INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

# Desenvolvimento técnico em tecnologias SAP

**Paulo Jorge Gonçalves Martins**

*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:

Rui Pedro Sanches de Castro Lopes

**Bragança**

Outubro de 2024

# Resumo

O presente relatório retrata o desenvolvimento do projeto realizado no âmbito do mestrado em Informática.

O projeto desenvolvido teve como propósito o desenvolvimento do *backend* de uma aplicação no *software* SAP de uma ferramenta de registo de horas. Com esta ferramenta, um colaborador vai indicar o trabalho realizado por si durante o expediente, registando a tarefa que efetuou, o tempo investido na conclusão da mesma e o cliente como requerente.

O objetivo deste projeto é otimizar e facilitar o processo de registo de horas, tornando-o homogéneo e de uso global. Pretende-se reduzir as falhas durante o preenchimento das horas recorrendo a mecanismos desenvolvidos especialmente para controlar determinados cenários. Pretende-se igualmente uma ferramenta de fácil acesso e utilização, que permita uma leitura inequívoca dos dados presentes sem comprometer a segurança e a confidencialidade.

O plano de trabalho consiste em três fases. Numa fase inicial, foi realizado um levantamento dos requisitos funcionais a serem desenvolvidos. A segunda fase, consistiu no estudo das tecnologias SAP existentes a utilizar, de modo a corresponder aos requisitos estabelecidos. No que concerne a terceira fase do projeto, foi efetuado o desenvolvimento da solução que corresponde ao desenvolvimento do *backend* da ferramenta em SAP e a sua conexão com um *frontend* previamente estabelecido na aplicação SAP Fiori, aplicação *web* do *software* SAP.

Os dados utilizados durante a realização de testes com o intuito de validar a solução foram genéricos, de forma a manter a confidencialidade exigida.

Os testes foram realizados na última fase do desenvolvimento da solução. Os testes basearam-se no preenchimento de horas relativas a um colaborador e a capacidade da plataforma em cumprir os objetivos pretendidos comprovando a sua viabilidade e o sucesso no seu desenvolvimento.

Palavras-Chave: SAP ABAP, Timesheet, CDS, OData.

# Abstract

This report outlines the development of the project undertaken as part of the master's program in Computer Science.

The project aimed to develop the backend of a time tracking tool within the SAP software. With this tool, an employee will indicate the work performed during their shift by recording the task completed, the time invested to finish it, and the client that request it.

The objective of this project is to optimize and facilitate the time registration process, making it uniform and globally usable. It aims to reduce errors during the register of the hours by implementing mechanisms specifically designed to control certain scenarios. Additionally, the goal is to provide an easily accessible and user-friendly tool that allows for unambiguous data interpretation without compromising security and confidentiality.

The work plan consists of three phases. In the initial phase, a requirement gathering that was conducted to identify the functional requirements to be developed. The second phase involved studying existing SAP technologies that will be utilized, ensuring they met the established requirements. Regarding the third phase of the project, the development of the solution corresponding to the backend of the SAP tool was conducted, along with its connection to a previously established frontend in the SAP Fiori application, which is a web application of the SAP software.

The data used during testing to validate the solution was generic to maintain the required confidentiality.

The tests were executed in the final phase of the solution development. These tests were based on the entry of hours related to an employee and the platform's ability to meet the intended objectives, thereby demonstrating its viability and success in development.

Keywords: SAP ABAP, Timesheet, CDS, OData.

# Índice Geral

# Lista de Siglas/Abreviaturas

**ABAP** *Advanced Business Application Programming.*

**ADT** *ABAP Development Tools***.**

**API** *Application Programming Interface.*

**BTP** *Business Technology Plataform.*

**CDS** *Core Data Services.*

**ERP** *Enterprise Resource Planning.*

**GUI** *Graphical User Interface.*

**IDE** *Integrated Development Environment.*

**ODATA** *Open Data Protocol.*

**SAP** *Systems Applications and Products.*

**SQL** *Structured Query Language.*

# Índice de Figuras

# Índice de Tabelas

# Capítulo 1 Introdução

## 1.1. Contexto

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 o uso de ferramentas de registo de horas é indispensável em consultoria. Esta necessidade levou a que fossem criadas as mais diversas ferramentas que permitissem um controlo e um registo efetivo das horas. Com o passar do tempo, observou-se uma evolução das mesmas para autênticas plataformas de gestão de trabalho onde também é possível controlar e analisar os registos de horas dos colaborados das empresas, alocar períodos de ausência, calcular despesas, entre outras funcionalidades.

Apesar das inúmeras ferramentas presentes no mercado, o recurso ao Microsoft Excel continua a ser uma possibilidade para o registo de horas, usufruindo da versatilidade desta aplicação e por ser uma ferramenta de baixo custo, alto rendimento e vastamente utilizada e conhecida.

Contudo, com o aumento do número de entradas presentes no documento, começam-se a detetar problemas relativamente à performance e a fiabilidade dos dados presentes na aplicação. Vislumbrou-se a necessidade de melhorar a ferramenta utilizada para registar as horas, vulgarmente designada como *timesheet*, recorrendo a um *software* capaz de conciliar todos os pressupostos definidos e assegurar um correto funcionamento da ferramenta e com capacidade de suportar a demanda exigida. A solução recaiu na utilização do *software* SAP devido ao conhecimento intrínseco das suas capacidades como uma plataforma de gestão de negócio.

A possibilidade do desenvolvimento tanto do *backend* como do *frontend* em tecnologias do SAP tornou-o na solução indicada. Contudo neste relatório vai-se abordar o desenvolvimento do *backend*, incluindo todo o processo efetuado até se chegar ao produto final.

## 1.2. Objetivos

O objetivo principal deste projeto é o desenvolvimento do *backend* de uma ferramenta que possibilite o registo e controlo de horas, com distinção de utilizadores tendo em consideração as permissões de acesso estabelecidas que vão derivar do perfil associado a cada utilizador e das funcionalidades designadas a cada um desses perfis. Esta distinção dos níveis de autorização estabelecidos para cada utilizador facilita no processo de preenchimento das horas e posteriormente de análise dos dados inseridos, permitindo aos *team leaders* e aos *managers*, encarregues dessas tarefas, efetuar uma análise mais detalhada e eficiente dos dados provenientes da sua equipa e executar assim estimativas temporais dos desenvolvimentos e orçamentos mais realistas para os futuros projetos em que possam estar envolvidos além de detetar anomalias nos registos com uma maior exatidão e rapidez.

Pretende-se assim criar uma ferramenta no SAP que seja capaz de cumprir com todos os requisitos estabelecidos, funcionais e não-funcionais, com uma base de dados adaptada as necessidades identificadas e com a conexão à aplicação *web*, designada de Fiori, que permita uma recolha de dados provenientes dos utilizadores sem recorrer a um sistema externo.

## 1.3. Estrutura do documento

Segue-se a estrutura do relatório que visa facilitar a leitura e a analise do trabalho realizado.

Neste primeiro capítulo encontra-se a introdução ao tema e dos objetivos proposto neste projeto.

No segundo capítulo é apresentado estado da arte.

No terceiro capítulo abordam-se as ferramentas e as tecnologias que serviram de apoio para a implementação da solução definida.

O quarto capítulo alberga a modelação da solução e no quinto capítulo encontra-se a arquitetura da solução. Nesse sentido, são apresentados os requisitos funcionais e não-funcionais da aplicação assim como o modelo da base de dados desenhada para a solução a implementar, assim como uma descrição da arquitetura da ferramenta e do seu funcionamento expectável.

No sexto capítulo encontra-se todo o processo de desenvolvimento de uma forma pormenorizada e metódica, dividida pelas etapas necessárias para a implementação deste projeto.

No penúltimo capítulo é apresentado o produto final que surgiu de todo o trabalho efetuado.

No último capítulo é realizada uma conclusão, onde se apresentada uma síntese global, assim como uma perspetiva relativa a melhorias a desenvolver futuramente.

# Capítulo 2 Estado da Arte

Neste capítulo realiza-se uma análise aprofundada do *software* SAP e as suas valências que levaram a que se optasse pelo mesmo para a implementação da solução. Serão abordadas as capacidades do *software* SAP assim como duas ferramentas similares existentes no mercado das aplicações de registo de horas.

## 2.1. Ferramentas de registo de horas

Existem ferramentas que permitem efetuar *time tracking* do trabalho realizado. Estas ferramentas podem ser de uso genérico como o Microsoft Excel ou com um propósito direcionado para o registo de horas como a *Clockify*, presente na Figura 1, ou a *Factorial*, presente na Figura 2 [1] [2].

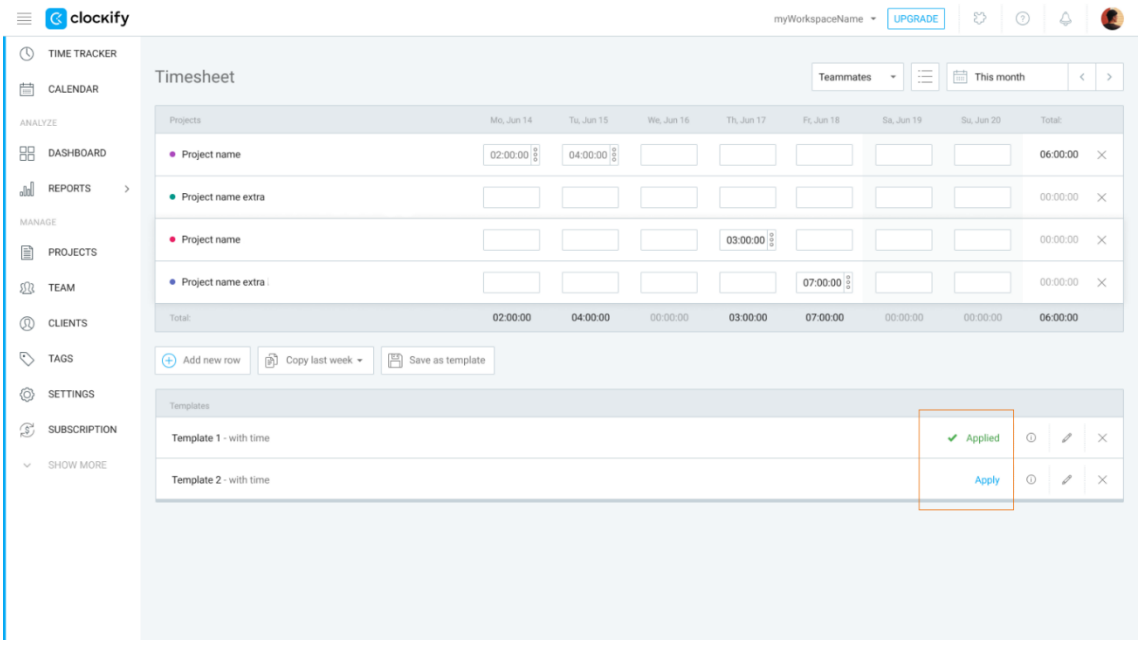


Figura 1 - Página da timesheet da ferramenta *Clockify* [3].

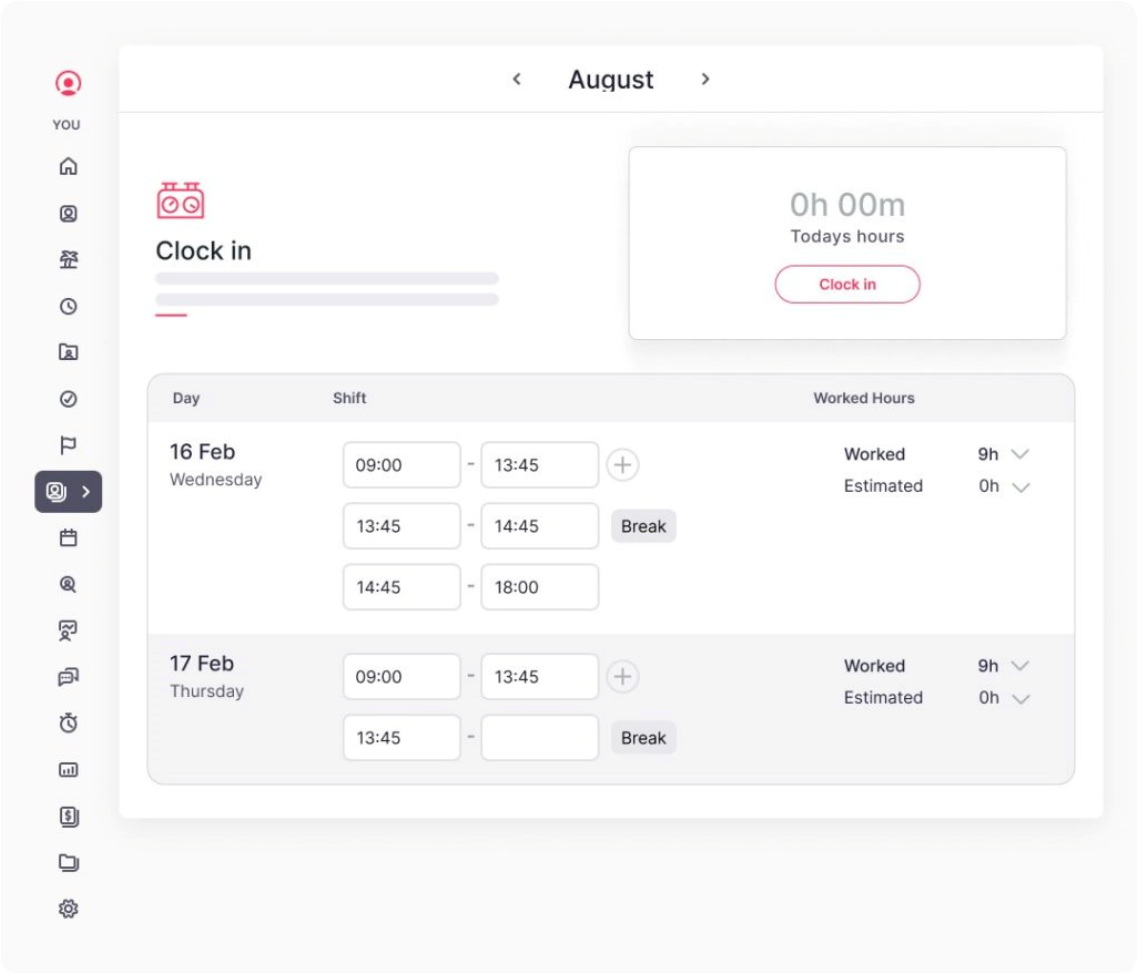


Figura 2 - *Time tracking* na ferramenta *Factorial* [4]

Cada ferramenta tem as suas características, o que as torna únicas, apesar das mesmas puderem ser adaptadas às necessidades do utilizador. Além disso cada uma tem o seu preço e dependente dos planos a subscrever tem diferentes funcionalidades, a vertente financeira seria sempre importante a ter em conta visto que o valor cresce quando maior for o número de utilizadores com necessidade de utilizar a ferramenta [5] [6].

Com a influência que a Microsoft tem no mundo empresarial, tornando-se o seu sistema operativo *Windows* e as ferramentas do mesmo praticamente indispensáveis a qualquer empresa e sendo este um serviço pago, divergir da utilização de uma ferramenta como o Microsoft Excel teria sempre um custo acrescentado ao custa já vinculado às aplicações fornecidas pela Microsoft [7].

O facto do *software* SAP além de estar disponível para uso sem necessidade de uma nova subscrição, tal com o Microsoft Excel, é uma aplicação previamente subscrita e ter as capacidades/funcionalidades que permitem criar uma ferramenta de registo de horas

aliado ao conhecimento existe sobre o produto e da familiarização com o mesmo, tornou-o na escolha indicada para resolver o problema identificado, evitando uma sobrecarga financeira com uma nova ferramenta e o tempo investido na familiarização e personalização da nova ferramenta é, invés disso, investido no desenvolvimento da solução.

## 2.2. SAP

A SAP SE é uma empresa alemã fundada em 1972 por cinco ex-funcionários da IBM, que pretendiam criar um *software* padrão para o processo de gestão de um negócio. Esta empresa multinacional, tem uma presença global, sendo reconhecida pelo desenvolvimento em *software* de planeamento de recursos empresariais (ERP) [8].

Figura 3 - Logotipo da SAP

O ERP, que significa "*Enterprise Resource Planning*"( em português, Planeamento de Recursos Empresariais), permite executar vários processos centrais num único sistema integrando departamentos como finanças, logística, produção, despesas ou compras. Esta é uma tecnologia líder no setor, que se foi estabelecendo ao longo de mais de quarenta anos e que se foi adaptando aos novos requisitos do mercado com a evolução para a *cloud* e a possibilidade de dimensionamento dependente do que lhe é exigido [7].

A capacidade e abrangência do ERP advém dos seus inúmeros módulos que são capazes de satisfazer as exigências que um negócio pode enfrentar recorrendo os módulos presente na Tabela 1.

Tabela 1- Módulos disponibilizados pela SAP para o ERP

| **Sigla** | **Descrição** |
|---|---|
| CO | *Controlling* (Contabilidade) |
| FI | *Finance* (Finanças) |
| HR | *Human Resources* (Recursos humanos) |
| MM | *Material Management* (Gestão de materiais) |
| PM | *Process Maintenance* (Manutenção de edifícios) |
| PP | *Production Planning* ( Planeamento de produção) |
| QM | *Quality Managemen*t (Gestão de qualidade) |
| SD | *Sales and Distribution* (Vendas e distribuição) |
| WM | *Warehouse Management* ( Gestão de armazém) |


Figura 4 - Módulos SAP ERP [10]

Para conseguirem aceder ao ERP, os utilizadores recorrem a aplicação SAP GUI. Esta aplicação, que tem como alvo o cliente que adquiriu a licença para o uso das tecnologias SAP, permite ao utilizador aos diferentes sistemas SAP. Após realizar a autenticação no sistema, através do SAP Logon, como demonstrado na Figura 5, são direcionados para o ecrã principal da aplicação, designado SAP *Easy Access*, Figura 6 [11].

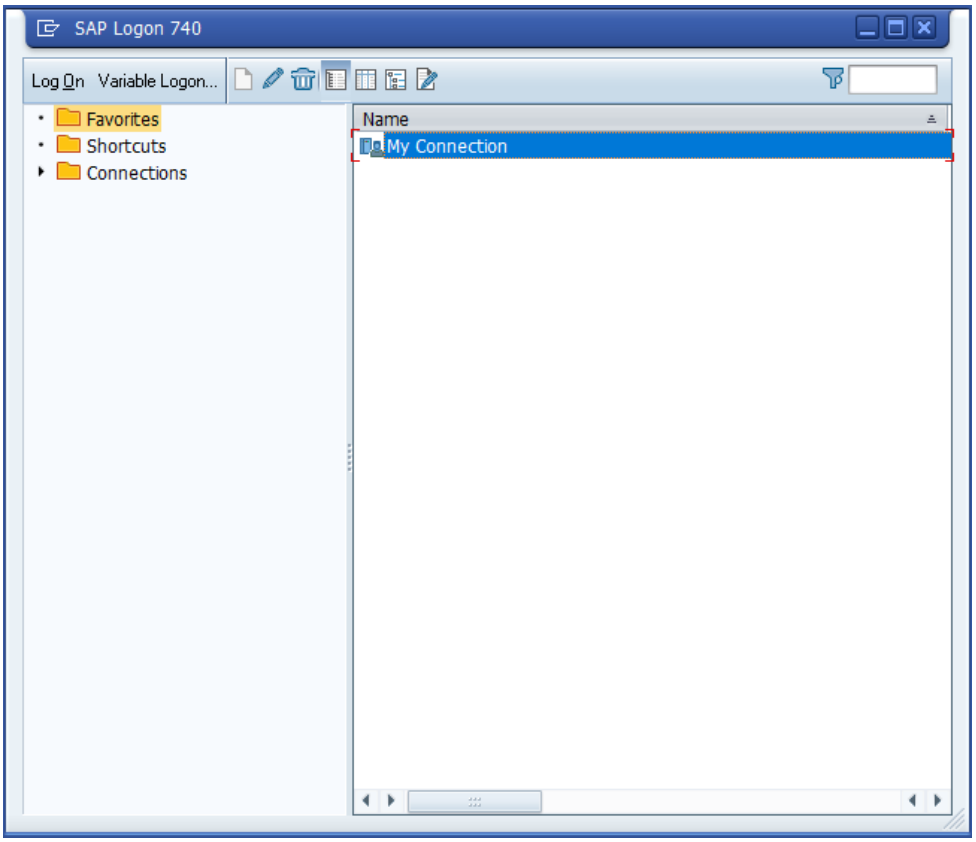


Figura 5 - SAP Logon [11].

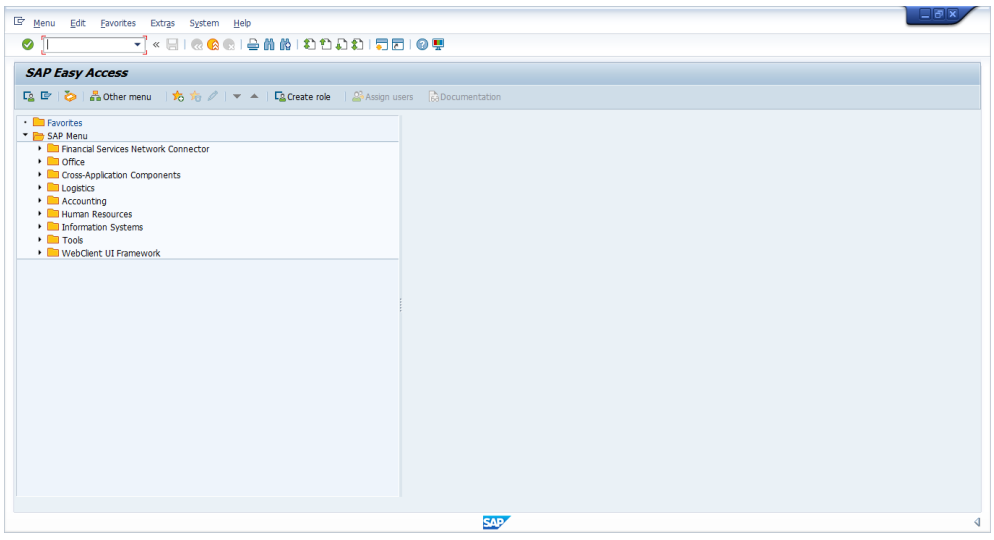


Figura 6 - SAP *Easy Access* [11].

A partir do SAP *Easy Access* é possível aceder a operações como realizar um pedido de compras, emitir faturas, visualizar ou criar um relatório, criar tabelas, entre outras. Mas todas as operações tem associado um código exclusivo designado como transação. Ao pesquisar diretamente por este código na barra de pesquisa, acede-se diretamente à transação pretendida.

Na Tabela 2, são apresentadas algumas das transações utilizadas no âmbito do desenvolvimento deste projeto.

Tabela 2 - Transações utilizadas no SAP GUI.

| **Código da Transação** | **Descrição** |
|---|---|
| SE11 | Permite o acesso ao dicionário de dados ABAP |
| SE16N | Permite a exibição geral de tabelas e os seus dados |
| SE24 | Permite o acesso a classes e aos seus métodos |
| SE37 | Permite a gestão de módulos de função |
| SE38 | Permite a gestão de programas |
| SE80 | Permite o acesso ao navegador de objetos ABAP |
| SE91 | Permite a gestão das classes de mensagens |
| SM30 | Permite adicionar novas entradas a uma tabela |
| ST22 | Permite visualizar os *logs* resultantes de um *dump*. |
| SEGW | Permite criar o modelo de dados e gerar um serviço |
| /IWFND/MAINT_SERVICE | Permite ativar um OData e o seu respetivo serviço |
| /UI2/FLP | Permite o acesso ao *launchpad* de Fiori *web app* |

Quando falamos da organização interna da SAP, é fundamental conhecer o conceito de Dados Mestre que envolve todo o tipo de informação necessária para um bom funcionamento do negócio como dados de clientes, fornecedores, contabilidade,

matérias, entre outros. O acesso aos dados presentes nas tabelas especificas requerem a utilização do Mandante.

O Mandante é uma unidade organizacional no sistema que representa o grupo comparativo e representa, ao mesmo tempo, o mais alto nível da hierarquia no sistema SAP. Dentro de um Mandante é possível criar um número ilimitado de empresas, cada uma com as suas tabelas e configurações. Deste modo, evitam-se duplicações de dados e informações, tornando o Mandante um espaço de dados exclusivo e único.

# Capítulo 3 Tecnologias/ Ferramentas

## 3.1. SAP ABAP

SAP ABAP, que significa *Advance Business Application Programing*,( em português, programação avançada para aplicações empresariais) é a linguagem de alto nível desenvolvida pela e para o *software* SAP.

Um ambiente ABAP é um ambiente de desenvolvimento e de execução totalmente customizado para satisfazer as necessidades do negócio. A linguagem suporta uma programação orientada a objetos, reduzindo assim a quantidade de código necessário e permite assim a reutilização do mesmo [12].

## 3.2. SAP FIORI

O SAP Fiori é um sistema de design desenvolvido pela SAP cuja o objetivo é ser utilizada pelos seus clientes, parceiros e pela própria SAP em aplicações empresariais, principalmente como aplicação *web* [13].

As aplicações que utilizam esta linguagem são frequentemente designadas por aplicações Fiori ou interfaces Fiori User Interface(FUIs).

## 3.3. ECLIPSE IDE e ABAP Development Tools (ADT)

A plataforma ECLIPSE é um IDE, que significa *Integrated Development Environment* (em português, ambiente de desenvolvimento integrado). Um IDE é um ambiente de desenvolvimento que permite programar em diversas linguagens devido ao seu suporte a diferentes *plugins* e *add-ons* [14].

ABAP *Development Tools* (ADT), permite aos programadores de ABAP efetuar as tarefas de desenvolvimento de aplicações num IDE com base na plataforma ECLIPSE, com essa IDE a possibilitar aos programadores desenvolverem para a linguagem ABAP [15]. É uma ferramenta que irá evoluir e na qual a SAP tem expectativas que possa substituir o SAP GUI no futuro, tornando-se a IDE primária para a programação em ABAP.

## 3.4. *Core Data Service* (CDS)

Com a adoção do ADT para a plataforma ECLIPSE, surgiu a possibilidade de criar as CDS. Estas CDS têm visões, que são uma projeção das tabelas presentes na base dados com os campos e os dados existentes sobre as quais a visão se encontra relacionada. Numa visão também é possível adicionar anotações e associações de forma a aumentar as capacidades da visão CDS e do que a mesma pode alcançar [16].

## 3.5. Visual Paradigm

O Visual Paradigm é uma ferramenta multiplataforma de conceção e gestão de sistema de TI. Nesta ferramenta é possível criar diagramas que definem o funcionamento e estrutura de uma aplicação [17].

# Capítulo 4 Requisitos e Modelação

Neste capítulo é apresentada a arquitetura definida para a solução implementada.

São abordados os requisitos funcionais e não-funcionais e os diagramas de casos de uso e de atividades, tendo com objetivo demonstrar com clareza e detalhe o funcionamento pretendido para a solução.

## 4.1. Requisitos Funcionais

### 4.1.1. Utilizador base com acesso ao preenchimento de horas sem visão dos restantes utilizadores.

Um utilizador com permissão de nível baixo pode preencher e visualizar as suas horas registadas na ferramenta. Apenas terá acesso aos seus registos, sendo assim impossibilitado de aceder a registos de outros utilizadores, garantido assim uma maior confidencialidade.

### 4.1.2. Controlo de horas registadas diárias e quinzenais.

Após efetuar o registo das horas nas células indicadas para o efeito tendo em consideração o dia do mês que se está a preencher, o sistema realiza uma verificação à

quantidade de horas registadas para cada dia e na totalidade dos quinze dias preenchidos. Com isto o sistema poderá identificar falhas como excesso/ausência de horas, quando não são atingidas as cotas diárias ou quinzenais, ou o preenchimento da célula errada, como preencher horas numa célula em que o dia se estipula como pertencente ao fim de semana.

### 4.1.3. Visualização/adição de projetos e clientes.

Permitir aos utilizadores com autorizações de configurador para criar clientes e *engagments* para os mesmos ou visualizar os clientes presentes na ferramenta e os respetivos *engagments* associadas a cada um deles.

### 4.1.4. Visão de gestão com visualização do registo de horas dos utilizadores.

Um utilizador com autorizações de perfil de gestor poderá visualizar as horas preenchidas pela equipa ao qual ele se encontra responsável para analisar as horas registadas e submetidas.

### 4.1.5. Possibilidade de configuração de utilizadores, clientes e calendários.

Permitir a um utilizador com o nível de autorização referente ao perfil de configurador, para o mesmo gerir clientes, restantes utilizadores e *engagments* existentes, além de registar ou eliminar clientes ou *engagments* ou alterar o perfil e o consequente nível de autorização alocado ao mesmo para um determinado utilizador.

### 4.1.6. Utilizadores com acesso personalizado.

Criação de um processo de autenticação para filtrar o conteúdo apresentado a cada utilizador, tornando o mesmo individualizado e único, de modo a garantir que cada utilizador apenas tem acesso para realizar algum tipo de edição ou visualização das suas horas registadas.

## 4.2. Requisitos não-funcionais

### 4.2.1. Performance e fiabilidade

O desenvolvimento desta ferramenta visa acondicionar um elevado número de acessos simultâneos e tratar de uma elevada quantidade de dados, evitando que os mesmos sejam corrompidos e que estejam guardados por elevados períodos de tempo. Com a elevada quantidade de informação presente no sistema, pretende-se que o mesmo mantenha uma performance elevada, com uma rápida resposta no que diz respeito ao tratamento de dados e envio dos mesmos entre sistemas.

### 4.2.2. Segurança

Com esta abordagem pretende-se atingir uma maior segurança, restringindo o acesso aos utilizadores, permitindo que apenas possam consultar determinados dados consoante as permissões indicadas para cada um, mantendo uma maior confidencialidade entre projetos e facilitando a vistoria dos mesmos pelos gestores responsáveis.

O facto de ser uma ferramenta desenvolvida localmente, também beneficia a segurança sendo que os dados apenas podem ser consultados e geridos por membros devidamente autorizados e pertencentes à empresa onde a ferramenta fora desenvolvida/adaptada, mitigando assim o acesso externo aos dados adicionados.

### *4.2.3.* Utilidade e *user friendly.*

Ao recorrer à ferramenta de desenvolvimento web da SAP, designada Fiori, permite uma *user interface* mais confortável e de fácil utilização por parte dos utilizadores, facilitando componentes como a criação de *engagments* e de clientes ou a consulta e submissão dos dados na ferramenta.

## 4.3. Diagrama de casos de uso

O diagrama de Casos de Uso, presente na Figura 7, evidencia as funcionalidades que o utilizador da aplicação pode desempenhar. Temos presentes três atores, condizentes com os três perfis de utilizadores existentes na nossa solução.

Qualquer utilizador ou gestor pode preencher a sua *timesheet*, mas apenas o gestor pode validar as mesmas após as analisar, desde que os utilizadores que preencheram essas *timesheets* estejam integrados na mesma área que o gestor. O utilizador só terá acesso à sua *timesheet*.

Os configuradores têm como principal função permitir que todo o processo de registo de horas ocorra sem percalços, isto é, este ator tem de garantir que todos os clientes e os respetivos *engagments* estão presentes no sistema e aptos a serem utilizados pelos utilizadores durante o preenchimento da sua *timesheet*.

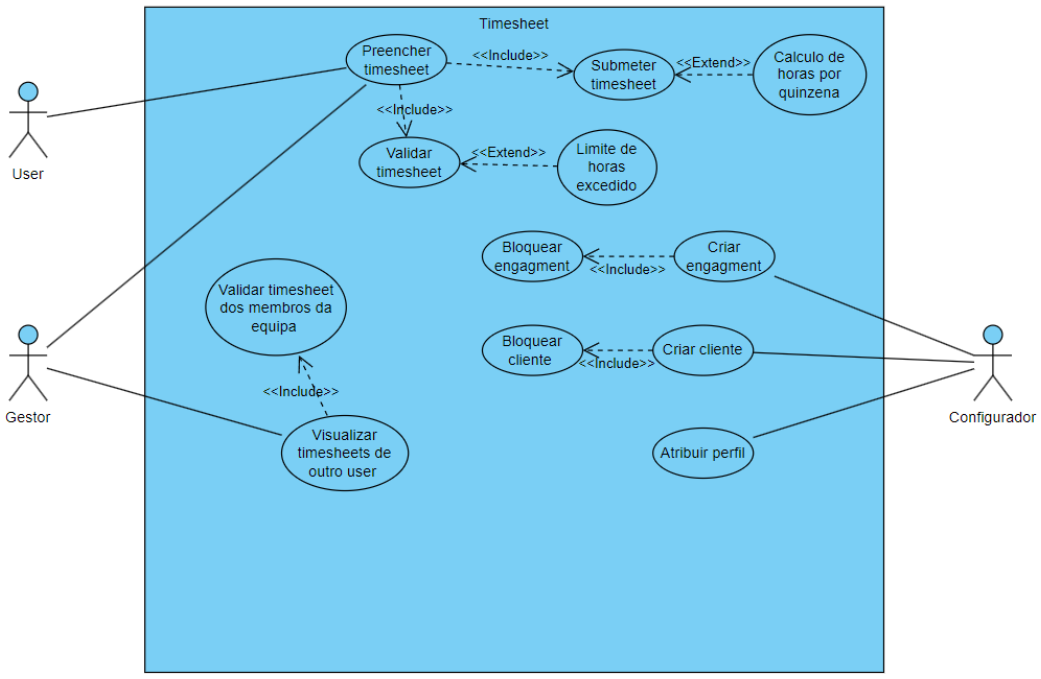


Figura 7 - Diagrama de casos de uso

## 4.4. Modelo da base de dados

Na Figura 8 encontra-se o modelo de base de dados que representa a modelação idealizada para a solução.

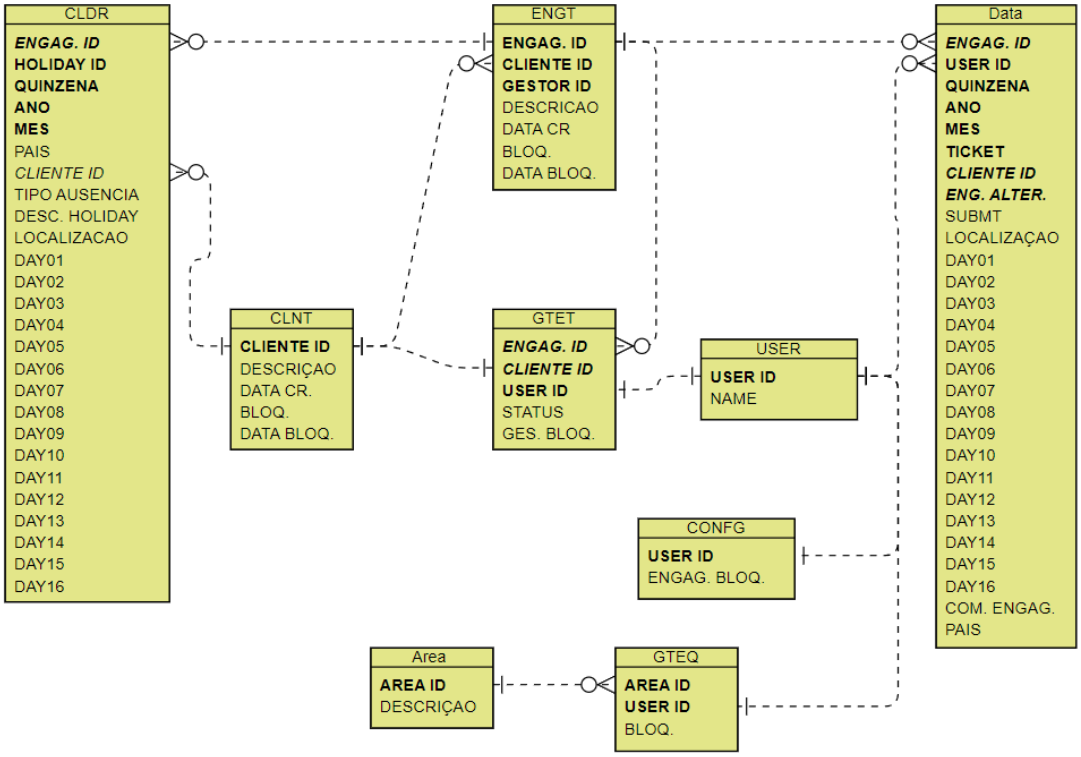


Figura 8 - Modelo da base de dados da ferramenta

# Capítulo 5 Arquitetura

Todos os utilizadores irão realizar a autenticação na ferramenta e posteriormente preencher o registo de horas, designado de *timesheet*, para um período quinzenal. Caso o seu perfil seja o de utilizador base, então apenas podem visualizar a sua *timesheet.* Durante este registo será indicado o cliente para o qual trabalharam, assim como um código predefinido que se encontra conectado com o cliente e que serve para identificar qual foi o trabalho realizado, este código é designado de *engagment.* O *engagment* permite identificar as diferentes tarefas realizadas para um cliente e identificar o que se realizou visto que cada *engagment* tem uma breve descrição do seu propósito. Após isto o utilizador vai colocar as horas investidas na realização desta tarefa e assim sucessivamente até preencher as horas diárias, semanais e quinzenais.

Um perfil de gestor poderá analisar/verificar as *timesheets* relativas à sua equipa. Um gestor terá acesso aos *engagments* criados apenas para o seu posto, diferenciando-o dos restantes utilizadores.

Um utilizador com um perfil de configurador pode criar/bloquear *engagments* e clientes, sendo uma peça fundamental para o correto funcionamento da ferramenta. Todo o processo de inserção de dados é realizado através do *frontend* e comunicado para o *backend* através do serviço OData. Após receber os dados no *backend,* os mesmos são tratados de forma a corresponder ao pedido efetuado pelo utilizador na aplicação *web* de Fiori.

# Capítulo 6 Desenvolvimento

Neste capítulo é abordado o desenvolvimento efetuado em prol da construção da solução.

## 6.1. Introdução

O desenvolvimento desta ferramenta inicia-se com a criação das tabelas necessárias para a implementação da solução dentro da base de dados do sistema SAP. Nesta etapa, juntamente com as tabelas e as respetivas estruturas, criaram-se os elementos de dados personalizados para definir a tipologia de dados inseridos.

Numa segunda etapa, efetuou-se a criação das visões CDS, que são uma ferramenta presente no eclipse. Tem por base as tabelas criadas na base de dados da SAP e visa disponibilizar as funcionalidades da solução desenvolvida além de melhorar a performance da ferramenta. Possibilita a conexão das tabelas com o OData criado posteriormente.

Em seguida criou-se os serviços OData com base nas visões CDS criadas anteriormente. Possibilitou-se assim a comunicação e a troca de informação entre o Frontend, desenvolvido em Fiori (plataforma web proveniente do SAP que permite aos utilizadores manusear os dados com uma interface mais amigável) e o backend da ferramenta desenvolvida no sistema SAP onde se encontra a base de dados que sustenta a ferramenta.

Como as CDS são visões da base de dados, foi necessário implementar as funcionalidades de criação, atualização e eliminação dos registos associados às tabelas das visões CDS. Para tal desenvolveu-se classes em linguagem ABAP, que recebem a informação proveniente dos serviços OData desenvolvidos.

Para acomodar todo o desenvolvimento, foi criado um único pacote nomeado ZHRA001 [18].

Concentrou-se neste pacote todas as tabelas da base de dados, visões, categorias de tabelas, estruturas, domínios dos elementos de dados, *data definitions*, classes e os respetivos métodos e os grupos de função se se viriam a utilizar [19] [20] [21].

## 6.2. Base de dados

A estrutura da base de dados fora desenvolvida de raiz e é totalmente personalizada. Para tal, criou-se todos os elementos de dados, que nada mais são do que a tipologia dos dados que aquele campo irá receber e o comprimento máximo que o campo, é capaz de receber, isto é, se o campo é do tipo char8 (caracter com um comprimento de 8 caracteres) ou um int4 (número inteiro com comprimento máximo de 4 caracteres).

A base de dados consiste em 9 tabelas, como é possível verificar na Figura 9, todas com a sua especificidade e propósito.


Figura 9- Tabelas presentes no dicionário de objetos.

Para auxiliar nas boas práticas da programação e desenvolver um código mais limpo possível, criou-se uma tabela, ZHRA001_TB_CONST, com o intuito de guardar os valores atribuídos às variáveis utilizadas como constantes fixas que estariam presentes na lógica desenvolvida, facilitando assim uma alteração futura concentrando todas as variáveis constantes numa tabela apenas, evitando-se que as mesmas estejam presentes como *hard code.*

Criou-se duas tabelas semelhantes, ZHRA001_TB_DATA e ZHRA001_TB_CLDR, com o intuito de separar o preenchimento das horas destinadas para um cliente/projeto com as horas destinadas a um feriado. Esta semelhança deveu-se à necessidade de ambos serem processadas no Fiori no mesmo preenchimento da *timesheet* e com um espaço temporal quinzenal.

Na tabela ZHRA001_TB_CLDR colocou-se os feriados que existiam anualmente com o dia a ser preenchido em um dos 16 campos existentes que se referiam a possibilidade de existirem 16 dias numa quinzena e os campos adicionais como o ano, mês, designação do feriado, código e comentário do *engagment*, país onde se celebra este feriado e quinzena do mês, sendo que um mês tem sempre duas quinzenas, coloca-se um ou dois dependendo se a *timesheet* irá abranger do dia 1 até dia 15 ou se será de dia 16 a dia 30/31.

Já a tabela ZHRA001_TB_DATA seguia o mesmo procedimento relativo à determinação da quinzena a preencher, partilhando alguns campos mencionados anteriormente como o ano, mês, código, comentário do *engagment* e código do país, acrescentando campos como nome do utilizador, *ticket*, número do cliente e indicação de submissão.

A tabela ZHRA001_TB_CLNT tem como propósito guardar os clientes para os quais são destinadas as horas submetidas no relatório. Esta tabela conta com campos como o número do cliente, descrição, data de criação, se o mesmo se encontra bloqueado e a data do bloqueio.

Para se distinguir os *engagments* destinados aos gestores dos projetos dos *engagments* destinados aos membros das equipas, criou-se duas tabelas distintas.

A tabela ZHRA001_TB_GTET tem como intuito sustentar os *engagments* dos gestores e conta com campos como código do *engagment*, número do cliente, nome do utilizador, *status* e se o gestor está bloqueado.

Já na tabela ZHRA001_TB_ENGT, os campos são semelhantes à tabela mencionada anteriormente à exceção do bloqueio do gestor e conta com campos distintos em adição como data de criação do *engagment*, descrição do mesmo, se este se encontra bloqueado e a data do bloqueio.

Para mitigar o acesso a possíveis alterações de terceiros a projetos aos quais não se encontram alocados, criou-se uma tabela designada ZHRA001_TB_AREA, para identificar a que área cada gestor de equipa pertence e à qual o gestor pode efetuar alterações ou consultas. Esta tabela conta com os campos código da área e descrição.

A tabela ZHRA001_TB_CONFG é uma tabela destinada aos gestores e ao processo de bloquear áreas, caso as mesmas não se encontrem em desenvolvimento, contando assim com os campos nome de utilizador e bloqueado como um campo booleano.

## 6.3. CDS e OData

### 6.3.1. CDS

A utilização de visões CDS permite o acesso aos dados presentes nas tabelas e ao mesmo tempo permite a utilização de serviços OData para recolher esses dados do Fiori. Com isso criou-se uma visão referente a cada uma das tabelas criadas, à exceção da tabela das constantes, devido ao seu uso específico. Adicionou-se duas visões que não se enquadram com nenhuma das tabelas criadas, sendo elas a visão referente aos utilizadores, para conseguir tratar os dados dos mesmo sendo que cada utilizador se encontra munido de um conjunto de credenciais únicas, permitindo assim distinguir os níveis de autorização que viriam a ser atribuídos aos utilizadores consoante as suas funções.

Esta visão distingue-se das restantes visto que encontra a sua referência numa tabela *standard*, com campos previamente criados pela SAP, que permitem guardar e identificar os utilizadores criados. Esta visão encontra-se como a base para o controlo de todas os campos referentes ao utilizador, fazendo a verificação deste campo em tabela como ZHRA001_TB_CNFG, ZHRA001_TB_GTET e ZHRA001_TB_GTEQ.

A outra visão que se criou e que não tem a sua referência numa tabela personalizada é a ZHRA001_CV_GLBL. Esta visão esta diretamente relacionada com a tabela ZHRA001_TB_DATA e conecta-se a outras recorrendo a associações e consegue filtrar os dados pretendidos recorrendo a projeções. As visões que estão conectadas à global são ZHRA001_CV_USERS, ZHRA001_CV_ENGT, ZHRA001_CV_CLNT, ZHRA001_CV_GTEQ, ZHRA001_CV_AREA, ZHRA001_CV_GTET e ZHRA001_CV_CLDR. Esta visão, tal como o nome indica, o ponto global que trata de conectar os dados existentes nas diversas tabelas e de onde são filtrados grande parte dos dados.

A ZHRA001_CV_CLDR conecta-se diretamente a tabela correspondente, algo que já não acontece com a ZHRA001_CV_DATA, que além de aceder à respetiva tabela da data, também se conecta à tabela do calendário para efetuar as verificações necessárias.

A ZHRA001_CV_ENGT além de conectar à tabela dos *engagments*, também recorre à tabela dos clientes conseguindo assim efetuar a verificação entre o *engagment* utilizado e o utilizador ao qual esta alocado.

A ZHRA001_CV_GTET serve-se da tabela dos *engagments* dos gestores, com o mesmo nome, mas também se conecta com a tabela dos clientes no processo de seleção de dados para efetuar a filtragem de dados necessária recorrendo ao campo número de cliente existente em ambas as tabelas.

### 6.3.2. OData

Tal como fora abordado anteriormente, as CDS possuem componentes designado como *annotations*, que permitem despoletar uma funcionalidade chamada OData que permite a comunicação entre o *backend* e o modelo de dados da solução com a aplicação *web*, visto que os utilizadores recorrem à *web* para visualizar os dados e adicionar as novas entradas na *timesheet* [22].

Com a anotação @OData.publish:true, é possível criar um OData e assim exportar e criar o modelo de dados. Esta etapa permite importar este modelo de dados para o SAP *Gateway Service Builder* onde se apresenta o modelo de dados e se gera a classe com os métodos e as respetivas funções de criação(*create)*, atualização(*update)*, eliminação*(delete)* e consulta de dados(*get).*

Para cada uma destas funções é gerado um método referente a cada uma das tabelas e consequentemente a sua respetiva visão, possibilitando assim o tratamento dos dados de forma separada condizendo com a ação pretendida. Esta ação é despoletada no Fiori, onde se pode consultar e adicionar os dados que serão carregados nas respetivas tabelas.

O facto de todo o ambiente se interligar providencia uma maior facilidade de desenvolvimento excluindo ferramentas externas e com isso manter uma maior integridade, segurança e performance de todo o processo e consequentemente da ferramenta desenvolvida.

## 6.4. Classes e métodos

A solução desenvolvida em SAP ABAP foi estruturada recorrendo a duas classes e diversos métodos que visam tratar os dados e viabilizar a componente lógica da ferramenta.

Devido a modelação da solução recorrendo a programação orientada a objetos, ambas as classes se encontram interligadas e são essenciais para o correto funcionamento da aplicação.

### 6.4.1. Classe ZCL_ZHRA001_CV_GLBL_CD_DPC_EXT

Para alcançar o correto funcionamento da ferramenta desenvolvida, criou-se duas classes. Uma delas foi gerada automaticamente devido ao OData, designada ZCL_ZHRA001_CV_GLBL_CD_DPC_EXT, onde se encontram todos os processos relativos às ações efetuadas entre o Fiori e a CDS.

- *Handle Error*

Este método é utilizado para lidar com as exceções despoletadas no sistema, provenientes de falhas nas ações realizadas. Envia-se para o ecrã uma mensagem referente ao erro detetado, permitindo ao utilizador estar informado sobre a falha na ação e a razão pela qual a mesma foi acionada.

Utiliza-se o módulo de função *standard BAPI_MESSAGE_DETAIL* para atribuir a mensagem de erro pretendida [23]. O método *standard get_message_container* cria uma caixa com uma mensagem que contém a informação a mostrar ao utilizador [24].

- Area

Para realizar a ação de *Create* recorreu-se ao método *standard* read_entry_data onde se transformam os dados provenientes da API para uma tabela com a estrutura da tabela Area [25].

Posteriormente vai-se aceder ao método prc_create_area onde se vai inserir os dados na tabela. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Para a ação de *Delete*, tal como no método *create*, os dados são transformados e inseridos numa estrutura interna do tipo da tabela Area e enviados para o método prc_delete_area onde a entrada será eliminada da tabela. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Para o *Update*, o processo de leitura da entrada proveniente da API é igual aos métodos já mencionados, após este passo, executa-se uma chamada ao método prc_update_area importando esta estrutura interna e atualizando os dados na tabela Area. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

- Client

Para a ação *Create*, recorre-se ao método *standard* read_entry_data onde se transformam os dados provenientes da API para uma tabela com a estrutura da tabela *Client*.

Posteriormente vai-se aceder ao método prc_create_client onde se vai inserir os dados na tabela. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Para o *Delete*, tal como no método *create*, os dados são transformando e inseridos numa estrutura interna do tipo da tabela *Client* e enviados para o método prc_blk_client onde a entrada será bloqueada recorrendo ao preenchimento do campo bloqueado e ao campo referente à data do bloqueio.

Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Para se realizar o *Update,* o processo de leitura da entrada proveniente da API é igual aos métodos já mencionados, após este passo, executa-se uma chamada ao método prc_update_client importando esta estrutura interna e atualizando os dados na tabela *Client*. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

- Configurator

Para a ação *Create* recorre-se ao método *standard* read_entry_data onde se transformam os dados provenientes da API para uma tabela com a estrutura da tabela *Config*.

Posteriormente vai-se aceder ao método prc_create_config onde se vai inserir os dados na tabela. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Relativamente à ação de *Update*, o processo de leitura da entrada proveniente da API é igual aos métodos já mencionados, após este passo, executa-se uma chamada ao método prc_update_config importando esta estrutura interna. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

- Engagment Manager

Para realizar a ação relativa ao *Create*, recorreu-se ao método *standard* read_entry_data onde se transformam os dados provenientes da API para uma tabela com a estrutura da tabela *gtet*.

Posteriormente vai-se aceder ao método prc_create_gtet onde se vai inserir os dados na tabela. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Para o *Delete* realizou-se um mapeamento manual dos dados provenientes da API referentes aos campos cliente, *engagment* e utilizador para uma estrutura com estes mesmos campos. Posteriormente importou-se esta mesma estrutura para o método prc_delete_gtet, onde os dados são devidamente tratados.

Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Para se realizar a ação *Update*, recorre-se ao método *standard* read_entry_data onde se transformam os dados provenientes da API para uma estrutura interna com a estrutura da tabela *gtet*.

Posteriormente vai-se aceder ao método prc_update_gtet para onde se vai exportar os dados na estrutura. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

- Engagment

Para se proceder à recolha de dados da base de dados e enviar os mesmos para o utilizador visualizar, executou-se a ação *Get* no método *standard* get_entityset, para se recolher a entrada requerida e comunicar os dados através do OData, para estes serem posteriormente colocados à disposição do utilizador. Esta ação não requer qualquer personalização da lógica funcional pois a mesma já se encontra totalmente funcional por defeito.

Para a ação *Create* recorre-se ao método *standard* read_entry_data onde se transformam os dados provenientes da API para uma tabela com a estrutura da tabela *Engagment*.

Posteriormente vai-se aceder ao método prc_create_engagment onde se vai inserir os dados na tabela. Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Após acionar a ação *Delete*, tal como no método *create*, os dados são transformados e inseridos numa estrutura interna do tipo da tabela *engagment* e enviados para o método prc_delete_engagment onde a entrada será eliminada da tabela dos *engagments*.

Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Para a ação *Update*, o processo de leitura da entrada proveniente da API é igual aos métodos já mencionados, após este passo, executa-se uma chamada ao método prc_update_engagment importando esta estrutura interna e atualizando os dados na tabela *engagment*.

Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

- Manager

Na ação *Create*, para conseguir interpretar os dados e posteriormente mapear os mesmos para a sua tabela destino, recorreu-se ao método *standard* read_entry_data, que

permite transformar os dados recebidos através da API igualando os mesmo a uma tabela com a estrutura pretendida, permitindo um preenchimento e uma leitura correta.

Neste método recorreu-se ao método prc_create_manager, importando com ele os dados necessários à criação de uma nova entrada na tabela ZHRA001_TB_GTEQ, utilizando como filtros a área e o utilizador, para evitar uma duplicação de dados.

Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

- Global

Na ação *Get*, é executada uma validação para averiguar a existência de um feriado e para tal recorrendo-se ao método chk_holiday.

Previamente utiliza-se o método get_entityset para se recolher os dados pretendidos.

Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

No método responsável por realizar a ação *Create*, a entrada é adicionada diretamente na *visão* global. Recorre-se ao método *standard* read_entry_data novamente para se obter uma correta leitura e inserção dos dados obtidos numa tabela com a estrutura correta e depois realiza-se a adição destes dados na tabela.

Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

Para realizar a ação de *Delete*, adiciona-se a uma estrutura local os campos do utilizador, quinzena, mês e ano e com esta estrutura local realiza-se uma eliminação dos dados presentes na tabela ZHRA001_TB_DATA que sejam condizentes com a entrada presente na estrutura local com o auxílio do comando delete from ITAB where FILTRO.

Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

- *Execute Action*

Neste método realiza-se a criação e a atualização da tabela data. Primeiramente identifica-se o *entityset* que se encarrega de despoletar uma ação e caso este seja *timesheetCreate* analisa-se os parâmetros provenientes da API para se separar os dados referentes ao ano, ao mês e a quinzena. Para se tratar os dados referentes à *payload* presente nesta entrada, utiliza-se o método *standard deserialize*, capaz de ler a estrutura enviado em formato JSON e mapear os dados presentes no JSON da *payload* para uma tabela interna do tipo da *visão* global.

Para evitar uma leitura incorreta dos dados devido a possíveis diferenças nos formatos dos mesmos provenientes do JSON e dos elementos de dados presentes na *visão* global, os mesmos passam por um ciclo de conversão de formato, forçando-os a ter exatamente o mesmo formato dos elementos de dados aos quais estão a ser mapeados. Esta etapa previne que se verifiquem *dumps* que poderiam acontecer devido a discrepâncias entre os formatos dos elementos de dados, compreenda-se como formatos dos elementos de dados o tipo de elemento de dados e o comprimento de cada campo da entrada analisada.

Posteriormente exporta-se a tabela interna para o método create_update_data, onde os dados são tratados e adicionados às respetivas tabelas seguindo a lógica descrita posteriormente.

Por último criou-se uma exceção capaz de lidar com os possíveis erros evitando assim um *dump* do sistema e envia-se uma mensagem para o ecrã notificando o utilizador que a operação não foi concluída com sucesso.

### 6.4.2. Classe ZCL_ZHRA001_TIMESHEET

A outra tabela, designada ZCL_ZHRA001_TIMESHEET foi criada para tratar os dados individualmente, recorrendo a uma programação orientada a objetos para tornar o código mais eficiente e limpo.

De forma a tratar os dados e suprimir os pedidos efetuados pelo utilizador durante a utilização da ferramenta, desenvolveram-se os métodos presente na Figura 10.

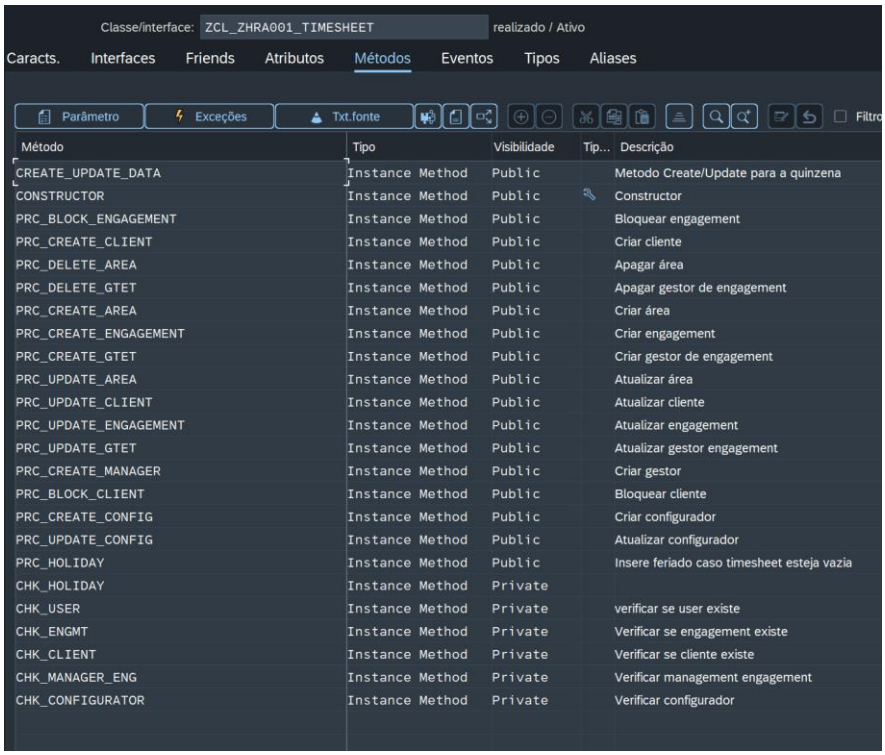


Figura 10 - Métodos presentes na classe ZCL_ZHRA001_TIMESHEET.

- Método PRC_CREATE_(indicativo final da tabela)

Para obter uma solução linear e de fácil interpretação, a lógica presente em todos os métodos com a nomenclatura indicada, tem como objetivo a criação de uma nova entrada numa das tabelas existentes na base de dados e funcionam de uma forma similar.

Em todos estes métodos, recolhe-se os valores importados através da classe do OData devido à comunicação do *frontend* e realiza-se uma validação à tabela para percecionar se a entrada já existe. Caso isso aconteça é acionada uma exceção com uma mensagem de erro, caso a entrada seja efetivamente nova, então a mesma é adicionada na tabela.

- Método PRC_UPDATE_(indicativo final da tabela)

Para a implementação da função de atualização de entradas, segue-se a mesma solução linear com a lógica a desenvolver nos métodos que seguem esta nomenclatura.

Em todos estes métodos, importa-se novamente os dados obtidos através do OData, que são provenientes da ferramenta Fiori, e realiza-se uma operação de *Update* na tabela em questão recorrendo ao comando *Update dbtab from structure*.

- Método PRC_BLK_(indicativo final da tabela)

Os métodos que se destinam a efetuar o bloqueio de entradas, são em tudo similares aos métodos de atualização de dados nas tabelas descritos anteriormente. Quando se efetua um bloqueio, preenche-se o campo destinado com a indicação de bloqueio, campo que está presente na estrutura base e os dados são provenientes do Fiori através da classe do OData. Previamente realiza-se o *update* à tabela presente na base de dados.

- Método PRC_DELETE_(indicativo final da tabela)

Os métodos que visam eliminar dados de uma tabela, importam estes mesmos dados da entrada que vão ser eliminados através do OData e do método específico para cada tabela. Coloca-se esses dados numa estrutura semelhante à da tabela pretendida, realiza o comando *delete* da tabela pretendido recorrendo a um filtro *where* onde o campo chave da tabela presente na base de dados e tem de ter o mesmo valor que o campo chave presente na estrutura preenchida durante a execução desta ação proveniente do Fiori.

- Método HOLIDAY

Realiza-se uma busca à tabela do calendário recorrendo à quinzena, ao mês e ao ano fornecidos. Posteriormente é feito um mapeamento manual campo a campo para introduzir o valor da entrada apenas no campo pretendido. Isto acontece devido a particularidade existente tanto nesta tabela como na tabela da data, em que cada dia da quinzena tem um campo na tabela. Após findar este mapeamento, as entradas são adicionadas na tabela da data, com o *engagment* definido como feriado para indicar que aquele dia já se encontra preenchido com oito horas.

- Método CHK_HOLIDAY

A par do método *holiday*, é feita a verificação à tabela do calendário recorrendo à quinzena, ao mês e ao ano fornecidos. Caso a tabela retorne alguma entrada, então é feita uma alteração recorrendo a um ciclo em que se elimina a entrada da tabela data se o feriado não existir.

- Método CHK_(indicação final da tabela)

Durante os métodos que executam verificações a uma tabela, filtram-se pela chave primária importada e pelo campo de bloqueado. Caso o mesmo exista, retorna-se o valor das entradas existentes na tabela relativamente a esse filtro.

Se o valor de entradas encontradas for igual a zero, uma *flag* é colocada como vazia, caso existam dados a *flag* é preenchida.

- Método CREATE_UPDATE_DATA

Após filtrar os dados na tabela da data onde realiza esta busca por utilizador, quinzena, ano e mês, executa-se uma verificação sobre o *engagment* para analisar se o mesmo existe e se se encontra válido. Esta verificação é feita recorrendo ao método chk_engmt. Em seguida recorre-se ao método chk_holiday para verificar se o dia não se trata de um feriado. Caso estas verificações retornem um resultado positivo, então esta entrada proveniente do Fiori é inserida na tabela.

- Método CONSTRUCTOR

Este método destina-se a guardar o utilizador que acedeu ao sistema numa variável global.

# Capítulo 7 Resultado do desenvolvimento

## 7.1. Funcionalidades da ferramenta

### 7.1.1. Página Inicial

Quando um utilizador entra na ferramenta, depara-se com o menu de login onde vai colocar as suas credenciais de acesso condizentes com as que lhe foram atribuídas para entrar na ferramenta SAP GUI [26]. Esta etapa é extremamente importante pois o utilizador inserido será determinante para se identificar o grau de autorizações que o utilizador tem e as ferramentas que tem ao seu dispor.

Após entrar na ferramenta, o utilizador depara-se com a aba da ferramenta, presente na Figura 11.

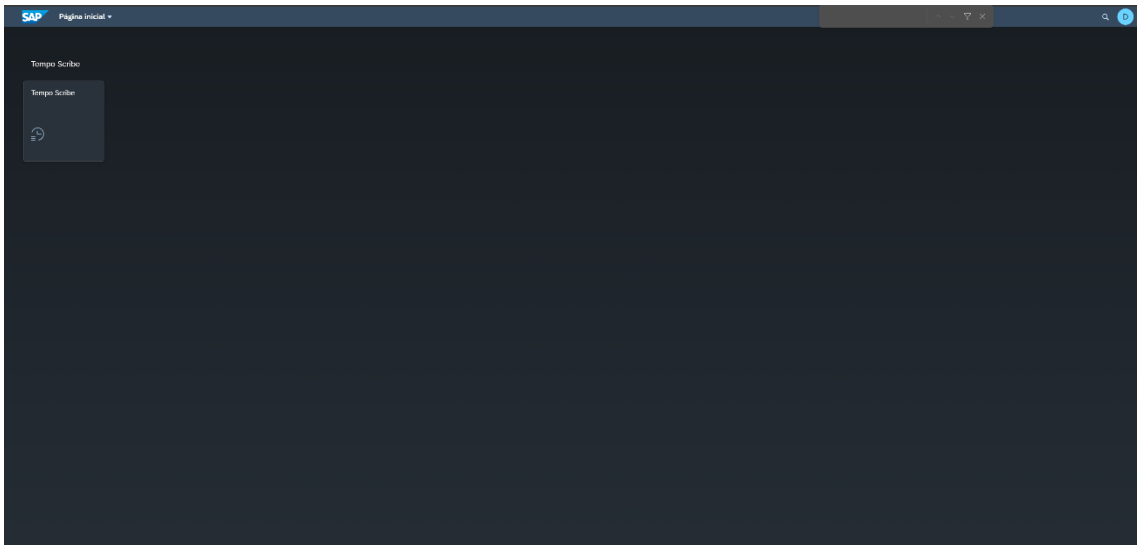


Figura 11- Página inicial

### 7.1.2. Página inicial da *timesheet*

Após entrar na ferramenta, é possível visualizar os separadores da *timesheet*. Na Figura 12 é possível visualizar os três separadores possíveis. Um utilizador de nível baixo apenas visualiza o separador relativo ao registo de horas. Caso o utilizador seja um gestor, tem acesso ao registo de horas e ao relatório das horas. Apenas um utilizador com autorização de configurador tem acesso a todos os separadores e capacidades da ferramenta.

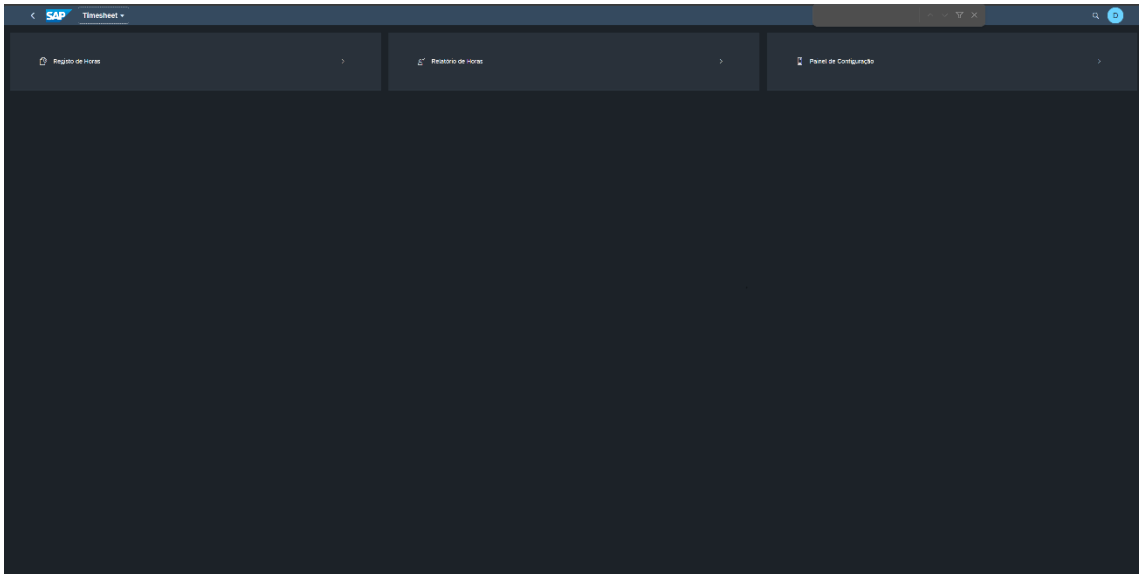


Figura 12- Página inicial *timesheet*

### 7.1.3. Registo de horas na *timesheet*

Na Figura 13 é possível visualizar uma *timesheet* completa, onde se registam as horas separadas por clientes e respetivos *engagments*, as horas que foram dedicadas a cada *engagments* e na parte inferior da imagem está presente um resumo do registo efetuado de horas inseridas em cada cliente.

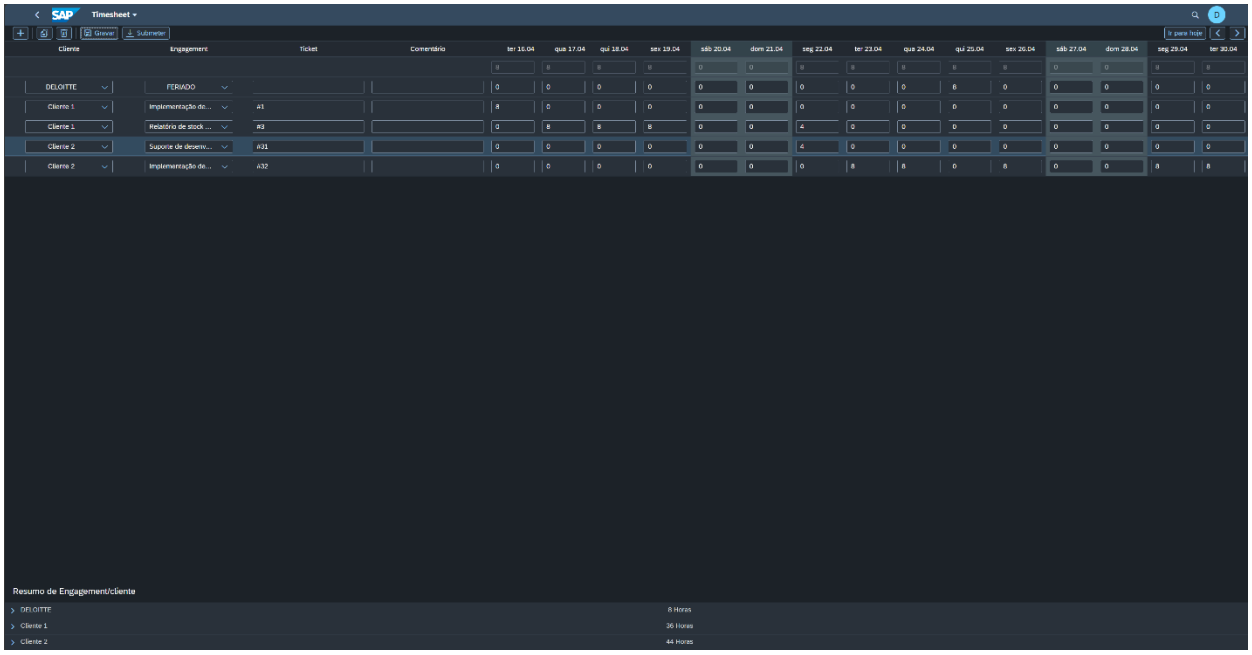


Figura 13- Registo de horas

No caso de se exceder o limite de oito horas diárias, o sistema emite o erro visível na Figura 14.

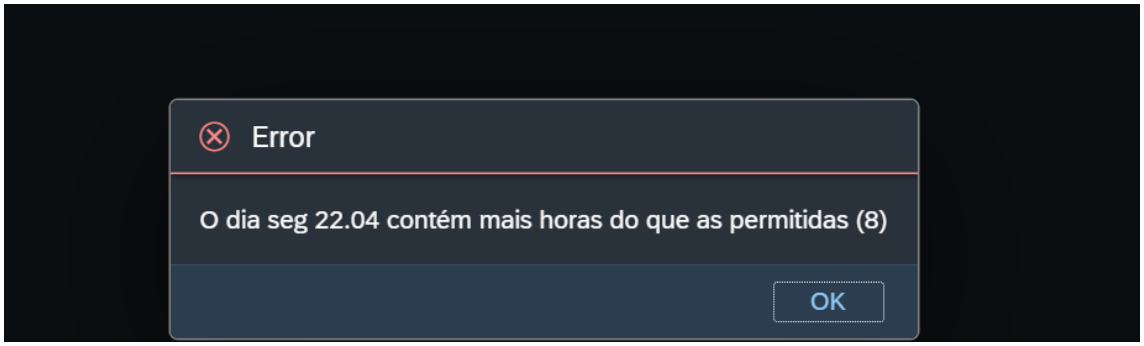


Figura 14- Limite de horas excedido.

Caso o utilizador registe horas fora dos dias úteis, o erro presente na Figura 15 é enviado para o ecrã.

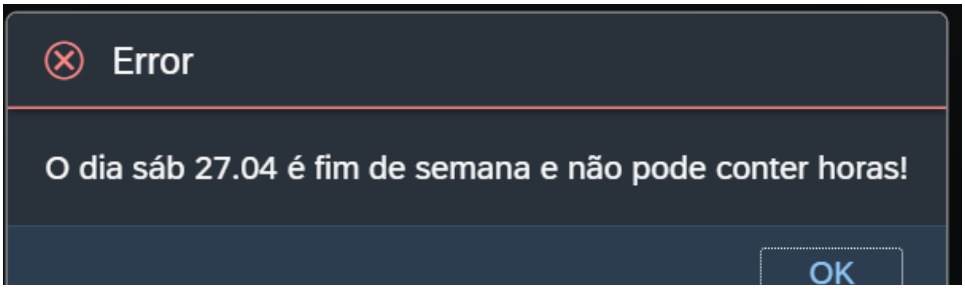


Figura 15 - Registo de horas no fim de semana.

Quando se encontra um feriado, o registo efetuado na ferramenta segue o que é apresentado na Figura 16.

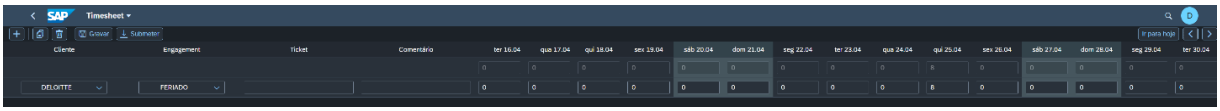


Figura 16 - Registo de um feriado.

### *7.1.4. Engagment*

Para criar um *engagment*, como é possível verificar na Figura 17, é necessário preencher o cliente para o qual o *engagment* será criado, o identificador do *engagment* e por fim o nome do *engagment* que servirá também de descrição. Apenas um utilizador com autorizações de configurador pode criar *engagments*.

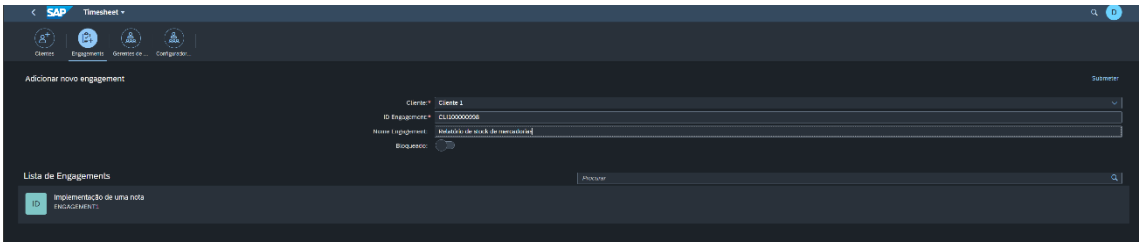


Figura 17- Criar um *engagment.*

Devido à possibilidade de atribuir diversos *engagments* a um cliente, é possível procurar por um *engagment* específico de um cliente específico, preenche-se o cliente pretendido e o *engagment* que se procura, como se verifica na Figura 18. Caso se pretenda verificar todos os *engagments* atribuídos a um cliente, basta preencher o cliente, como se verifica na Figura 19.

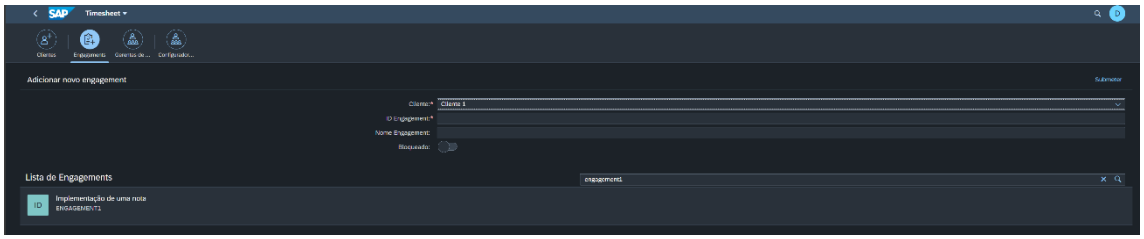


Figura 18- Visualizar *engagment* relativo ao cliente 1.

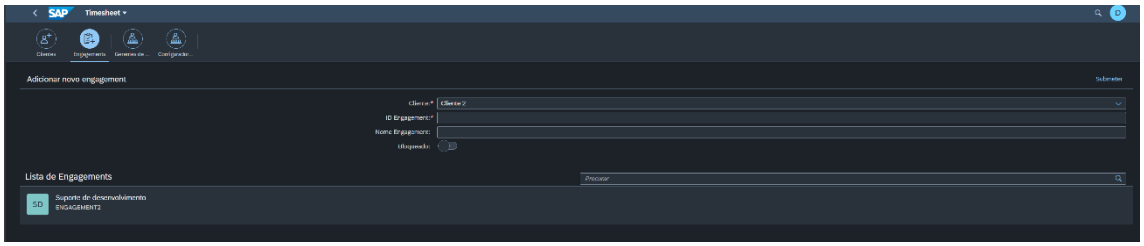


Figura 19- Visualizar *engagment* relativo ao cliente 2.

### *7.1.5. Engagment* para *manager*

Para se associar um *engagment* a um *manager* é necessário preencher o identificador do *engagment* e o nome de utilizador atribuído ao *manager*, como se pode verificar na Figura 20. Caso se pretenda visualizar o manager a que um *engagment* está atribuído, basta colocar o identificador do *engagment* na barra de pesquisa e o sistema vai mostrar o utilizador associado a esse mesmo *engagment*, como se pode verificar tanto na Figura 20 como na Figura 21.

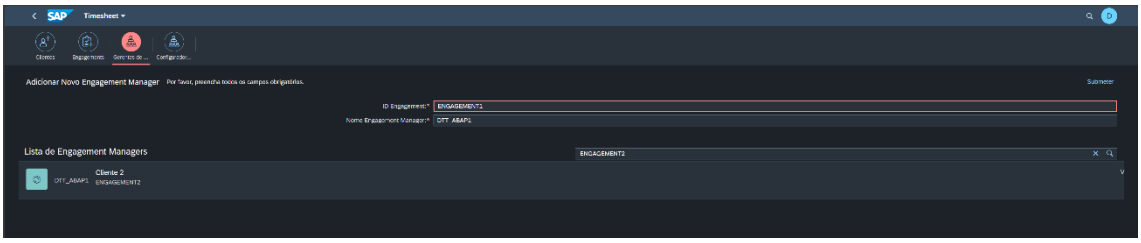


Figura 20- Associar um *engagment* a um *manager*.

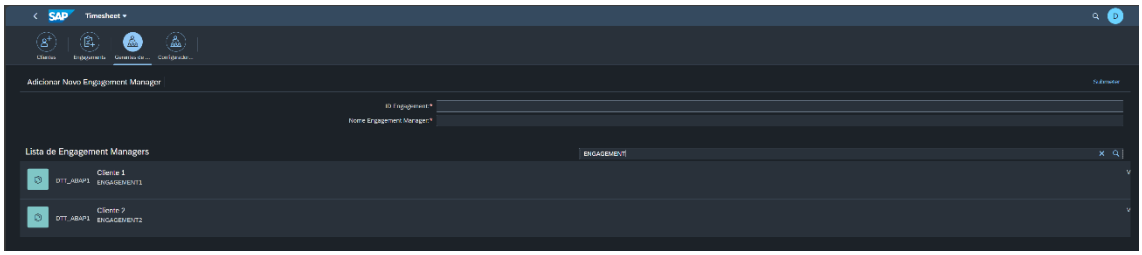


Figura 21 - *Engagments* atribuídos a um *manager*.

### 7.1.6. Clientes

Para adicionar um cliente, é necessário preencher o identificador e o nome do cliente, sendo estes campos obrigatórios, como se pode verificar na Figura 22. Esta funcionalidade apenas está disponível para o utilizador configurador.

Já para visualizar um cliente presente no sistema, coloca-se o nome do mesmo na barra de pesquisas. Caso se pretenda uma lista de todos os clientes existentes no sistema, basta efetuar a pesquisa com a barra vazia.

Para se bloquear um cliente é necessário fornecer o identificador do cliente, o nome do cliente e acionar o botão de bloqueado, como ilustra a Figura 22.

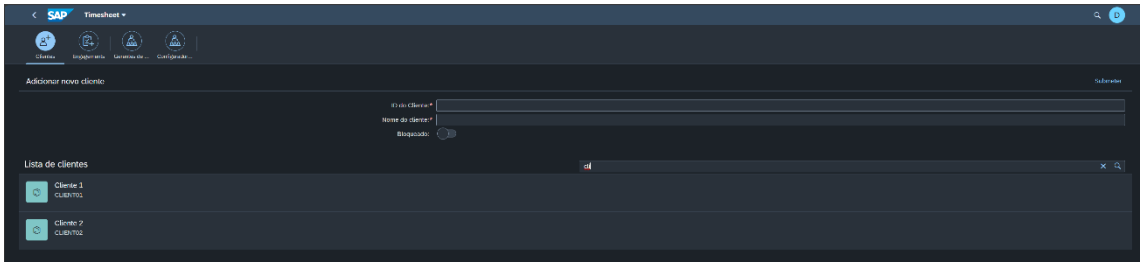


Figura 22 - Adicionar e visualizar clientes.

### 7.1.7. Configurador

Para se atribuir uma autorização a um utilizador para que este se torne configurador é necessário preencher o nome do utilizador na barra nome de configurador, de caráter obrigatório, e submeter. Também é possível bloquear um configurador,

acionando o botão de bloqueio acompanhado pelo nome do configurador. Ambas as funcionalidades estão ilustradas na Figura 23.

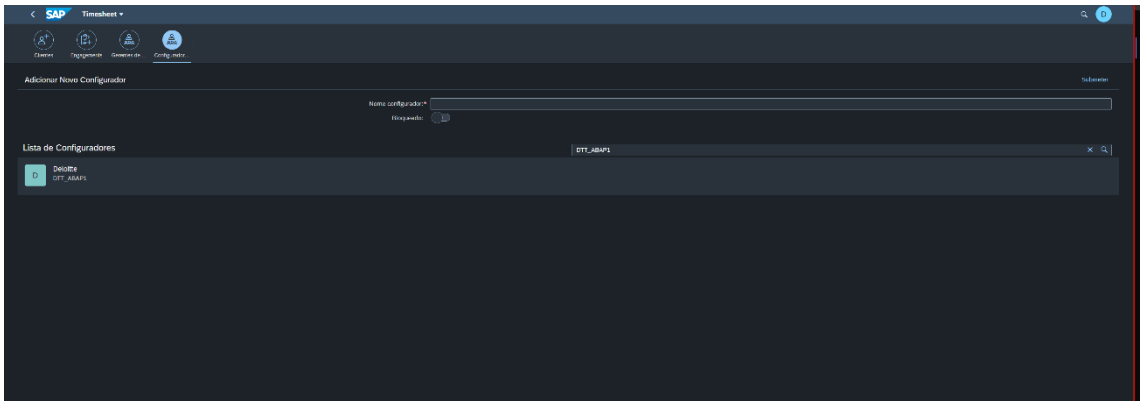


Figura 23- Adicionar um configurador.

# Capítulo 8 Conclusões

## 8.1. Síntese

A tese apresentada tem como objetivo o desenvolvimento do *backend* para uma ferramenta de registo de horas, oferecendo aos utilizadores uma plataforma capaz, estruturada, de fácil uso e compreensão. Os objetivos foram atingidos na sua globalidade.

Foi possível desenvolver uma ferramenta onde se registam as horas tendo em conta o cliente e o *engagment* que se realizou, as horas investidas e o dia em que o mesmo foi efetuado. Funcionalidades como a visualização e controlo do registo de outro colaborador por parte de um líder de equipa e a criação de novos clientes ou *engagments* foi um desenvolvido com sucesso.

A integração do *backend* com a aplicação *web*, aliada com a recolha e o tratamento de dados provenientes dos utilizadores após inserir os mesmo no Fiori *web app* e a comunicação entre o *backend* e *frontend* foram objetivos alcançados que permitiram obter o produto final indicado anteriormente.

Apesar do sucesso alcançado, existiram algumas funcionalidades que não foram completamente implementadas e impediram um sucesso total do trabalho proposto. A funcionalidade que almejava colocar automaticamente o feriado independente do registo do utilizador não foi concluída, permitindo apenas colocar um feriado manualmente como se um *engagment* comum se tratasse.

Apesar dos diversos percalços existentes durante o processo de desenvolvimento, como a estruturação de tabelas, a comunicação e uniformização dos campos para o tratamento de dados entre o *backend* e *frontend* foram algumas das dificuldades encontradas ao longo do processo. Este projeto foi importante para a aquisição de novos

conhecimentos, nomeadamente relacionados com tópicos como OData e visões CDS, dois *hot topics* no mundo SAP devido às suas extraordinárias capacidade e polivalências.

Em conclusão, a ferramenta foi desenvolvida com sucesso, os requisitos estabelecidos foram executados e o produto final é capaz de satisfazer o seu propósito e no futuro continuar a evoluir para se tornar uma ferramenta mais abrangente e preparada para os desafios do mercado.

## 8.2. Trabalho futuro

De modo a evoluir a ferramenta e adicionar funcionalidades uteis, a conexão e adaptação da ferramenta para trabalhar diretamente com a plataforma JIRA é um dos pontos fundamentais para o sucesso da mesma a curto prazo, tornando-a assim numa ferramenta mais útil conectando se com uma plataforma mundialmente usada para a gestão de projeto [27].

Conseguir preencher as ausências na plataforma seria um *must to have* e iria acrescentar imenso valor à ferramenta. Desenvolver uma ferramenta que funcione como um lembrete para um utilizador via e-mail quando este ainda não realizou o preenchimento das horas para a quinzena em que se encontra seria uma característica interessante que iria auxiliar o utilizador.

# Bibliografia


[1] Cake.com, “Timesheet,” Clockify, [Online]. Available: https://clockify.me/features/timesheet. [Acedido em Setembro 2024].

[2] Factorial, “Sobre a Factorial,” Factorial, [Online]. Available: https://factorialhr.pt/sobre-a-factorial. [Acedido em Setembro 2024].

[3] Clocify, “Timesheet,” [Online]. Available: https://clockify.me/help/track-time-and-expenses/timesheet-view. [Acedido em Outubro 2024].

[4] Factorial, “Simplify Time Tracking with Factorial's Software,” [Online]. Available: https://factorialhr.co.uk/time-attendance-signing. [Acedido em Outubro 2024].

[5] Clockify, “Plans & Pricing,” [Online]. Available: https://clockify.me/pricing. [Acedido em Outubro 2024].

[6] Factorial, “Preço,” [Online]. Available: https://factorialhr.pt/preco. [Acedido em Outubro 2024].

[7] Microsoft, “Encontre o melhor plano do Microsoft 365 para a sua empresa,” [Online]. Available: https://www.microsoft.com/pt-pt/microsoft-365/business/compare-all-microsoft-365-business-products?market=pt. [Acedido em Outubro 2024].

[8] SAP, “O que é a SAP?,” SAP, [Online]. Available: https://sap.com/portugal/about/what-is-sap.html. [Acedido em Setembro 2024].

[9] SAP, “O que é o ERP da SAP?,” SAP. [Online]. [Acedido em Setembro 2024].

[10] Hub_AI Technologies Pvt. Ltd, “Unveiling the Diversity of SAP Modules,” 15 12 2023. [Online]. Available: https://www.linkedin.com/pulse/unveiling-diversity-sap-modules-hubai-technology-private-limited-kdk7c. [Acedido em Outubro 2024].

[11] I. e. Rasheed, “SAP Easy Access,” 9 Junho 2019. [Online]. Available: https://erproof.com/sap-easy-access/. [Acedido em Outubro 2024].

[12] Rheinwerk Publishing, “ABAP Programming for SAP,” [Online]. Available: https://learning.sap-press.com/abap. [Acedido em Outubro 2024].

[13] Simple VIsion, “SAP Fiori: saiba o que é,” 20 10 2021. [Online]. Available: https://simplevisionit.com/sap-fiori/. [Acedido em Outubro 2024].

[14] Eclipse Foundation, “What Is Eclipse?,” [Online]. Available: https://www.eclipse.org/home/whatis/. [Acedido em Outubro 2024].

[15] SAP, “ABAP Development Tools - Eclipse,” [Online]. Available: https://help.sap.com/doc/saphelp_nw75/7.5.5/en-US/a3/314a7fd9384ce8a40eff2d3b144628/content.htm?no_cache=true. [Acedido em Outubro 2024].

[1 “ABAP Core Data Services - Introduction (ABAP CDS view),” 09 Setembro 2017.
6] [Online]. Available: https://community.sap.com/t5/technology-blogs-by-members/abap-core-data-services-introduction-abap-cds-view/ba-p/13320992. [Acedido em Outubro 2024].

[1 Visual Paradigm, “Diagramas,” [Online]. Available: https://online.visual-
7] paradigm.com/pt/diagrams/. [Acedido em Outubro 2024].

[1 SAP, “ABAP Packages,” [Online]. Available:
8] https://help.sap.com/docs/ABAP_PLATFORM_NEW/c238d694b825421f940829321ffa326a/4ec14bab6e391014adc9fffe4e204223.html. [Acedido em Outubro 2024].

[1 SAP, “ABAP Dictionary,” [Online]. [Acedido em Outubro 2024].
9]

[2 SAP, “Class Builder,” [Online]. Available:
0] https://help.sap.com/doc/saphelp_nw73ehp1/7.31.19/en-US/ca/c035baa6c611d1b4790000e8a52bed/content.htm?no_cache=true. [Acedido em Outubro 2024].

[2 SAP, “Function Builder,” [Online]. Available:
1] https://help.sap.com/doc/saphelp_nw73ehp1/7.31.19/en-US/d1/801e9a454211d189710000e8322d00/content.htm?no_cache=true. [Acedido em Outubro 2024].

[2 R. Gupta, “How to create ABAP CDS view and OData with SAP Annotations,” 13
2] Outubro 2016. [Online]. Available: https://community.sap.com/t5/technology-blogs-by-sap/how-to-create-abap-cds-view-and-odata-with-sap-annotations/ba-p/13303847. [Acedido em Fevereiro 2024].

[2 Desconhecido, “SAP BAPI_MESSAGE_GETDETAIL Function Module for Read
3] long text of error message,” [Online]. Available: https://www.se80.co.uk/sap-function-modules/?name=bapi_message_getdetail. [Acedido em Outubro 2024].

[2 S. C. -. former_member716571, “Error Handling In Odata,” 27 Julho 2021.
4] [Online]. Available: https://community.sap.com/t5/application-development-blog-posts/error-handling-in-odata/ba-p/13505968. [Acedido em Outubro 2024].

[2 kallolathome, “Easy way to read data via a deep entity in OData,” 23 Fevereiro
5] 2022. [Online]. Available: https://community.sap.com/t5/technology-blogs-by-members/easy-way-to-read-data-via-a-deep-entity-in-odata/ba-p/13539428. [Acedido em Outubro 2024].

[2 SAP, “SAP GUI,” [Online]. Available:
6] https://help.sap.com/doc/saphelp_nw74/7.4.16/en-US/9a/d405e746ef43288755cb80a14be542/content.htm?no_cache=true. [Acedido em Outubro 2024].

[2 JIRA, “Great outcomes,” [Online]. Available:
7] https://www.atlassian.com/software/jira. [Acedido em Outubro 2024].

[28] SAP, "ODATA," SAP, [Online]. Available: https://help.sap.com/docs/HANA_SMART_DATA_INTEGRATION/7952ef28a6914997abc01745fef1b607/d9f0a3b09e0f4b3eb010be8bd36871e5.html. [Acedido em Setembro 2024].

[29] SAP, "Introduction to ABAP Core Data Services (CDS)," SAP, 01 2022. [Online]. Available: https://www.sap.com/documents/2022/01/96489f20-157e-0010-bca6-c68f7e60039b.html. [Acedido em Setembro 2024].