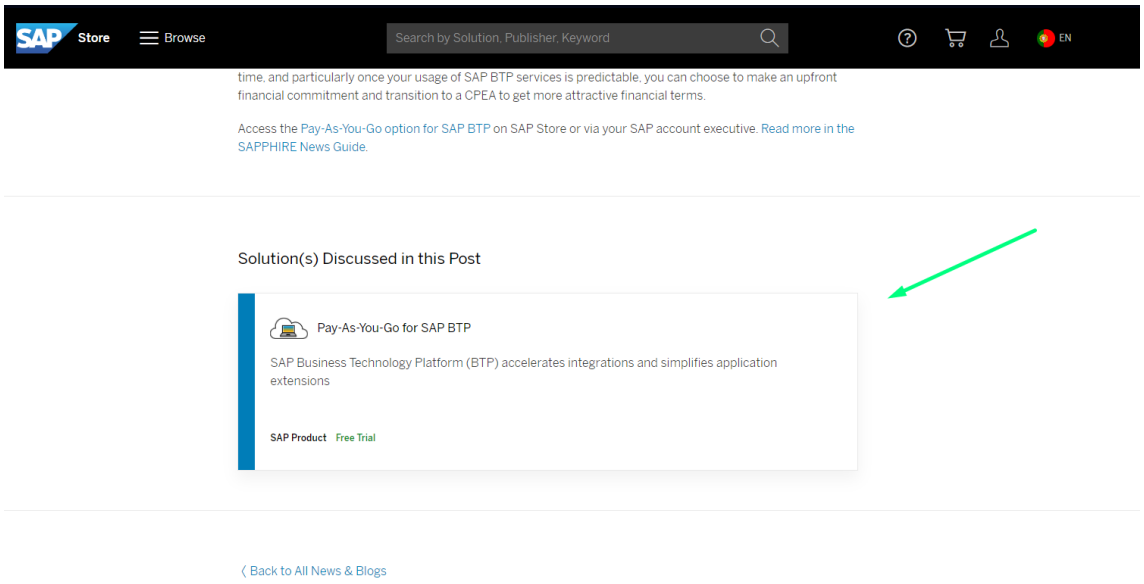


Figura 15 - Pay as You Go for SAP BTP

A partir desse ponto, foi feita a subscrição do serviço (SAP, SAP Build Apps Service).

Posto isto, os seguintes serviços foram disponibilizados numa subconta do SAP BTP.

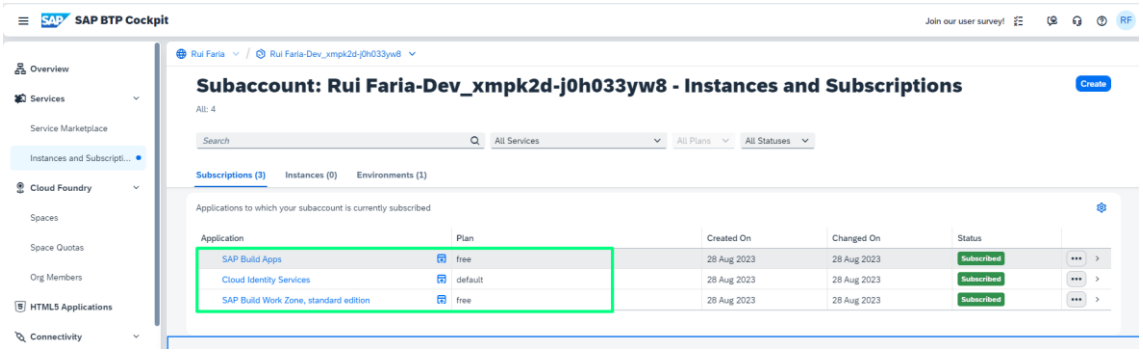


Figura 16 - SAP Build Apps

Posto isto, iniciou-se o serviço SAP Build Apps e criou-se o projeto com o nome PipelineManager.

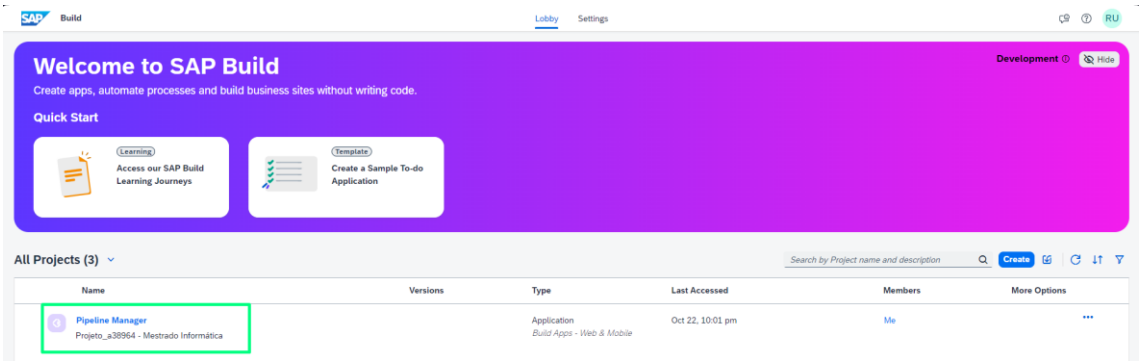
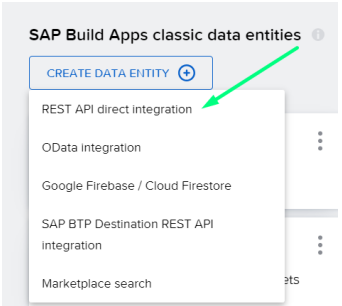


Figura 17 - SAP Build Apps – Criação do projeto

Posteriormente, procedeu-se à implementação da comunição entre os dois sistemas, PipelineManager e SAP. Para isto, na plataforma SAP Build Apps, mais propriamente no projeto, na aba “DATA” verificou-se que era necessário disponibilizar um url de uma API REST para ser consumido pela aplicação.



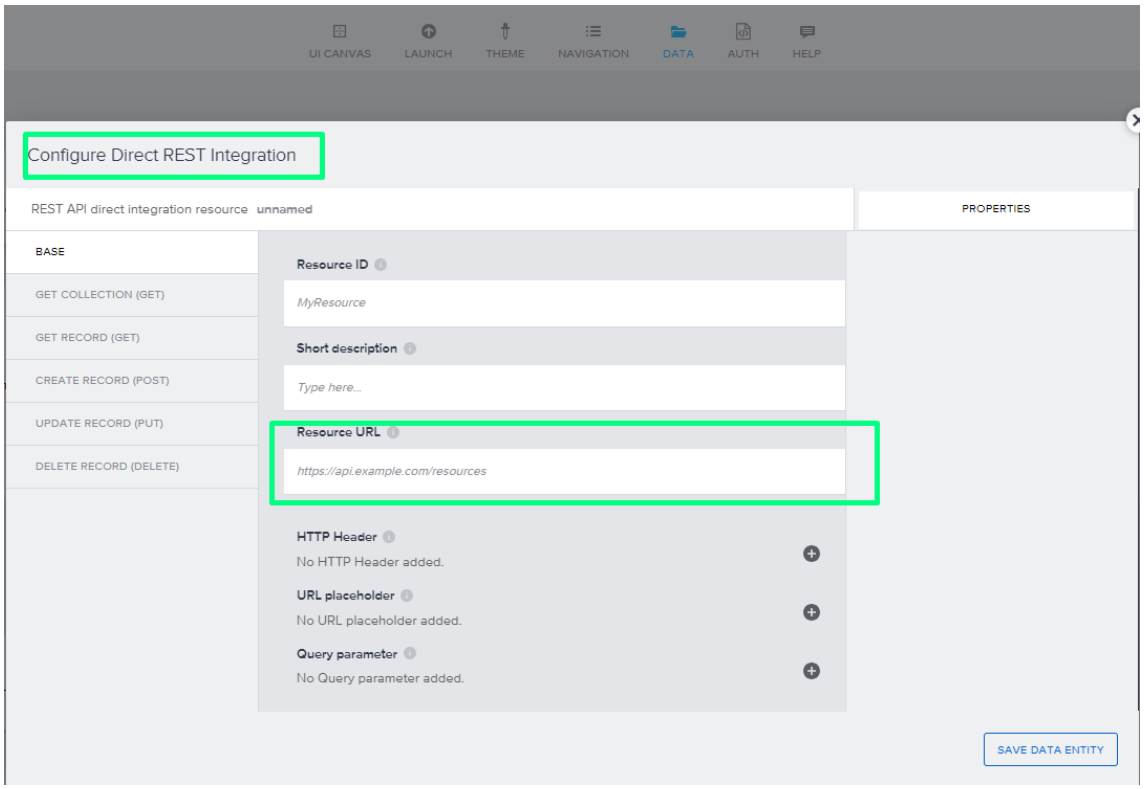


Figura 18 - Configuração do Consumo do WebService REST

Desta forma, começou-se pela criação do WebService REST, implementando o necessário para estabelecer uma ligação e construir o resto das funcionalidades à medida que a aplicação fosse sofrendo alterações.

5.3.1. Criação do WebService REST

Inicialmente, foi criado um serviço através da transação SICF (Bammidi) e, posteriormente a respetiva configuração e implementação de um código simples, apenas para verificar se do lado da aplicação é obtida uma resposta por parte do WebService (Significa que a comunicação está ok).

1. Criação do Serviço ZPIPE_SERVICE pela Transação SICF

   Para criar um serviço na transação SICF é necessário clicar com o botão direito do rato no elemento “BC” da árvore de elementos de serviços e, posteriormente, clicar em “New Subelement”.

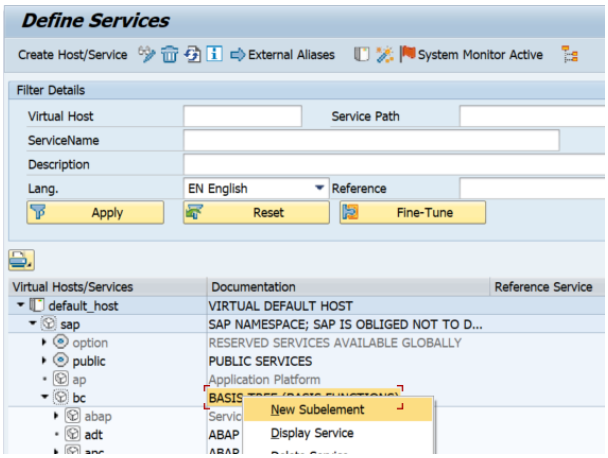


Figura 19 - Criação do Serviço na SICF

2. Configuração do Serviço Criado (ZPIPE_SERVICE)

Neste ponto, a única configuração necessária são os dados de logon de um user da máquina na tab logon data com o pisco SSL (ligação https, única suportada pelo SAP Build Apps, Figura 20 - Logon Data) e a classe que irá servir de *Handler* (Figura 21 – Handler*)* para os respetivos métodos GET, POST, PUT e DELETE das funcionalidades da App.

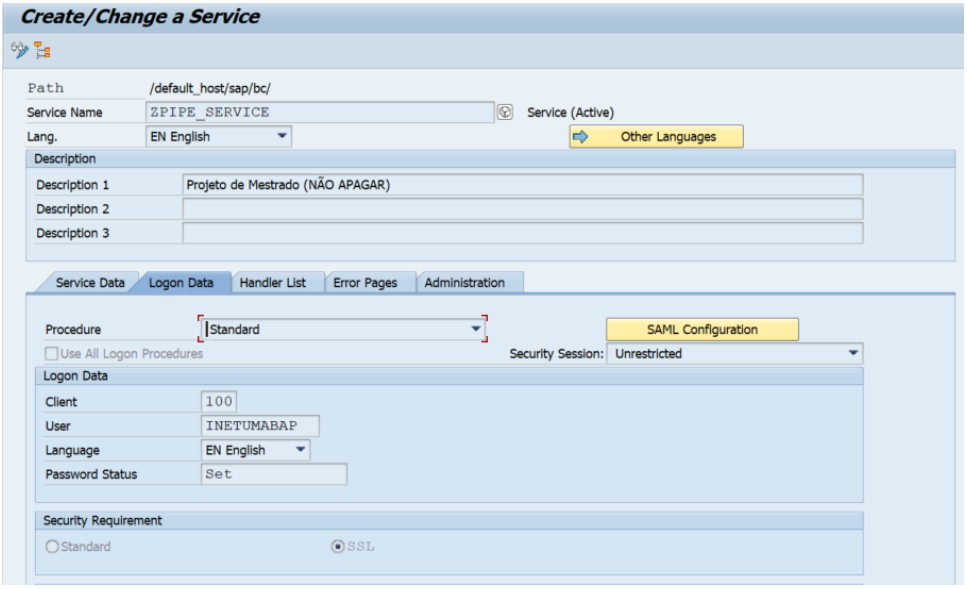


Figura 20 - Logon Data

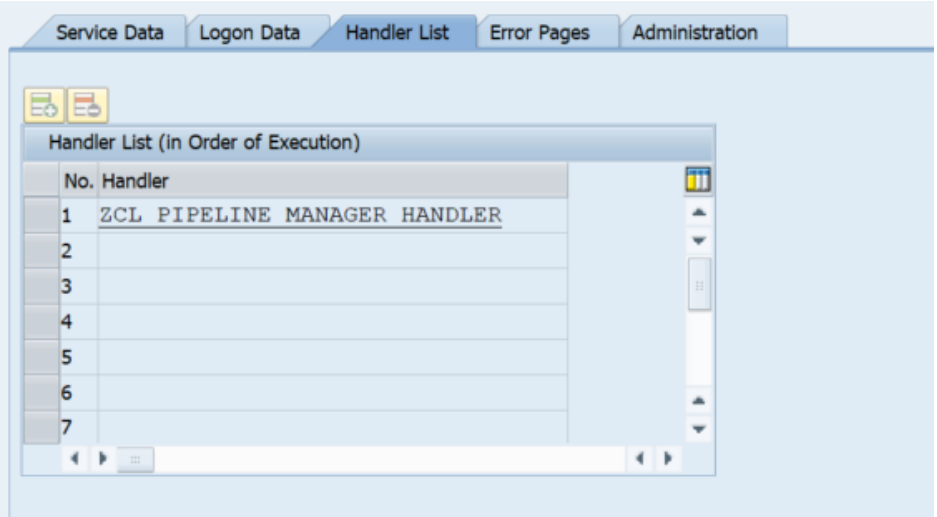


Figura 21 – Handler

A classe ZCL_PIPELINE_MANAGER_HANDLER foi criada na transação SE24 (SAP, Introduction to the Class Builder), inicialmente com a implementação da interface standard IF_HTTP_EXTENSION, que recebe o método HANDLE_REQUEST.

Neste momento, foi possível criar um simples código, como enviar uma string como resposta para a aplicação.

De seguida, testou-se a ligação do lado da aplicação e ocorreu um erro de autorizações. Com isto e após alguma pesquisa, verificou-se que tinha de arranjar forma de "expôr" o webservice na rede pública, uma vez que este fica retido na rede interna da máquina. Desta forma, surgiu o uso das ferramentas SAP Cloud Connector (gratuito) e SAP Cloud Platform integration (*Trial*).

Efetuou-se todas as configurações necessárias no SAP Cloud Connector, que envolve simplesmente a criação guiada de um servidor público que consome o privado (ZPIPE_SERVICE) por ligação interna com a máquina. No SAP Cloud Platform integration efetuou-se a criação de iFlow de forma a consumir o servidor público criado no SAP Cloud Connector. Contudo, obteve-se o mesmo erro de autorizações ainda que com uma mensagem diferente.

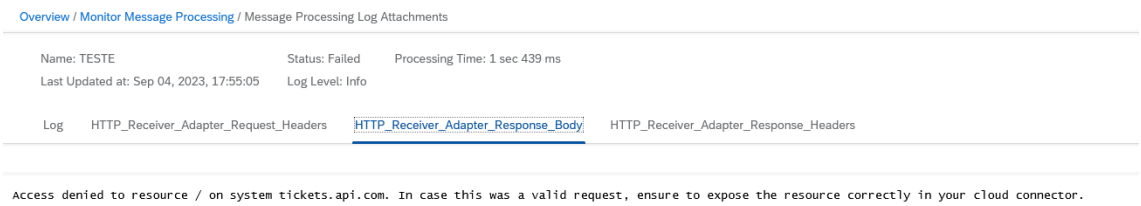


Figura 22 - Erro de conexão entre WebService e Aplicação

Desta forma, foi necessário fazer uma pesquisa profunda pelo erro ou até mesmo verificar a configuração completa de webservice no que toca à area de administração de sistemas.

Posto isto, foi possível verificar que era necessário instalar os certificados da plataforma SAP Build Apps na máquina através da transação STRUST (SAP, Install SSL Certificates) (Figura 23 - STRUST- certificado SAP Build Apps) e, além disso, realizar uma série de parametrizações por parte do administrador de sistemas. Feito isto, foi possível obter resposta com o WebService criado anteriormente, ZPIPE_SERVICE, e com a última configuração ficou desnecessário o uso das ferramentas SAP Cloud Connector e SAP Cloud Platform.

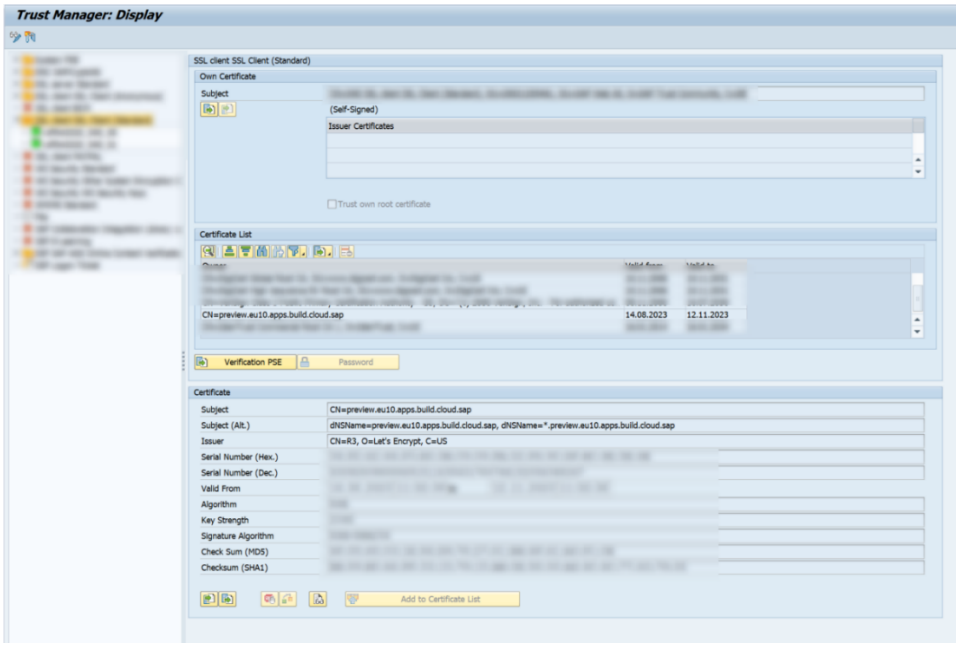


Figura 23 - STRUST- certificado SAP Build Apps

Contudo, ao testar a aplicação no browser, verificou-se que a ligação era insegura (Figura 24 - Ligação Insegura) e isto deve-se ao facto de a máquina onde o webservice tem origem, ter um certificado próprio, ou seja, *self-signed*, não gerado por uma Entidade certificadora. Este problema afeta a obtenção de dados

quando a aplicação é testada no telemóvel, uma vez que não era possível obter qualquer resposta do WebService. No browser, bastou aceitar a ligação e conseguiu-se obter resposta.

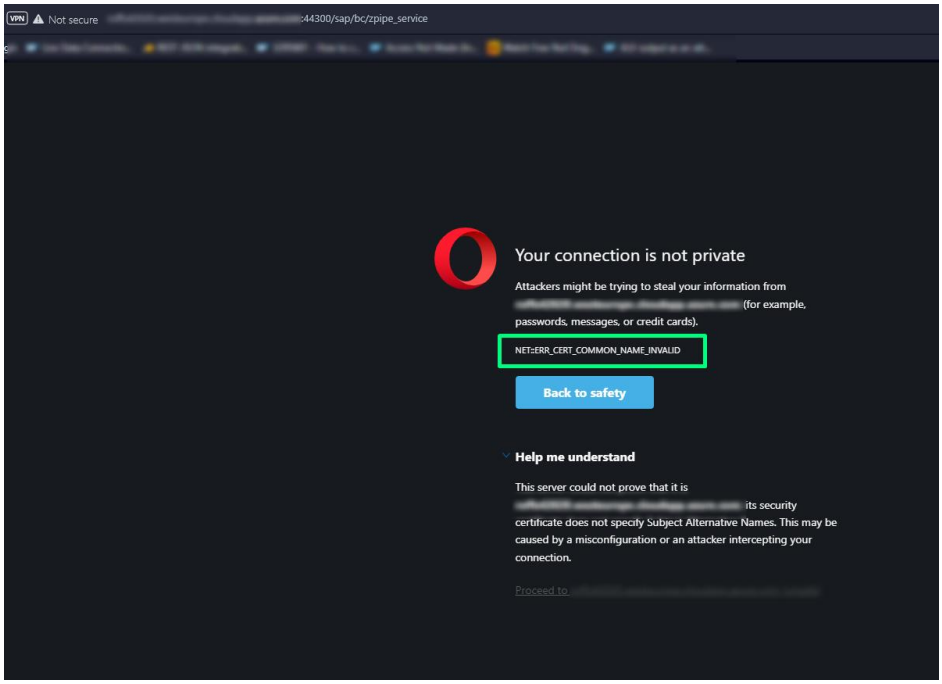


Figura 24 - Ligação Insegura

Com isto, decidiu-se não proceder com nenhuma resolução, uma vez que implicava custos e não inibia a evolução deste projeto.

5.4. Criação das tabelas e estruturas

Como referido no Capítulo 4, foram criadas diversas tabelas através da transação SE11, para dar suporte à base de dados da aplicação e estruturas para guardar temporariamente informação durante a comunicação entre o SAP e a aplicação.

5.4.1. Tabela ZTICKETS

A primeira tabela a ser criada e das principais tabelas, foi a ZTICKETs. Uma tabela que contém todos os pedidos de todos os utilizadores e respetivos detalhes dos pedidos. A tabela contém dois campos chave, o USERNAME e NRTICKET, de forma a ser possível na aplicação visualizar todos os tickets para um determinado utilizador e respetivos detalhes de cada pedido.

Tabela 1 - Tabela ZTICKETS

Campo	Elemento de dados	Tipo de Dados	Descrição
USERNAME	ZUSERNAME	CHAR30	Nome de utilizador
NRTICKET	ZNRTICKET	CHAR30	Nrº do pedido
CLIENTE	ZCLIENT	CHAR30	Nome do Cliente
NOTE	ZNOTA	CHAR1024	Nota do pedido
ZFILE	ZFILE	/PLMB/DIR_FILE_CONTENT	Ficheiro anexado
PRIORIDADE	ZPRIORITY	CHAR50	Prioridade do Pedido
ICON_COLOR	ZICONCOLOR	CHAR30	Cor do Icon Status
STATUS	ZPDS	CHAR50	Estado do Pedido
ICON_STATUS	ZICONSTATUS	CHAR50	Icon do Estado
FILENAME	ZFILENAME	CHAR50	Nome do ficheiro anexado
TITULO		CHAR30	Titulo do pedido
DATA_CRIACAO	DATUM	DATS8	Data de criação
HORA_CRIACAO	UZEIT	TIMS6	Hora de criação
NR_CARACT_DESC		CHAR30	Nr de caracteres da descrição

5.4.2. Tabela ZPIPELINE_USERS

Criou-se também uma tabela de utilizadores ZPIPELINE_USERS que contém todas informações essenciais dos utitlizadores, como dados de acesso à aplicação, dados de contacto e de controlo de autoriazações na aplicação. Relativamente aos dados de acesso, USERNAME e PASSWORD, o campo password é devidamente encriptado. O campo chave desta tabela é o USENAME.

Tabela 2 - Tabela ZPIPELINE_USERS

Campo	Elemento de dados	Tipo de dados	Descrição
USERNAME	ZUSERNAME	CHAR30	Nome de utilizador
PASSWORD	DBCON_PWD	CHAR255	Password do utilizador
EMAIL	ZCHAR90	CHAR90	Email do utilizador
ROLE	ZROLE	CHAR30	Grupo de autorização

5.4.3. Tabela ZUSERS_STATS e Estrutura ZSTATISTICS

Como foi apresentado anteriormente nas mockups existem detalhes estastísticos a serem apresentados por casa utilizador e no geral progresso dos pedidos na aplicação.

Relativamente aos detalhes por cada utilizador, foi necessário criar a tabela ZUSERS_STATS, que contêm dados como número de pedidos por cada utilizador e respetiva disponibilidade do mesmo que é alterada tendo em conta certas condições que serão explicadas mais à frente.

O campo chave nesta tabela é apenas o USERNAME.

Tabela 3 - Tabela ZUSERS_STATS

Campo	Elemento de dados	Tipo de dados	Descrição
USERNAME	ZUSERNAME	CHAR30	Nome de Utilizador
NR_TICKETS	INT4	INT4(10)	Nrº de pedidos
DISPONIBILIDADE	CHAR1	CHAR1	Disponibilidade

No que toca aos detalhes estatísticos gerais, foi apenas necessário criar uma estrutura designada por ZSTATISTICS, uma vez que será apenas uma linha de valores e não um conjunto de linhas de valores como acontece numa tabela.

Tabela 4 - Estrutura ZSTATISTICS

Campo	Elemento de ados	Tipo de dados	Descrição
NRTICKETS	INT4	INT4(10)	Nrº de pedidos
PROGRESSO	NUMC4	NUMC4	Progresso dos pedidos no geral
NRTICKETS_10	INT4	INT4(10)	Nrº de pedidos em status 10
NRTICKETS_20	INT4	INT4(10)	Nrº de pedidos em status 20
NRTICKETS_30	INT4	INT4(10)	Nrº de pedidos em status 30
NRTICKETS_40	INT4	INT4(10)	Nrº de pedidos em status 40

5.4.4. Tabela ZNOTES

Como foi apresentado anteriormente, num pedido é possível ter várias notas de diferentes utilizadores. Desta forma e como não é possível criar “deep tables” na base de dados em SAP, foi necessário criar uma tabela complementar à ZTICKETS, apenas com a informação relativa às notas por utilizador e por pedido.

Além disso, como cada utilizador e cada pedido pode ter várias notas, nenhum dos dois poderia ser campo chave. Assim, foi necessário ter um único campo chave, neste caso o campo NOTE_ID. No entanto, as ligação à tabela ZTICKETS é feita pelo campo NRTICKET, sendo que é em cada pedido que residem as notas dos utilizadores e não o contário.

Tabela 5 - Tabela ZNOTES

Campo	Elemento de dados	Tipo de dados	Descrição
NOTE_ID	INT4	INT4(10)	Id da nota
USERNAME	ZUSERNAME	CHAR30	Nome de utilizador

NRTICKET	ZNRTICKET	CHAR30	Nrº do pedido
ZDATE	DATS	DATS8	Data de criação
ZTIME	UZEIT	UZEIT6	Hora de criação
NOTE	ZNOTA	CHAR1024	Nota

5.4.5. Tabela ZFILES

Da mesma forma que a tabela ZNOTES, foi igualmente necessário criar a tabela ZFILES. Cada pedido pode conter um ou mais ficheiros de diferentes utilizadores. E, por isso, também existe apenas o campo chave FILE_ID.

Tabela 6 - Tabela ZFILES

Campo	Elemento de dados	Tipo de dados	Descrição
FILE_ID	INT4	INT4(10)	Id do ficheiro
NRTICKET	ZNRTICKET	CHAR30	Nrº do pedido
USERNAME	ZUSERNAME	CHAR30	Nome do utilizador
ZFILE	/PLMB/DIR_FILE_CONTENT	CHAR1024	Conteúdo do ficheiro
ZSIZE	INT4	INT4(10)	Tamanho do ficheiro
NAME	ZFILENAME	CHAR50	Nome do ficheiro
ZDATE	DATS	DATS8	Data de criação
ZTIME	UZEIT	UZEIT6	Hora de criação

5.4.6. Estrutura ZSTICKETS

Como foi dito anteriormente, não é possível criar “Deep tables” em SAP, ou seja, tabelas dentro de tabelas, mas sim “Deep Structures”, ou seja, tabelas dentro de uma estrutura.

Assim, o objetivo desta estrutura é guardar temporariamente informação de um pedido nos momentos em que existe uma atualização no pedido.

Tabela 7 - Estrutura ZSTICKETS

Campo	Elemento de dados	Tipo de dados	Descrição
USERNAME	ZUSERNAME	CHAR30	Nome de utilizador
NRTICKET	ZNRTICKET	CHAR30	Número do pedido
CLIENTE	ZCLIENT	CHAR30	Cliente
NOTE	ZTNOTE		Notas do pedido
ZFILE	ZTFILE		Ficheiros anexados no pedido
PRIORIDADE	ZPRIORITY	CHAR50	Prioridade do pedido
ICON_COLOR	ZICONCOLOR	CHAR30	Cor do icon indicativo do estado
STATUS	ZPDS	CHAR50	Estado do pedido

ICON_STATUS	ZICONSTATUS	CHAR50	Icon do estado do pedido
TITULO		CHAR30	Titulo do pedido
DATA_CRIACAO	DATUM	DATS8	Data de criação
HORA_CRIACAO	UZEIT	TIMS6	Hora de criação
NR_CARACT_DESC		CHAR30	Numero de caracteres da nota

Uma vez que um pedido pode ter mais do que uma nota e ficheiro, foi necessário criar uma estrutura em que os campos NOTE e ZFILE são dos tipos de tabelas ZTNOTE e ZTFILE, respetivamente.

Estes dois tipos de Tabelas foram criados com os tipos de estruturas (linhas), ZSNOTE e ZSFILE, respetivamente.

5.4.6.1. Estrutura ZSNOTE

Tabela 8 - Estrutura ZSNOTE

Campo	Elemento de dados	Tipo de dados	Descrição
USERNAME	ZUSERNAME	CHAR30	Nome de utilizador
NRTICKET	ZNRTICKET	CHAR30	Numero do pedido
ZDATE	DATS	DATS8	Data de criação
ZTIME	UZEIT	TIMS6	Hora de criação
NOTE	ZNOTA	CHAR1024	Nota do pedido

5.4.6.2. Estrutura ZSFILE

Tabela 9 - Estrutura ZSFILE

Campo	Elemento de dados	Tipo de dados	Descrição
NAME	ZFILENAME	CHAR50	Nome do ficheiro
ZFILE	/PLMB/DIR_FILE_CONTENT	CHAR1024	Conteúdo do ficheiro
ZSIZE	INT4	INT4(10)	Tamanho do ficheiro
USERNAME	ZUSERNAME	CHAR30	Nome do utilizador
ZDATE	DATS	DATS8	Data de criação
ZTIME	UZEIT	TIMS6	Hora de criação

5.5. Criação das integrações RESTs na Aplicação

Com o WebService REST criado e em estado funcional, passou-se à criação das integrações REST no projeto da aplicação. São funcionaliades que vão permitir a construção da lógica que por sua vez vai permitir os utilizadores fazerem uso das diferentes funcionalidades da aplicação.

Para criar a integração é necessário preencher a aba base com o nome, descrição e url da API, neste caso, do WebService REST (Figura 18 - Configuração do Consumo do WebService REST).

A transferência de dados é feita pelos diversos métodos disponibilizados pela funcionalidade (GET, POST, PUT, DELETE) que serão associados a métodos da classe *handler* em SAP, através da criação de um "query parameter".

Para a implementação dos métodos em SAP, é necessário fazer uma pré-implementação no método IF_HTTP_EXTENSION~HANDLE_REQUEST, para podermos obter o pedido HTTP da aplicação com os respetivos dados. É neste método que é definida toda a lógica necessária para comunicar com base de dados SAP ou o que quer que seja necessário enviar para a aplicação como resposta ao pedido HTTP.

Assim sendo, declarou-se duas tabelas internas do tipo de tabela TIHTTPNVP para guardar informação do header do pedido HTTP e dos "query parameters", respetivamente. Posto isto, fez-se uso do método get_header_fields da interface IF_HTTP_ENTITY para obter a informação do header do pedido, nomeadamente, o tipo de pedido (GET, POST, PUT, DELETE, OPTIONS). Também, fez-se uso do método get_form_fields da mesma interface para obter informação dos "query parameter", ou seja, ação dos utilizadores. Com isto, fez-se a leitura dessas duas tabelas para obter as linhas com o tipo de pedido ("~request_method" – linha 16) e com a ação do utilizador ("action" – linha 19), respetivamente.

Listagem 1 - Código Base

```
1   DATA: it_header_fields TYPE  tihttpnvp.
2   DATA: it_form_fields TYPE  tihttpnvp.

3   DATA: wa_method LIKE LINE OF it_header_fields.
4   DATA: wa_action LIKE LINE OF it_header_fields.
5
6   CALL METHOD server->request->if_http_entity~get_header_fields
7    CHANGING
8      fields = it_header_fields.
9
```

```
10  CALL METHOD server->request->if_http_entity~get_form_fields
11   CHANGING
12     fields = it_form_fields.
13
14
15  * Obter o tipo de pedido
16  READ TABLE it_header_fields INTO wa_method  WITH KEY name = '~request_method'.
17
18  * Obter ação do utilizador.
19  READ TABLE it_form_fields INTO wa_action  WITH KEY name = 'action'.
```

Posto isto, segue-se a implementação da lógica necessária para todas as funcionalidades da aplicação. Assim, para organizar o código, diferenciar a lógica tendo em conta o pedido HTTP feito e melhorar o "handling de erro", optou-se por encapsular as diferentes lógicas em diferentes métodos.

Class Builder: Display Class ZCL_PIPELINE_MANAGER_HANDLER

Local Definitions/Implementations | Local Test Classes | Source Code-Based | Class docu

Class/Interface: ZCL_PIPELINE_MANAGER_HANDLER — Implemented / Active

Properties | Interfaces | Friends | Attributes | Methods | Events | Types | Aliases

Parameters | Exceptions | Sourcecode | Filter

Method	Level	Visibility	Me...	Description
IF_HTTP_EXTENSION~HANDLE_REQUEST	Instance Method	Public		Called for request handling for each incoming HTTP request
GET_TICKETS_BY_USER	Instance Method	Private		Obter todos os tickets da tabela ZTICKETS por user
CREATE_TICKET	Instance Method	Private		Criar um novo ticket na tabela ZTICKETS
DELETE_TICKET	Instance Method	Private		Eliminar um ticket
GET_TICKET	Instance Method	Private		Obter um ticket
UPDATE_TICKET	Instance Method	Private		Atualizar um ticket
CHECK_USER	Instance Method	Private		Autenticação na aplicação
GET_ALL_TICKETS	Instance Method	Private		Obter todos os tickets e os respetivos users
GET_STATISTICS	Instance Method	Private		Obter estatisticas das pedidos
GET_USER_STATS	Instance Method	Private		Obter estatisticas dos users
GET_USERS	Instance Method	Private		Obter info dos users
ADD_USER	Instance Method	Private		Adicionar User
DELETE_USER	Instance Method	Private		Delete User
SEND_EMAIL	Instance Method	Private		Enviar email de pds

Figura 25 - Métodos da classe ZCL_PIPELINE_MANAGER_HANDLER

Desta forma, criou-se uma lógica para direcionar o pedido para um determinado método da classe *handler*, tendo em conta o tipo de pedido e ação.

Existem ainda outras pré-implementações que são importantes de referir, uma vez que são necessárias para que realmente seja efetuada com a sucesso a troca de informação entre os dois sistemas (SAP e aplicação).

Os métodos que exigem a introdução de dados por parte do utilizador requerem igualmente uma pré-implementação em que haja a receção dos dados e posterior conversão de JSON para um tipo de dados adequado em SAP, seja esse uma tabela, estrutura ou variável.

Listagem 2 - Código para obtenção de dados introduzidos pelo utilizador

```
CALL METHOD server->request->if_http_entity~get_cdata
  RECEIVING
    data = body.

CALL METHOD /ui2/cl_json=>deserialize
  EXPORTING
    json = body
  CHANGING
    data = CONTENT
```

Nesta pré-implementação é feita uma receção dos dados introduzidos pelo utilizador na aplicação através do método GET_CDATA da interface IF_HTTP_ENTITY e posterior conversão de JSON para um tipo de dados em SAP através do método DESERIALIZE da classe /UI2/CL_JSON.

Assim como a pré-implementação anterior, também existe necessidade de uma para a resposta.

No final de cada método existe sempre uma resposta para a aplicação, seja com informação necessária para o utilizador ou uma notificação de sucesso no que toca ao pedido HTTP (Conexão estabelecida). Assim, foi igualmente necessário fazer uma pré-implementação para cada método.

Listagem 3 - Código para envio de resposta

```
lv_result = /ui2/cl_json=>serialize( data        = RESPONSE_CONTENT
                                     compress    = abap_false
                                pretty_name = /ui2/cl_json=>pretty_mode-camel_case ).

server->response->set_header_field( name  = 'Content-Type'  "#EC NOTEXT
                                    value = 'application/json' ).

IF ( wa_origin-value = 'https://xmpk2d-j0h033yw8.design-
   time.eu10.apps.build.cloud.sap' OR
   wa_origin-value = 'https://xmpk2d-
   j0h033yw8.preview.eu10.apps.build.cloud.sap' ).

   ********   enable CORS Access
   server->response->set_header_field( name  = 'Access-Control-Allow-Origin'
                                      value = wa_origin-value ).
ENDIF.

 server->response->set_cdata( data = lv_result ).
 server->response->set_status( code = 200 reason = 'OK' ).
```

Nesta pré-implementação é feita uma conversão da informação obtida na lógica do método, seja de uma estrutura, tabela ou variável para JSON e posteriormente será incluído na resposta a ser enviada para a aplicação. Além disso, fez-se a ativação do cors para permitir a transferência de dados entre os diferentes protocolos (HTTP e HTTPS), neste caso do WebService Rest e o SAP Build Apps, (MDN).

5.5.1. Tickets

A integração REST designada por Tickets, como o nome indica, é a funcionalidade que permite aos utilizadores a gestão dos seus pedidos. Isto é, obter, editar, criar e eliminar pedidos. Para isso, foram criados os respetivos métodos na classe que faz o *handle* (Figura 21 – Handler) dos pedidos http (get, post, put, delete).

Assim sendo, procedeu-se com a configuração necessária, ativaram-se todos os métodos, "get collection, get record, create record, update record e delete record".

Figura 26 - Integração REST Tickets

Para cada método criou-se um "query parameter" para definir o tipo de ação desejada pelo utilizador. Dependendo do método, criou-se um "url placeholder" que serve como um parameter para filtrar dados.

5.5.1.1. Método "Get Collection" (Get_Tickets_By_User) - GET

O objetivo deste método é obter todos os pedidos do utilizador logado através do método Get_Tickets_By_User em SAP.

Assim, criou-se um "url placeholder" para passar o nome do utilizador logado e em SAP fez-se a leitura do campo "~path_info", criado dinamicamente, da tabela interna header.

De seguida, declarou-se uma tabela interna do tipo da tabela ZTICKETS e fez-se uma seleção dos dados da tabela ZTICKETS para a tabela interna onde o username é igual ao valor do "~path_info".

A tabela interna é atribuída ao "RESPONSE_CONTENT" para ser enviado para a aplicação.

5.5.1.2. Método "Get Record" (Get_Ticket) - GET

Neste método obtém-se toda a informação relativa a um pedido que o utilizador pesquisa através do método Get_Ticket em SAP. A informação do pedido inclui todos os campos da estrutura ZSTICKETS.

Criou-se igualmente um "url placeholder" para passar o numero do pedido e em SAP fez-se a leitura do campo "~path_info".

De seguida, declarou-se uma estrutura do tipo da ZSTICKETS e fez-se uma seleção à tabela ZTICKETS com o número do pedido no campo "~path_info" para uma estrutura.

Posto isto, declarou-se duas tabelas internas do tipo das tabelas ZNOTES e ZFILES, respetivamente, e fez-se a seleções às tabelas com o número do pedido.

Os campos NOTE e ZFILE da estrutura ZSTICKETS são ambos tabelas dos tipos de tabelas ZTNOTE e ZTFILE e por isso para passar a informação das tabelas internas para a estrutura final, foram necessários fazer dois ciclos separados das tabelas internas, onde por cada iteração são passadas as informações de linha para as estruturas temporárias dos tipos ZSNOTE e ZSFILE e de seguida, é feita a adição das linhas aos campos NOTE e ZFILE da estrutura principal.

Finalmente, é atribuída a estrutura declarada, LS_STICKETS ao "RESPONSE_CONTENT" para ser enviado para a aplicação (Listagem 3 - Código para envio de resposta).

5.5.1.3. Método "Create Record" (Create_Ticket) - POST

O método Create_Ticket, como o nome indica, permite a criação de um pedido com informação dada pelo utilizador.

Inicialmente, declara-se uma estrutura do tipo da ZSTICKETS, designada por LS_STICKETS e é feita a obtenção dos dados introduzidos na aplicação pelo utilizador (Listagem 2 - Código para obtenção de dados introduzidos pelo utilizador).

De seguida, declara-se outra estrutura do tipo de linha da tabela ZTICKETS, designada por LS_TICKETS e atribui-se os dados da LS_STICKETS para a LS_TICKETS, tendo em conta os campos correspondentes.

Posto isto, faz-se um INSERT na tabela ZTICKETS com os valores que se encontram na estrutura LS_TICKETS.

No caso da criação da entrada na tabela anterior for bem sucedida, prossegue-se com a criação das entradas nas tabelas ZNOTES, ZFILES e ZUSER_STATS.

Assim sendo, para as tabela ZNOTES e ZFILES faz-se, primeiramente, um SELECT COUNT(*) para verificar qual o número do ID, tanto da nota como do ficheiro, deverá ser criado. Ou seja, se, por exemplo, o número de entradas for 1, então o ID a criar será 2.

De seguida, em dois ciclos separados, um das notas e outro dos ficheiros anexados pelo utilizador, são passadas as informações para estruturas declaradas e, posteriormente, são feitos os respetivos INSERTs nas tabelas tendo em conta a informação das estruturas. Isto, repetidamente, tendo em conta o número de entradas nas tabelas. No caso da notas é apenas 1 entrada, uma vez que trata-se da criação do pedido. Relativamente aos ficheiros, pode ser mais do que 1 ficheiro.

Posto isto, para a tabela ZUSERS_STATS, inicialmente, é feito um SELECT SINGLE * tendo em conta o utilizador para uma estrutura temporária. De seguida, verifica-se se existem dados para o utilizador em questão, se existirem, então será feito apenas um UPDATE na tabela através da estrutura, onde o campo NR_TICKETS é incrementado 1 unidade. Caso contrário, o campo NR_TICKETS fica com valor 1, o de DISPONIBILIDADE a vazio (assim indica que o utilizador está indisponível) e, porteriormente, faz-se um INSERT na tabela.

5.5.1.4. Método “Update Record” (Update_Ticket) - PUT

O objetivo neste método é atualizar alguns detalhes do pedido, nomeadamente, o utilizador, estado, prioridade, adiocinar uma nota e ficheiros.

Como no método Create_Ticket, declara-se uma estrutura do tipo da ZSTICKETS, designada por LS_STICKETS e faz-se uso do Código na Listagem 2 - Código para obtenção de dados introduzidos pelo utilizador.

De seguida, declara-se outra estrutura do tipo de linha da tabela ZTICKETS, designada por LS_TICKETS e atribui-se os dados da LS_STICKETS para a LS_TICKETS, tendo em conta os campos correspondentes.

Caso exista uma atualização do campo utilizador, é feita uma eliminição do pedido com o antigo utilizador e uma inserção do pedido com o novo na tabela ZTICKETS.

Posto isto, para as tabela ZNOTES e ZFILES faz-se, primeiramente, um SELECT COUNT(*) para verificar qual o número do ID, tanto da nota como do ficheiro, deverá ser criado. Ou seja, se, por exemplo, o número de entradas for 1, então o ID a criar será 2.

De seguida, em dois ciclos separados, um das notas e outro dos ficheiros anexados pelo utilizador, são passadas as informações para estruturas declaradas e, posteriormente, são feitos os respetivos INSERTs nas tabelas tendo em conta a informação das estruturas. Isto, repetidamente, tendo em conta o número de entradas nas tabelas. No caso da notas é apenas 1 entrada, uma vez que trata-se da criação do pedido. Relativamente aos ficheiros, pode ser mais do que 1 ficheiro.

Posteriormente, verifica-se se existem dados na tabela ZUSERS_STATS, se sim, então faz-se um SELECT COUNT(*) da tabela ZTICKETS para o respetivo utilizador e com os estados “Enviado para Testes” e “A aguardar resposta”. Caso existam entradas e o número de entradas for igual ao numero de tickets do utilizador então significa que o este está disponível, logo atualiza-se o campo DISPONIBILIDADE com “X” e faz-se um UPDATE à tabela ZUSERS_STATS. Caso contrário, o campo DISPONIBILIDADE fica a vazio e faz-se, igualmente, o UPDATE.

5.5.1.5. Método “Delete Record” (Delete_Ticket)

Além de outras funcionalidades, o utlizador pode eliminar um pedido da sua lista. Este método permite essa mesma funcionalidade.

Existe uma particularidade neste método que foi até complexo de idealizar. Isto é, quando um utilizador deseja eliminar um ficheiro anexado num determinado pedido, aqui é feita a lógica para esse pedido, uma vez que na Plataforma SAP Build Apps teve-se de usar o método delete, obrigatoriemente, e por isso é identificado como um DELETE no pedido HTTP. Assim, implementou-se uma lógica que permite-se a eliminação de ficheiros sem entrar em conflito com o resto.

Desta forma, assim como nos dois métodos anteriores, é preciso obter a informação do pedido que o utilizador deseja eliminar com o Código identificado em Listagem 2 - Código para obtenção de dados introduzidos pelo utilizador.

Com a informação anterior, para verificar se existia apenas a eliminação de um ficheiro, na aplicação apenas envia-se a informação relativa ao ficheiro e o número do pedido. Assim, verifica-se se o campo USERNAME está a vazio e o campo tabela ZFILE está preenchido.

Com isto, se se verificar efetua-se um ciclo ao campo tabela ZFILE e atribui-se a informação a um estrutura temporária. Posto isto, faz-se uma seleção dos dados à tabela ZFILES com o número do pedido e o nome do ficheiro que está contido na estrutura.

De seguida, procede-se com o DELETE à tabela ZFILES com a informação do ficheiro e interrompe-se a execução do restante código com a instrução RETURN.

Caso não seja uma ação de eliminação de um ficheiro, então a lógica anterior é ignorada e executa-se o DELETE na tabela ZTICKETS com a informação do pedido.

De seguida, é feita a eliminação do restante, ou seja, das notas e ficheiros contidos no pedido.

Posteriormente, para a tabela ZUSERS_STATS faz-se a lógica implementada no método anterior.

5.5.2. Users

A integração REST designada por Users, é a funcionalidade que permite aos utilizadores "Admins" a gestão dos utilizadores. Isto é, obter, criar e eliminar utilizadores. Para isso, foram criados os respetivos métodos na classe que faz o handle (Figura 19 – Handler) dos pedidos http (get, post, e delete).

Assim sendo, procedeu-se com a configuração necessária, ativaram-se os métodos, "get collection, create record, e delete record".

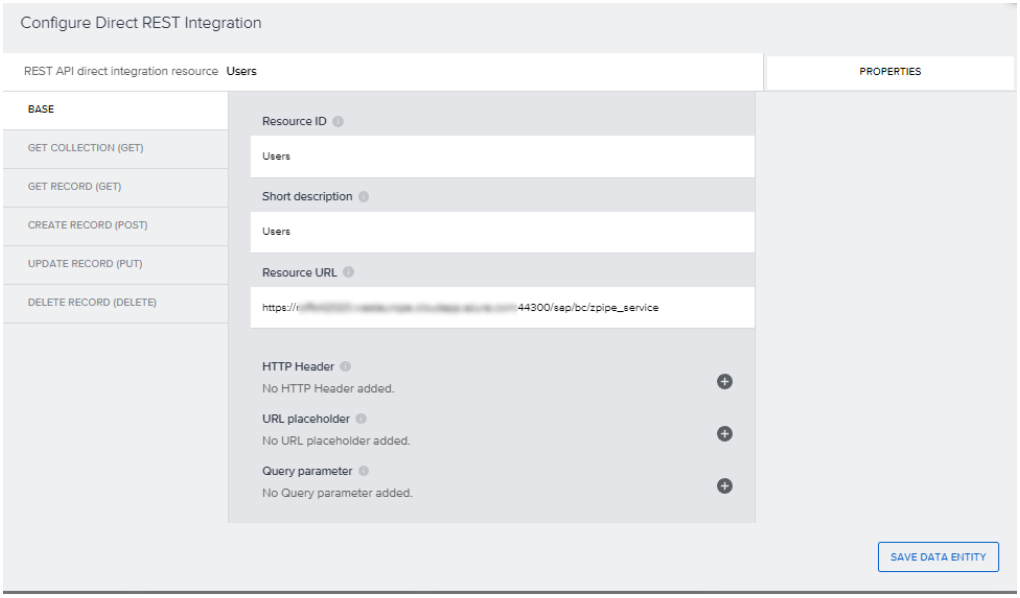


Figura 27 - Integração REST Users

Para cada método criou-se um "query parameter" para definir o tipo de ação desejada pelo utilizador.

5.5.2.1. Método "Get Collection" (Get_Users) – GET

Como o nome indica, este método permite obter todos os users registados na aplicação. Este método apenas é executado para utilizadores "Admin".

Aqui declarou-se uma tabela interna do tipo da tabela ZPIPELINE_USERS e fez-se uma seleção de todos os dados da tabela ZPIPELINE_USERS para a tabela interna .

A tabela interna é atribuída ao "RESPONSE_CONTENT" para ser enviado para a aplicação.

5.5.2.2. Método "Create Record" (Add_User) – POST

Para este método, o objetivo principal é integrar novos uitlizadores na aplicação, cuja informação dos acessos, autorizações e email são registados pelo utilizador "Admin".

Desta forma, utilizou-se a lógica implementada em Listagem 2 - Código para obtenção de dados introduzidos pelo utilizador, para obter a informação registada. De seguida, a password do utilizador passa por um processo de encriptação através do módulo de função DB_CRYPTO_PASSWORD (SE80, SAP DB_CRYPTO_PASSWORD Function Module)que retorna o password encriptada.

Posto isto, efetua-se o INSERT na tabela ZPIPELINE_USERS com a informação anterior.

5.5.2.3. Método "Delete Record" (Delete_User) – DELETE

Este método permite ao utilizador "Admin" a possibilidade de eliminar qualquer utilizador da aplicação.

Para isso, usa-se igualmente a lógica definida em Listagem 2 - Código para obtenção de dados introduzidos pelo utilizador, para obter a informação do utilizador a elminar e, posteriormente, faz-se o DELETE da tabela ZPIPELINE_USERS.

Além disso, efetua-se, igualmente, o DELETE da tabela ZUSERS_STATS e ZTICKETS, ou seja, eliminação das estatísticas associadas ao utilizador e todos os pedidos que estavam na sua lista de pedidos.

5.5.3. UsersTickets

A integração REST designada por UsersTickets, é a funcionalidade que permite aos utilizadores "Admins" verificar a atividade dos utilizadores. Isto é, obter todos os pedidos de todos os utilizadores. Para isso, foi criado o respetivo método na classe que faz o handle (Figura 19 – Handler) dos pedidos http (get).

Assim sendo, procedeu-se com a configuração necessária, ativou-se o método, "get collection".

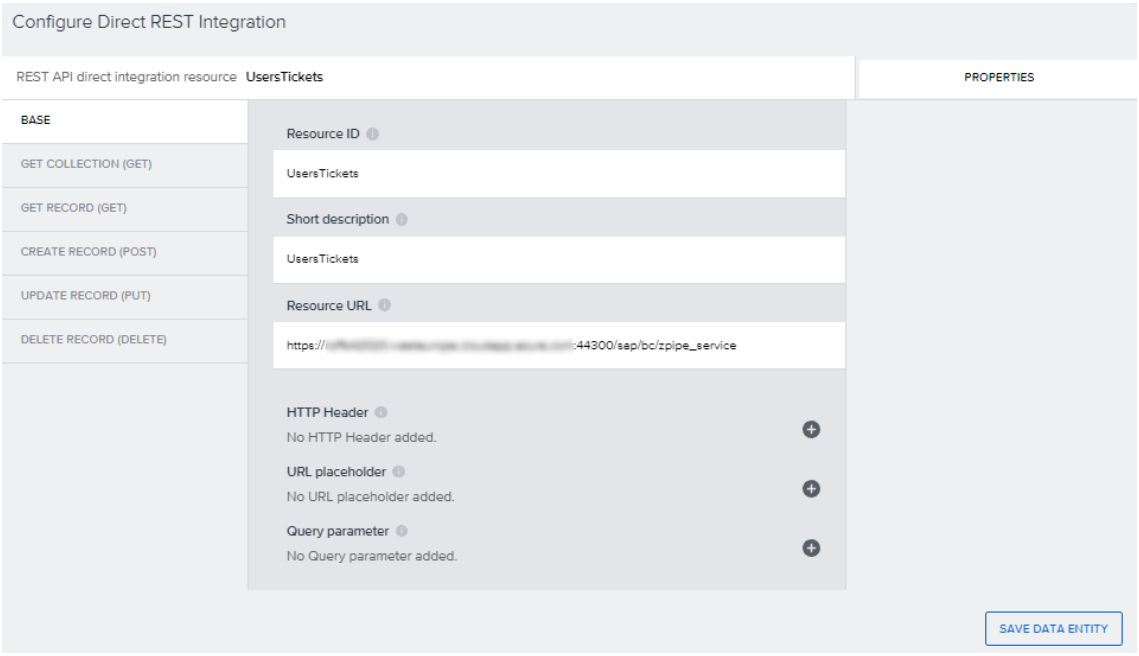


Figura 28 - Integração REST UsersTickets

Para cada método criou-se um "query parameter" para definir o tipo de ação desejada pelo utilizador.

5.5.3.1. Método "Get Collection" (Get_All_Tickets) – GET

Neste método, a informação enviada para a aplicação é da tabela ZTICKETS.

Simplesmente fez-se uma seleção de todos os dados da tabela ZTICKETS e atribuiu-se ao "RESPONSE_CONTENT".

5.5.4. Stats

A integração REST designada por Stats, é a funcionalidade que permite aos utilizadores "Admin" verificar as estatísticas da aplicação, nomeadamente, os pedidos de todos os utilizadores, o progresso geral, os pedidos por estados e a percentamge de pedidos que cada utilizador possui.

Assim sendo, procedeu-se com a configuração da integração e ativaram-se os métodos "Get Collection" e "Get Record".

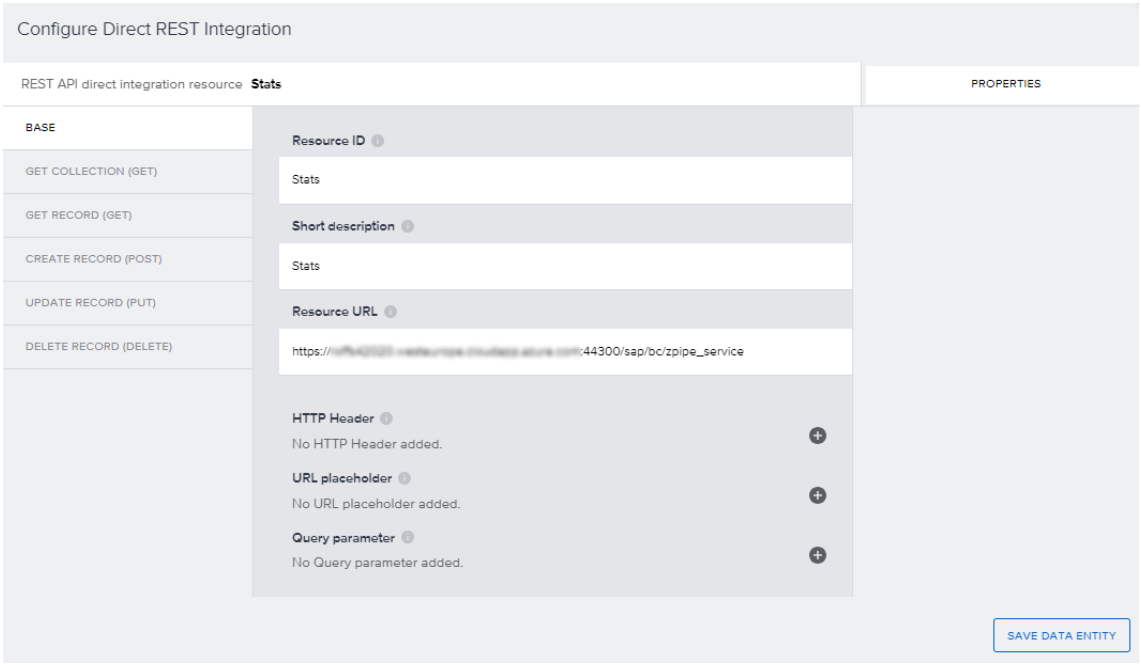


Figura 29 - Integração REST Stats

Para cada método criou-se um "query parameter" para definir o tipo de ação desejada pelo utilizador.

5.5.4.1. Método "Get Collection" (Get_User_Stats) – GET

Neste método, a informação enviada para a aplicação é relativa à tabela ZUSERS_STATS.

Fez-se uma seleção a todos os dados da tabela e atribuiu-se ao "RESPONSE_CONTENT".

5.5.4.2. Método "Get Record" (Get_Statistics) – GET

Este método tem como objetivo calcular uma série de estatísticas através da tabela ZTICKETS.

Inicialmente, declara-se uma estrutura do tipo ZSTATISTICS. Seleciona-se todos os pedidos da tabela ZTICKETS. De seguida, num ciclo incrementa-se 1 unidade a uma variável, designada total, verifica-se os pedidos com estado "Enviado para testes" e incrementa-se 1 unidade a uma variável, designada progresso, por cada pedido com esse estado. Posteriormente, por cada estado incrementa-se 1 unidade a variáveis correspondentes ao estado.

Depois do ciclo, atribui-se a variável total ao campo nrtickets da estrutura e calcula-se o progresso dos pedidos, ou seja, dividi-se a variável progresso pela total e multiplica-se por 100, atribuindo o resultado ao campo progresso da estrutura.

Posto isto, atribui-se a estrutura ao "RESPONSE_CONTENT".

5.6. Criação da Aplicação no SAP Build Apps

Após toda a preparação da lógica de backend, iniciou-se a construção da aplicação no SAP Build Apps.

Inicialmente, criaram-se variáveis de dados para as páginas que apresentam informação útil para os utilizadores, ou seja, a página Pipeline, TicketDetails e Users . Este tipo de variáveis tem um esquema definido pelas entidades de dados criadas, as integrações REST. Além disso, contêm uma lógica pré-definida para obter os dados do backend e preenchê-los na variável. Posto isto, criaram-se as variáveis de página para as páginas onde exigem o preenchimento de informação por parte do utilizador. Ao contrário das variáveis anteriores, pode ser definido qualquer esquema e não estão associadas a nenhuma entidade. Ambas as variáveis existem no contexto da página atual. São inicializadas quando a página é aberta e removidas do estado da aplicação quando a página é fechada. Devem ser utilizadas para componentes que existem no contexto da página atual, tais como dados de formulários, momento de carregamento da página atual, filtros seleccionados, etc.

De seguida, começou-se a definir as páginas com base nas mockups criadas em 4.4 Mockups, usando a particularidade da tecnologia de Low Code No Code, o drag and drop. Com isto, foi possível definir facilmente e rapidamente todos os componentes necessários para cada página.

Após a composição do design desejado para cada página, passou-se à fase de associação das variáveis de dados aos componentes de output e das variáveis de páginas aos componentes de input, juntamente com a criação da lógica necessária para cada página, de forma a projetar o funcionamento da aplicação, idealizado anteriomente.

5.6.1. Página Logo

A página do Logo é a página inicial de introdução à aplicação, sendo por isso visualizada num curto espaço de tempo. Nesta página, sendo a inicial, definiu-se o Logo da aplicação com o componente Image, juntamente com um spinner.

Ao nível de lógica, apenas associou-se o componente de animação, com um delay de 2 segundos. Assim que terminar o delay, utilizou-se o componente de navegação para abrir a página de Login.

5.6.2. Página Login

Na página de Login, efetuou-se uma série de alterações para validar a informação de login introduzida pelo utilizador.

Primeiramente, criou-se um novo método na classe *handler* em SAP, designado por Check_User. Neste método, utilizou-se a pré-implementação definida em Listagem 2 - Código para obtenção de dados introduzidos pelo utilizador, para obter a informação de login introduzida pelo utilizador na aplicação. Posto isto, faz-se uma seleção de dados à tabela ZPIPELINE_USERS com o nome de utilizador recebido. De seguida, encripta-se a password inserida pelo utilizador e verifica-se se coincide com a guardada na base dados. Se for igual, envia-se uma reposta de sucesso para a aplicação com o código 200, caso contrário é enviado o código 206 com a mensagem “Login Incorreto”.

Conseguinte, do lado da aplicação a variável de página foi criada do tipo object com um esquema de 2 campos, username e password, de forma a serem associados aos respetivos campos de input, username e password. Foi também adicionada um variável de dados associada à entidade Users, de forma a obter, posteriormente, informação sobre o role do User para definir autorizações de acesso às páginas. A nível de lógica, definiu-se um componente de animação para o container onde reside a janela de login para introduzir a própria página. No botão login, criou-se uma lógica para validar o Login do utilizador. Assim, usou-se o componente que emite um pedido HTTP (método POST) com a informação da variável de página, associado ao método Check_User criado em SAP na classe *handler*. Com a resposta do pedido, usa-se o componente da condição IF para verificar se o valor do código recebido é 200. Se não for esse o código, então é usado o componente Alert para notificar o utilizador que o login está incorreto. Caso contrário,

são despoletadas três ações: o componente de navegação para a página pipeline, a associação do username que o utilizador inseriu numa variável de aplicação (Esta variável fica disponivel para todas as páginas) e associação da role do username que fez login numa outra variável de aplicação, através da seleção do username da variável de dados definida anteriormente.

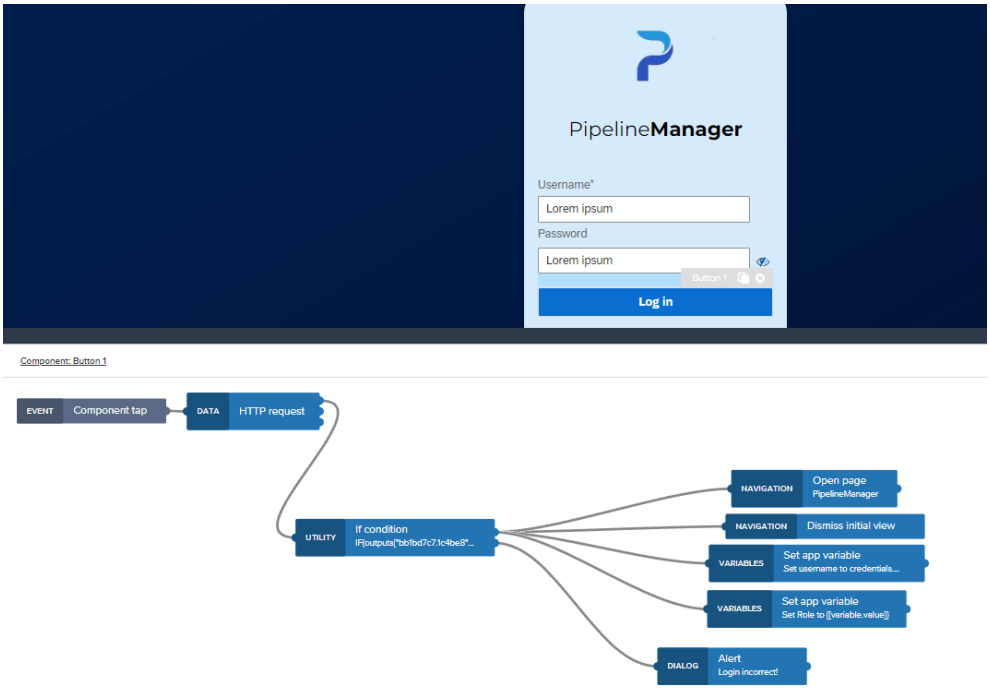


Figura 30 - Validação de Login

5.6.3. Página Pipeline

A página do Pipeline, apresenta uma lista com todos os pedidos de utilizadores através da variável de dados da entidade Tickets. Na lista é possível clicar em cada pedido para verificar os seu detalhes. Ora, de forma a ser possível verificar os detalhes do pedido clicado, criou-se antes de mais uma variável de parametro para a página TicketDetails, para que seja possível identifcar qual o pedido que se deseja ver os detalhes. Posto isto, no evento de *tap* foi definido o componente de navegação para a página de TicketDetails onde é passado o parameter com o número do pedido que foi clicado.

Além da funcionalidade anterior, existe também a possibilidade do utilizador eliminar um pedido. Assim, no icon do lixo foi definida uma lógica no qual existe o componente delete record onde se definiu a entidade Tickets e nas propriedades a informação do pedido que se deseja eliminar através da variável de runtime, designada por current. Posto isto, se o delete for bem sucedido, são despoletadas 2 ações, notificação de sucesso e uma

atualização dos dados que estão a ser apresentados na página. Este "Refresh" está definido no escopo da página e por isso pode ser chamado em qualquer aba de lógica na mesma página. A sua lógica executa um GET à base de dados pela entidade Tickets com o username logado (url_placeholder definido na integração REST) através do componente Get Record Collection. A resposta do GET é novamente associada à variável de dados da entidade Tickets.

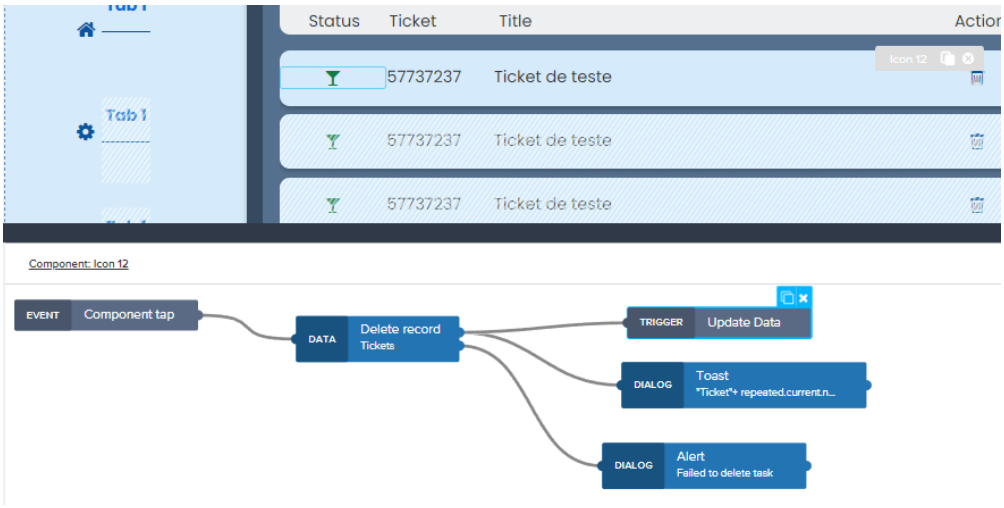


Figura 31 - Eliminação de pedido

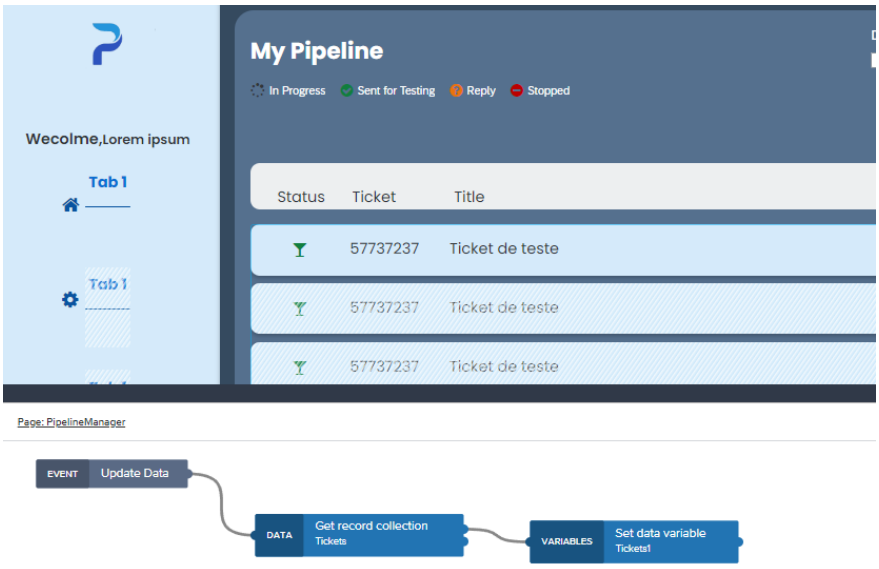


Figura 32 - "Refresh" de dados

5.6.4. Página Add Ticket

A página para a criação dos pedidos, basicamente é um formulário. O preenchimento de todos on inputs está associado a uma variável de página. Isto é, os valores inseridos são guardados na variável. No entanto, para anexar os ficheiros foi necessário usar o componente "pick files" para associar a informação dos mesmos.

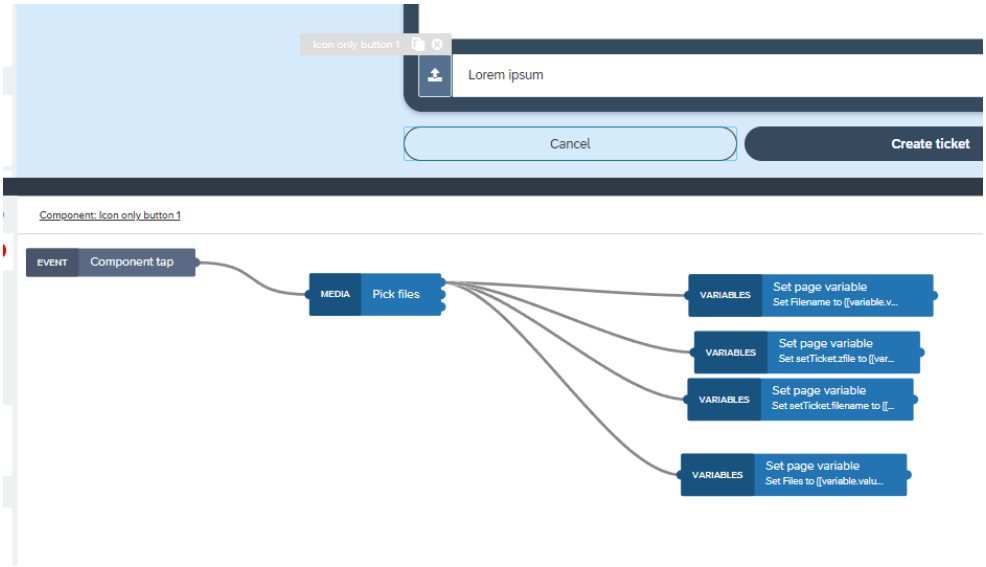


Figura 33 - Upload de ficheiros

No botão "Create Ticket", definiu-se o componente "Create Record" com a entidade Tickets e a informação contida na variável de página. No caso da criação for bem sucedida é despoletada uma mensagem de sucesso.

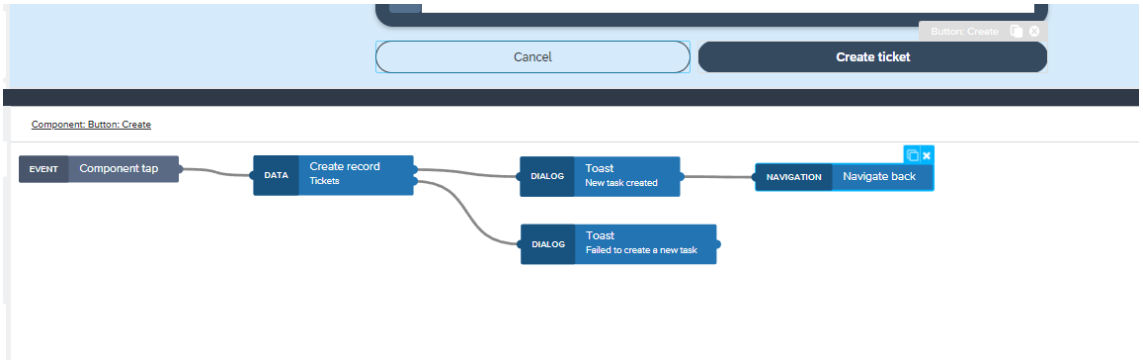


Figura 34 - Criação de um pedido

5.6.5. Página TicketDetails

Inicialmente, na página TicketDetails pensou-se em fazer uma página para a visualização dos detalhes e outra para a edição. No entanto, conseguiu-se tudo numa, através da criação de uma lógica que permitiu alternar entre as duas.

Primeiramente, criaram-se 2 variáveis de páginas do tipo boolean para esse controlo, uma para verificar se se está em modo edição e outra para verificar a utilização de uma dropdown que contém os componentes "Editar" e "Delete". Desta forma, quando se entra nesta página ambas as variáveis estão definidas a "false". A lógica para alternar os modos foi definida no icon de 3 pontos no canto superior direito, onde são despoletadas, através do evento *tap*, uma série de verificações com o componente de condição if, em que se a variável de página de edição estiver a "false", então verifica-se o valor da variável de página da dropdown que no caso de estar a "true", é mostrada a dropdown com os itens "Editar" e "Delete". Caso contrário, se a variável de página da drowpdown estiver a "true", então significa que a dropdown já está visível e por isso passa a não estar. Se a variável de página de edição estiver a "true", significa que entramos no modo de edição e então o icon de 3 pontos passa a ser uma disket e aparece um segundo icon, um "X". Assim, o utilizador fica habilitado a editar os campos User, Status, Priority, Notas e pode também anexar mais ficheiros ao pedido.

Com o modo de edição usa-se a mesma variável de dados que no modo de visualização, de forma a facilitar e otimizar o "refresh" da informação na página. Com o evento *tap* no icon disket verifica-se o valor da variável de página de edição, sendo que se estiver a "true", além do que foi dito anteriormente existe a utilização do componente "Update Record" com a entidade Tickets e a informação da variável de dados. Posto isto, se a atualização for bem sucedida, é feito um "refresh" à página e a variável de página de edição é definida como "false".

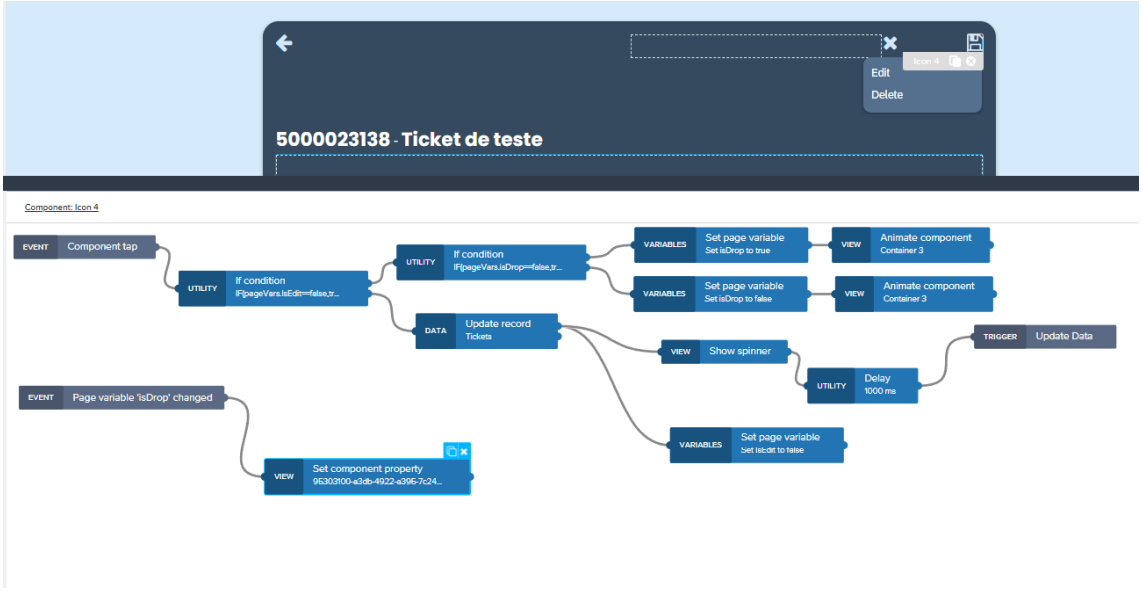


Figura 35 - Atualização de detalhes de um pedido

Relativamente à eliminação de ficheiros, no evento "tap" do icon do lixo, foi definida com o componente "Delete record" com a entidade Tickets e informação do respetivo ficheiro indicado pela variável de dados. O momento de espera de eliminação do ficheiro fica associada ao aparecimento de um spinner que depois de 1 segundo desaparece e permite o "refresh" da lista de ficheiros.

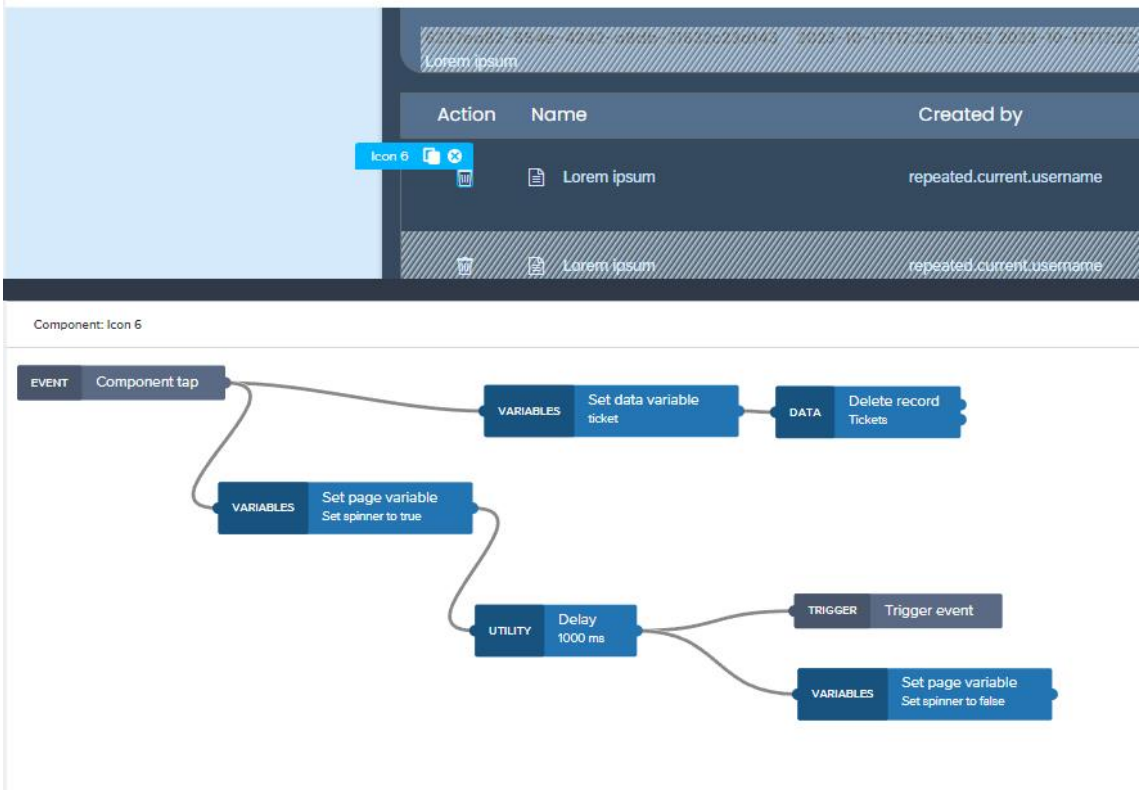


Figura 36 - Eliminação de ficheiro anexado

5.6.6. Página Settings

Nesta página apenas foi definida a possibilidade de tradução de toda a aplicação. No componente "Dropdown" criou-se um lista com duas linguagens, "PT" e "EN". De seguida, atribuiu-se ao evento de *tap* da "Dropdown" o componente de condição IF para verifica a linguagem selecionada pelo utilizador. A escolha da linguagem será efetivada através do componente "Set Current Language".

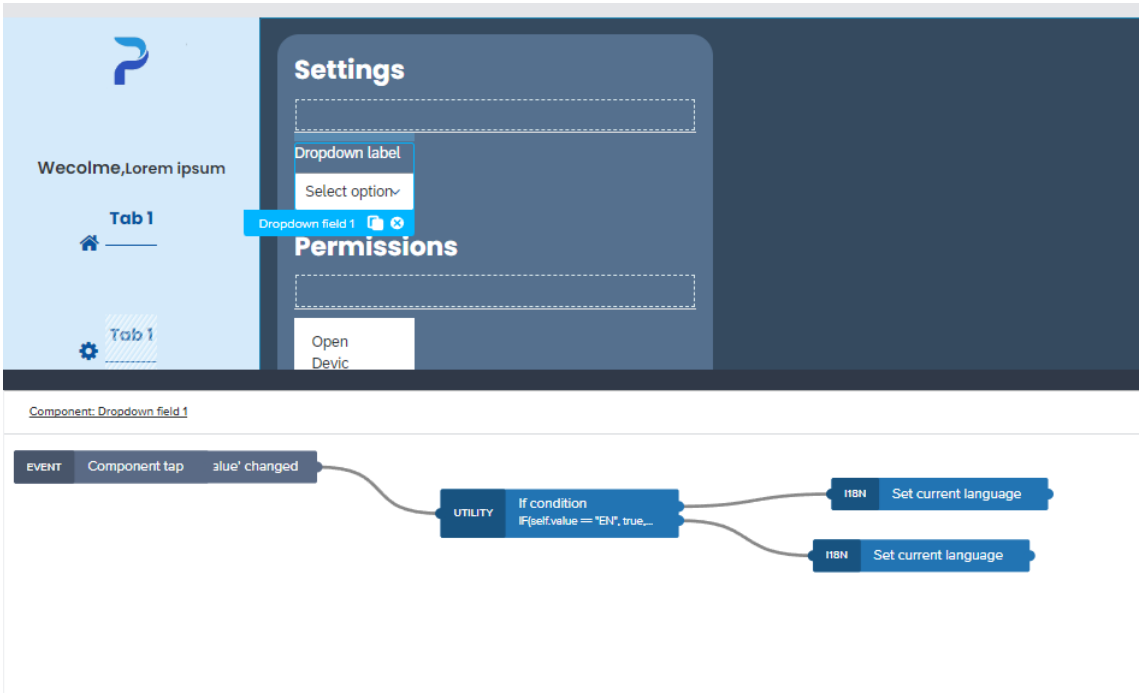


Figura 37 - Tradução de aplicação

5.6.7. Página Statistics

Nesta página não foi necessário definir qualquer tipo de lógica. Serve meramente para disponibilizar dados estatísticos, como o nome indica.

5.6.8. Página Users

Aqui é feita uma listagem de todos os utilizadores da aplicação, onde é possível eliminar ou adicionar novo utilizadores. Além disso, também é possível envir um email para todos os utilizadores através do botão “Send Email”.

Relativamente à funcionalidade de adicionar utilizadores, no botão “New User”, se clicado é despoletado o componente “Set Component Property” em que fica a “true” a visibilidade do container dos inputs para a adição dos utilizadores e a “false” a visibilidade do botão “New User”. Posteriormente ao preenchimento dos inputs que está associado a uma variável de página, procede-se o componente “Create Record” com a entidade Users e a informação da variável de página. Se a criação for bem sucedida, são despoletadas 2 ações: o componente “Set Component Property” é defindo com “false” para a visibilidade do container dos inputs para a adição dos utilizadores e a “true” para

a visibilidade do botão “New User”; “refresh” da página, igualmente como foi explicado em 5.6.3 Página Pipeline mas, neste caso, para a entidade Users.

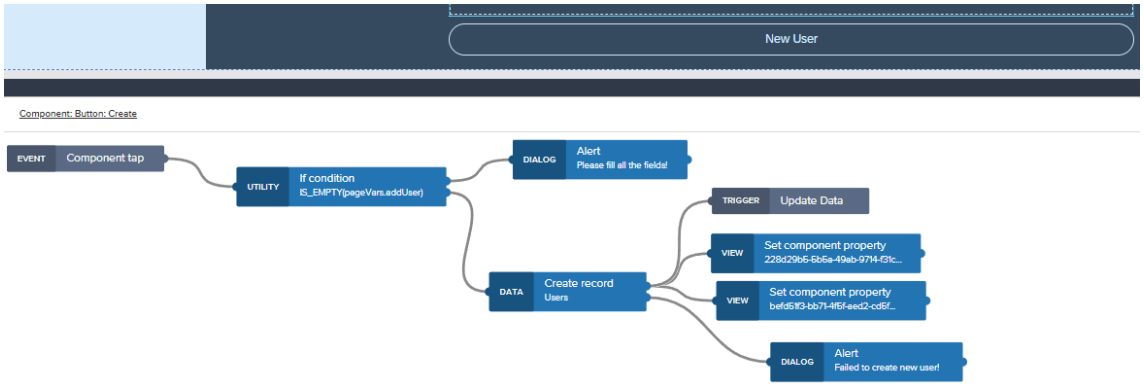


Figura 38 - Adição de novos utilizadores

No que toca á lógica para a funcionalidade de eliminição de um utilizador, definiu-se o componente “Delete Record” para a entidade Users e respetiva informação dada pela variável de runtime, designada por current. Se a eliminação for bem sucedida, é emitida uma notificação de sucesso.

Em relação ao botão “Send Email”, criou-se, primeiramente, um método na classe *handler*, designado por Send_Email. No método, fez-se a seleção de todos os dados da tabela ZPIPELINE_USERS e num clico usou-se essa informação, nomeadamente, o username e email, para a configuração do envio de email.

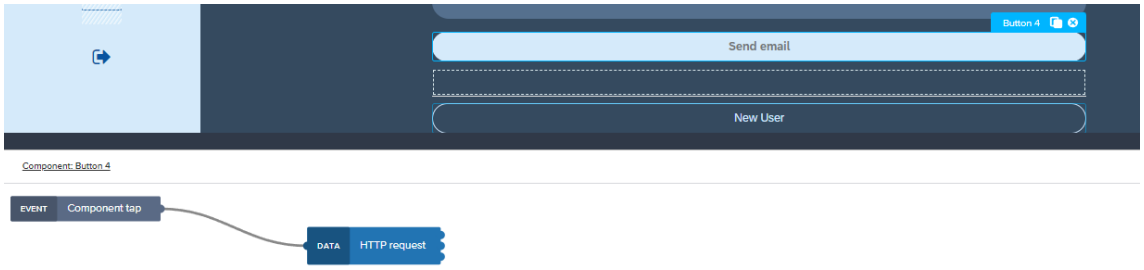


Figura 39 - Envio de email

5.7. Notificações via Email – Automações

Com a finalização do desenvolvimento da aplicação e a respetiva implementação da sua comunicação foi possível começar a implementação das automações das notificações via Email.

Estas automações baseiam-se em programas criado em ambiente SAP com a lógica necessária para o envio do email e a criação de um JOB na transação SM36 (SAP, Scheduling Background Jobs) em SAP com a configuração necessária para executar o programa periodicamente.

Desta forma, implementaram-se 2 programas através da transação SE38. O primeiro verifica a atividade do utilizador, ou seja, se o utilizador estiver inativo durante 7 dias então será enviado um email para relembrar o utilizador para atualizar o ponto de situação. O segundo verifica o pipeline de todos os utilizadores e envia um email aos team leaders com um relatório com o ponto de situação de todos os pedidos dos utilizadores.

5.7.1. Programa de verificação de inatividade

A implementação deste programa foi dividida em 3 fases:

Na primeira fase, declarou-se uma tabela interna do tipo da tabela ZTICKETS e fez-a a seleção de todos os dados da tabela ZTICKET para essa tabela interna.

Na segunda fase, fez-se um ciclo à tabela interna onde se verifica em cada iteração a diferença entre a data do dia atual e data de criação ou atualização dos pedidos. No caso de ser inferior a 7 dias, esses pedidos são eliminados da tabela interna. Caso contrário, mantem-se na tabela para ser usados na terceira fase.

Na terceira fase, declara-se uma estrutura do tipo de linha da tabela ZPIPELINE_USERS e faz-se um ciclo aos restantes dados na tabela interna. Em cada iteração faz-se uma seleção de dados à tabela ZPIPELINE_USERS para a estrutura onde o username coincide com o da tabela interna e, de seguida é feita a construção do corpo do email para o respetivo username, definição do sender com base no respetivo email do username e configuração do email através de métodos standard.

5.7.2. Programa de Ponto de Situação

A implementação deste programa foi dividida em 2 fases.

Na primeira fase, criou-se um programa para gerar o relatório com o ponto de situação de todos os utilizadores. Inicialmente, declarou-se um tabela interna do tipo da tabela ZTICKETS e fez-se a seleção de todos os dados da tabela ZTICKETS para essa tabela interna. Posteriormente, fez-se toda uma configuração necessária ao nível de estruturação do relatório para utilização do módulo de função REUSE_ALV_GRID_DISPLAY_LVC (SE80, SAP REUSE_ALV_GRID_DISPLAY_LVC Function Module) que possibilita a visualização do relatório.

Na segunda fase, criou-se o programa principal que faz uso do primeiro para depois enviar o email. Primeiramente, são usados os módulos de função standard JOB_OPEN, JOB_SUBMIT e JOB_CLOSE (SAP, SUBMIT - job_options) para executar o primeiro programa em modo background e através de uma seleção de dados à tabela TSP01 foi possível obter o ID do relatório que é gerado. Posteriormente, é usado o módulo de função standard CONVERT_ABAPSPOOLJOB_2_PDF (SE80, SAP CONVERT_ABAPSPOOLJOB_2_PDF Function Module) para obter o relatório em PDF. De seguida, é feita a construção do corpo do email para os team leaders, definição do sender com base no respetivo email dos team leader e configuração do email através de métodos standard. A adição do relatório ao email está compreendida nos seguintes pontos: utilização dos módulos de função SX_TABLE_LINE_WIDTH_CHANGE (SE80, SAP SX_TABLE_LINE_WIDTH_CHANGE Function Module) e SCMS_BINARY_TO_XSTRING (SE80, SAP SCMS_BINARY_TO_XSTRING Function Module), utilização do método xstring_to_solix da classe cl_document_bcs e, finalmente o uso do método add_attachment da classe cl_document_bcs para adicionar o ficheiro.

Capítulo 6 Demonstração

Inicialmente faz-se o login com o utilizador Admin.

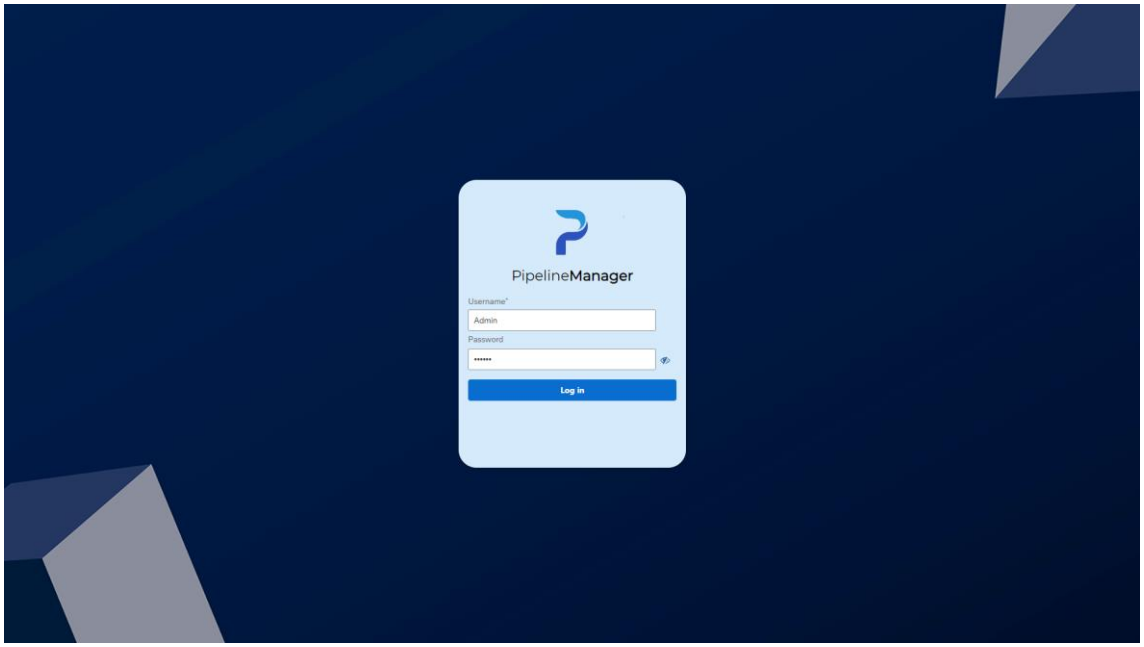


Figura 40 - Demonstração de Login

O utilizador Admin que tem autorizações de admin tem apenas um pedido em estado "Parado", ou seja, ainda por fazer. É possível verificar também que o utilizador tem visibilidade nas abas "Statistics" e "Users". No caso de um utilizador com autorizações de user apenas veria as abas "Home" e "Settings", como é o caso do utilizador Rui na Figura 42 - Demonstração na página Pipeline – Utilizador.

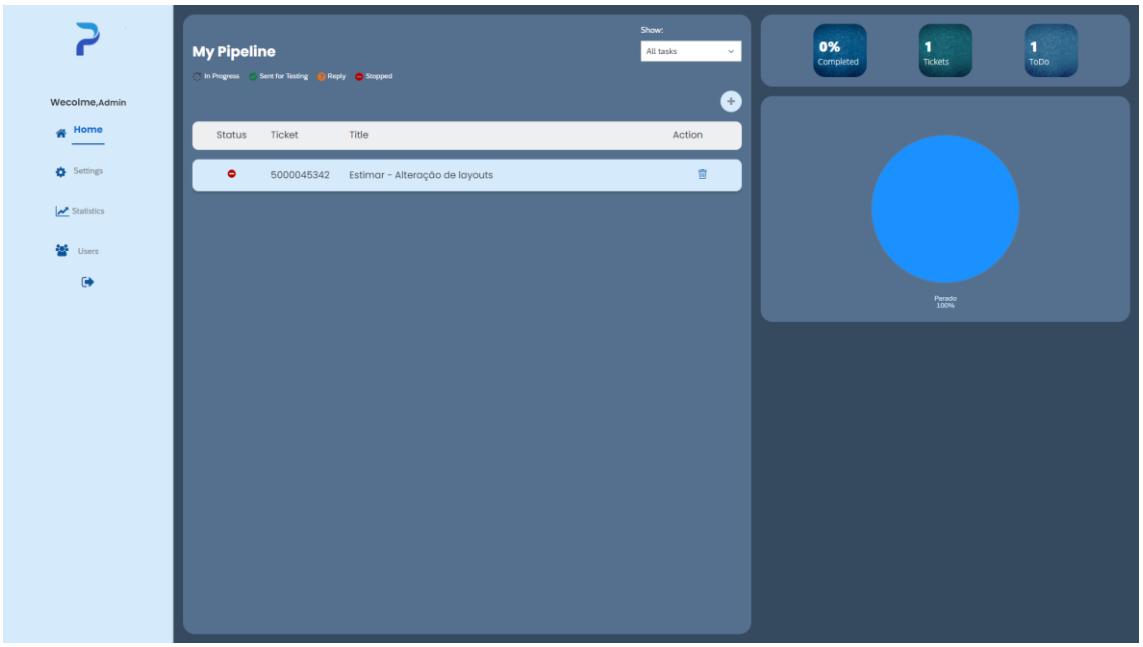


Figura 41 - Demonstração na página Pipeline – Admin

Com o utilizador Rui verifica-se que tem apenas 1 pedido em estado "Enviado para testes" e, por isso, apresenta 100% dos pedidos desenvolvidos.

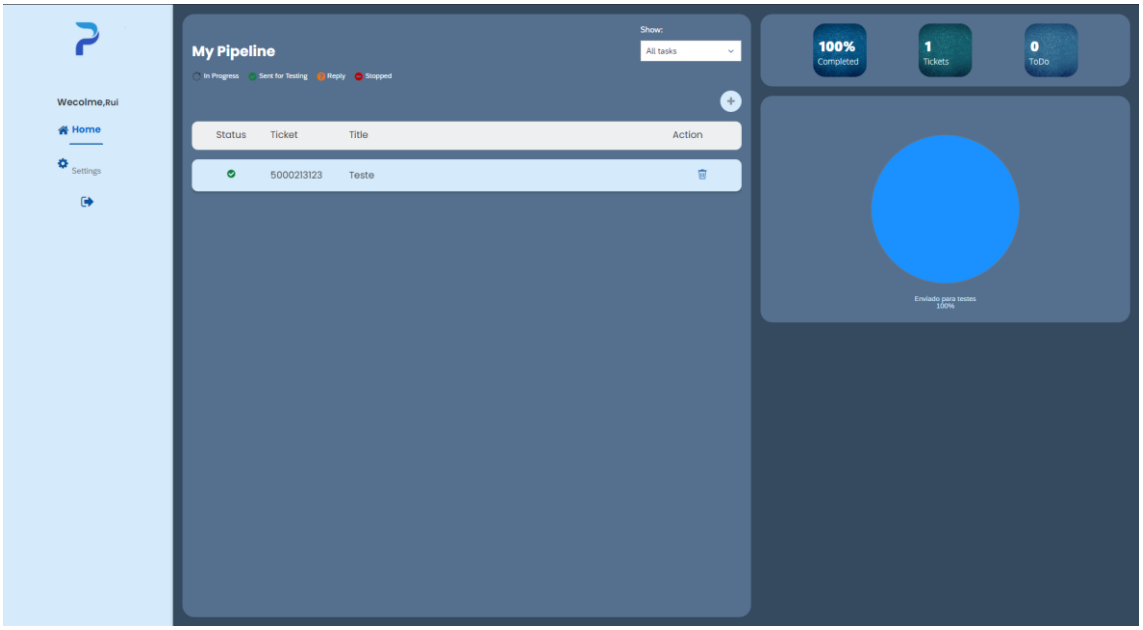


Figura 42 - Demonstração na página Pipeline – Utilizador

Relativamente à página de estatísticas, o utilizador Admin consegue verificar os pedidos desenvolvidos em percentagem, neste caso, 50%. Além disso, pode ver o número de pedidos por estado e por utilizador (em percentagem). A listagem de pedidos por

utilizador também é algo presente e que permite ao Admin monitorizar melhor o ponto de situação de cada utilizador.

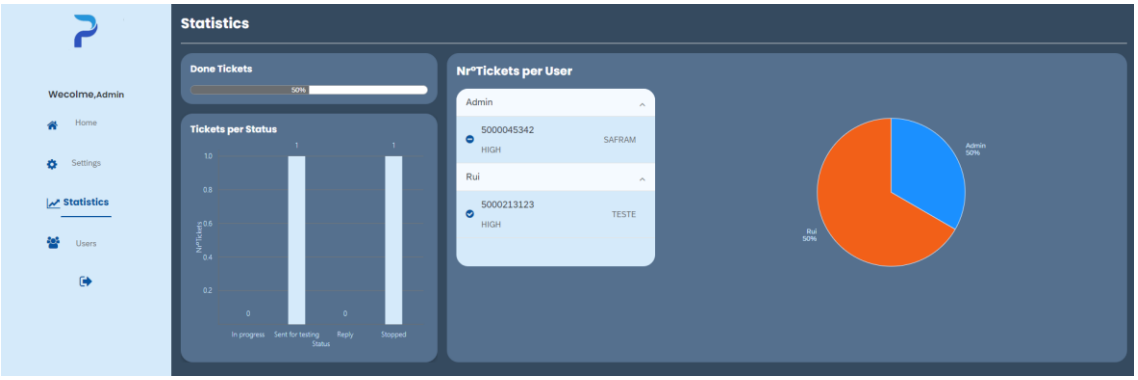


Figura 43 - Demonstração na página Statistics

Na página de utilizadores, o Admin consegue verificar e procurar por utilizadores na aplicação, assim como adicionar e eliminar. Além disso é também possível o envio do email de alerta para os utilizadores atualizarem os seus pedidos através do botão “Send email”.

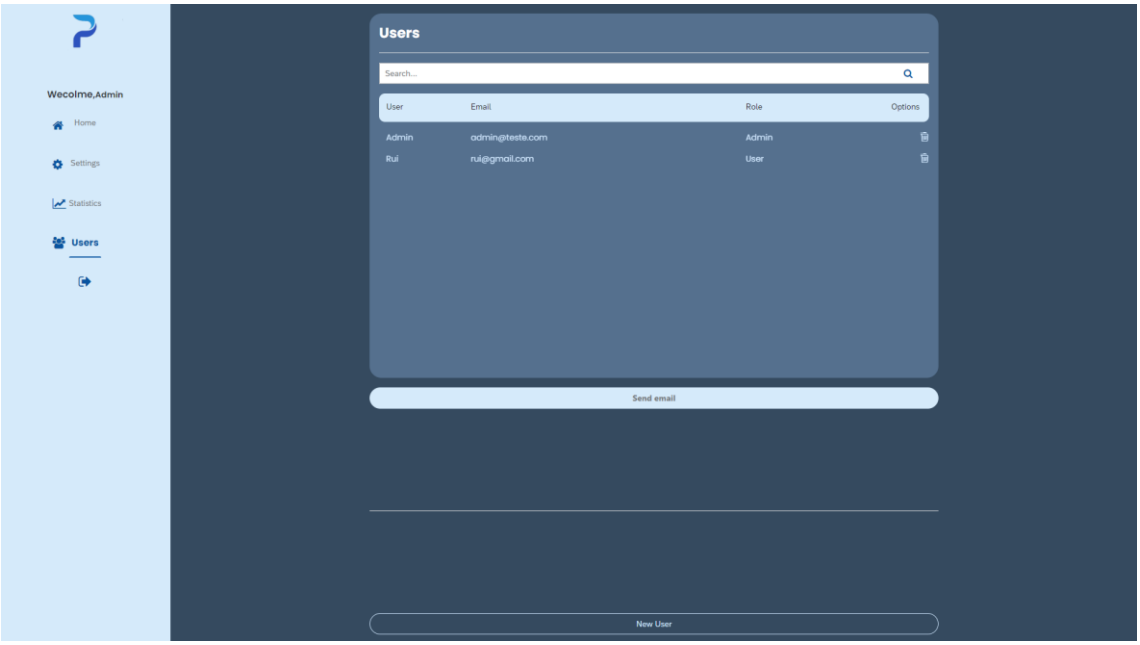


Figura 44 - Demonstração na página Users

Capítulo 7 Conclusões

Numa 1ª fase do projeto, tive alguma dificuldade em iniciar o projeto no SAP Build Apps, uma vez que era necessário ter a subscrição de um plano no SAP BTP. Nesta parte foi necessário perder bastante tempo para analisar bem o fluxo necessário para ter acesso de forma gratuita a essa plataforma e não acabar por subscrever algo pago.

Posteriormente, na implementação houve dificuldades na parte de comunicação entre o SAP e a aplicação que não permitiam o avanço do projeto, uma vez que era necessário instalar os certificados do SAP Build Apps na máquina SAP usada para que fosse estabelecida a conexão. Houve também problemas que não foram ultrapassados pela incapacidade da máquina SAP usada, problemas estes como SMTP não configurado ou instalado e a tecnologia adobe livecycle designer não configurada. Contudo, a funcionalidade de envio de emails foi implementada na mesma e é possível verificar na transação SOST em SAP que ficam em lista de espera. No que toca ao Adobe foi possível usar outra tecnologia que acabou por satisfazer a funcionalidade inicialmente idealizada.

Posto isto, senti bastante facilidade em tudo o que foi implementado em SAP uma vez que é o meu trabalho no dia-a-dia. No entanto, foi a primeira vez que fiz uma integração entre dois sistemas SAP através de um WebService REST e, por isso, foi necessário algum estudo inicial na parte de criação do WebService e implementação da gestão dos pedidos HTTP. Relativamente ao SAP Build Apps, tive de seguir um tutorial (SAP, Tutorial SAP Build Apps) que ajudou bastante para a criação da aplicação, mais especificamente, na implementação de lógica, manuseamento dos componentes e construção da UI.

Finalmente, sinto que a solução implementada cumpriu todos os aspetos idealizados e acredito que este projeto me tenha ajudado ao nível de entendimento do processo de gestão de pedidos por parte de uma empresa. Além disso, aprendi um nova tecnologia que penso que seja uma mais validas para possíveis implementações no futuro.

Bibliografia

Alura. (s.d.). *Figma.* Obtido de https://www.alura.com.br/artigos/figma

Bammidi, N. (s.d.). *Creation of RESTful Webservice in SAP.* Obtido de https://blogs.sap.com/2013/09/16/creation-of-restful-webservice-in-sap/

Hanfi, M. (s.d.). *SAP Cloud Platform Integration.* Obtido de https://blogs.sap.com/2023/01/19/sap-cloud-platform-integration/

Infowester. (s.d.). *O que é um Sistema ERP*. Obtido de https://www.infowester.com/erp.php

JSON. (s.d.). *JSON.* Obtido de https://www.json.org/json-en.html

MDN. (s.d.). *Cross-Origin Resource Sharing (CORS).* Obtido de https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Omie. (s.d.). *Sistema ERP.* Obtido de https://blog.omie.com.br/tudo-sobre-sistema-de-gestao-erp/

Press, S. (s.d.). *ABAP Programming for SAP.* Obtido de https://learning.sap-press.com/abap

Press, S. (s.d.). *How SAP Consultants Create Queries With These Three T-Codes.* Obtido de https://blog.sap-press.com/how-sap-consultants-create-queries-with-these-three-t-codes

SAP. (s.d.). *Adobe forms.* Obtido de https://help.sap.com/docs/SAP_NETWEAVER_700/109111ab6c531014b0f4bc2b77d18606/b764348655fb46149098d95bdca103d0.html

SAP. (s.d.). *Install SSL Certificates.* Obtido de https://help.sap.com/docs/help/e976e9e7cb1b41c18c5299021cb0ed70/5cf57884784a470fb7b5df83df0dacba.html?version=SP8

SAP. (s.d.). *Introduction to the Class Builder.* Obtido de https://help.sap.com/saphelp_aii710/helpdata/en/ee/e440a670a111d1b44c0000e8a52bed/content.htm?no_cache=true

SAP. (s.d.). *Low Code No Code.*

SAP. (s.d.). *Low-code/no-code application development.* Obtido de https://www.sap.com/products/technology-platform/low-code/what-is-low-code-no-code.html

SAP. (s.d.). *Pay As You Go for SAP BTP.* Obtido de https://store.sap.com/dcp/en/news-blogs/announcements/get-started-commitment-free-with-pay-as-you-go-for-sap-btp-now-available-on-sap-store

SAP. (s.d.). *Plataform SAP.* Obtido de https://www.sap.com/portugal/about/what-is-sap.html

SAP. (s.d.). *REST WebService.* Obtido de https://help.sap.com/docs/SAP_DATA_SERVICES/e54136ab6a4a43e6a370265bf0a2d744/5749dada6d6d1014b3fc9283b0e91070.html?version=4.2.9&locale=en-US

SAP. (s.d.). *SAP BTP.* Obtido de https://account.hana.ondemand.com/#/home/welcome

SAP. (s.d.). *SAP Build Apps.* Obtido de https://learning.sap.com/products/sap-build/build-apps

SAP. (s.d.). *SAP Build Apps Service.* Obtido de https://blogs.sap.com/2022/11/25/sap-build-apps-free-tier-individual-access/

SAP. (s.d.). *SAP Solution Manager.* Obtido de https://help.sap.com/docs/SAP_Solution_Manager/c3c5ec585ee248228ddb6c3f08073ea9/b80ac35796fa0e2be10000000a441470.html?version=7.2.07

SAP. (s.d.). *Scheduling Background Jobs.* Obtido de https://help.sap.com/doc/saphelp_nw73ehp1/7.31.19/en-us/4b/2b2954365474fee10000000a421937/content.htm?no_cache=true

SAP. (s.d.). *Smartforms.* Obtido de https://help.sap.com/doc/saphelp_ewm70/7.0/en-US/4e/5ba5f357af4f47e10000000a42189e/frameset.htm

SAP. (s.d.). *SUBMIT - job_options.* Obtido de https://help.sap.com/doc/abapdocu_752_index_htm/7.52/en-us/abapsubmit_via_job.htm

SAP. (s.d.). *Tutorial SAP Build Apps.* Obtido de https://learning.sap.com/learning-journey/utilize-sap-build-for-low-code-no-code-applications-and-automations-for-citizen-developers

SAP, A. M. (s.d.). *SAP Cloud Connector.* Obtido de https://blogs.sap.com/2022/02/03/cloud-connector-explained-in-simple-terms/

SE80. (s.d.). *SAP CONVERT_ABAPSPOOLJOB_2_PDF Function Module.* Obtido de https://www.se80.co.uk/sap-function-modules/?name=convert_abapspooljob_2_pdf

SE80. (s.d.). *SAP DB_CRYPTO_PASSWORD Function Module.* Obtido de https://www.se80.co.uk/sap-function-modules/?name=db_crypto_password

SE80. (s.d.). *SAP REUSE_ALV_GRID_DISPLAY_LVC Function Module.* Obtido de https://www.se80.co.uk/sap-function-modules/?name=reuse_alv_grid_display_lvc

SE80. (s.d.). *SAP SCMS_BINARY_TO_XSTRING Function Module.* Obtido de https://www.se80.co.uk/sap-function-modules/?name=scms_binary_to_xstring

SE80. (s.d.). *SAP SX_TABLE_LINE_WIDTH_CHANGE Function Module.* Obtido de https://www.se80.co.uk/sap-function-modules/?name=sx_table_line_width_change

TutorialsPoint. (s.d.). *SAPScripts.* Obtido de https://www.tutorialspoint.com/sap_abap/sap_abap_sapscripts.htm

Visual Paradigm. (s.d.). Obtido de https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html