Mestrado em Engenharia Informática

Estágio

Relatório Final

Operations Dashboards

Estagiário

João Martins

jcmart@student.dei.uc.pt

Orientador do DEI

Professor Doutor Carlos Fonseca

cmfonsec@dei.uc.pt

Orientador da ISA

Engenheira Ana Guimarães

aguimaraes@isa.pt

1 de julho de 2014

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Agradecimentos

Aos meus pais, avós e irmão, pela motivação, pelo apoio e pelos esforços que realizaram para que eu pudesse chegar ao fim deste percurso.

À Rafaela, minha namorada e minha melhor amiga, por todo o amor e por toda a força dada ao longo destes anos. Por todas as horas despendidas a rever este documento, pela ajuda emocional que me transmitiu e que me permitiu ultrapassar tantos obstáculos nesta jornada.

Ao Doutor Carlos Fonseca, pelos conselhos e acompanhamento prestados ao longo deste ano letivo.

Aos orientadores da empresa, engenheira Ana Guimarães e engenheiro António Damasceno, por todo apoio e conhecimentos transmitidos.

À empresa ISA, por me oferecer uma excelente primeira experiência de trabalho e por me dar a oportunidade de contactar com um conjunto de pessoas que contribuíram imenso para a expansão dos meus conhecimentos técnicos.

Aos meus amigos e colegas de faculdade com quem partilhei todo este caminho desde a licenciatura até à conclusão deste grau académico.

Aos meus colegas estagiários da ISA, pelos saberes trocados e acima de tudo pela amizade criada.

Resumo

A quantidade de informação que uma empresa necessita de manter aumenta substancialmente a cada dia de atividade. Como tal, a capacidade de organizar e analisar esses dados para extrair indicadores de negócio fiáveis é cada vez mais uma necessidade de maior para os seus colaboradores. É neste paradigma que se torna imprescindível o uso de ferramentas com capacidade para aglomerar e disponibilizar de forma intuitiva indicadores importantes retirados dessa informação. Para este conjunto de necessidades surgem aplicações que permitem a criação e gestão de *dashboards* interativos, onde os utilizadores podem definir o tipo de *KPI*s a extrair de uma quantidade massiva de informação emergente de várias fontes de dados – ficheiros Microsoft Excel, ERPs, bases de dados, entre outros.

O presente documento apresenta o trabalho científico realizado no âmbito do estágio "Operations Dashboards" inserido no Mestrado de Engenharia Informática. Este estágio teve como objetivo o desenvolvimento de uma plataforma integrada na *Intranet* da ISA que permite visualizar e gerir um conjunto de *dashboards* com informação intuitiva e sintetizada relativamente ao departamento de operações da empresa.

Palavras-Chave

Business Intelligence, Dashboards, Monitorização de projetos, Monitorização de recursos, Monitorização de despesas, Aplicação Web, Relatórios.

Índice

Lista de Figuras

Lista de Tabelas

Glossário

Termo	Descrição
Back-end	Conjunto de componentes, aplicações, serviços e servidores responsáveis pelo desempenhar da camada de lógica e por fazer o *deployment*.
Business Intelligence	Conjunto de teorias, metodologias e tecnologias que permitem transformar enormes conjuntos de dados em informação com conteúdo significativo.
Commit	Corresponde ao envio/registo para o repositório das alterações em ficheiros ocorridas entre o último *commit* e o estado atual do repositório. É possível reverter o projeto para o estado em que se encontrava em determinada *revision* (normalmente corresponde ao número de *commits* realizados até essa *revision)*.
Daily Digest	Mecanismos com o propósito de produzir diariamente pequenos relatórios com informação resumida.
Daily Meeting	Reuniões diárias com curto período de tempo para ser discutido entre a equipa de um projeto o trabalho realizado no dia anterior.
Dashboard	Painel constituinte de uma aplicação que permite apresentar gráficos, tabelas e *widgets* com indicadores de performance.
Data Source	Termo utilizado para especificar fornecedores de dados digitais tais como bases de dados, ficheiros, APIs, etc.
Data Warehouse	Sistema de armazenamento de dados que facilita a produção de relatórios, a análise de grandes volumes de dados e a obtenção de informações estratégicas que podem facilitar a tomada de decisão.
Deployment	Conjunto de atividades que permitem disponibilizar uma versão final ou não de uma aplicação com vista a ser testada ou usada.
Desktop Application	Aplicação nativa que corre no sistema operativo.
Drill Down	Ato de interagir com um componente para obter informação mais detalhada sobre determinado aspeto.
Event Driven	Funcionalidades que determinada aplicação possui que são acionadas por eventos – botões, temporizadores, etc.
Faceted Search	Técnica usada para filtrar dados ao especificar determinadas categorias sobre os resultados obtidos.
Front-end	Conjunto de componentes, aplicações, serviços e servidores responsáveis por disponibilizar a interface gráfica do sistema e com a qual o utilizador final interage.
Interop	Ficheiro que permite a comunicação/acesso a métodos, componentes ou objetos de um sistema em produção a partir de outro projeto. Funciona essencialmente como uma biblioteca que pode ser importada para um projeto em desenvolvimento, com vista a interagir com outro sistema.
Intranet	Rede de computadores privada que apenas pode ser acedida pelos clientes internos a essa rede.
Issue	Corresponde a um registo no JIRA de uma tarefa/*bug*/*backlog item*.
LDAP	Lightweight Directory Access Protocol, ou LDAP, é um protocolo que funciona sobre TCP/IP que permite manter um repositório de registos numa rede, nomeadamente contas de utilizador.
Mockup	Protótipo não utilizável da interface gráfica de um projeto de *software*.
Responsive Layout	É uma abordagem que permite o *design* de páginas web adaptáveis ao tamanho do ecrã dos dispositivos.
Sprint	Intervalos de tempo para desenvolver um conjunto de funcionalidades de um projeto. Os intervalos de tempo são normalmente de uma semana até um mês.
Stakeholder	Termo usado para identificar os intervenientes num determinado projeto.
Thin Client	Na área de *software* define-se como uma aplicação que necessita de poucos ou nenhuns recursos para disponibilizarem rapidamente as suas funcionalidades.
Trial	Termo usado para se referir a distribuições de *software* que possuem um tempo limite para teste, podendo apresentar limitações em algumas funcionalidades.
Web Application	Aplicação que executa num *web browser* utilizando tecnologias como HTML, JavaScript, JSON, AJAX, etc.

Web Service	Sistema utilizado para a integração e troca de dados entre aplicações, podendo estes serviços terem sido desenvolvidos em diferentes linguagens de programação.
Web Part	Frequentemente chamados de *web widgets* tratam-se de zonas de uma página *web* de serviços Microsoft Sharepoint associadas a determinado conjunto de funcionalidades desenvolvidas em ASP.NET.
Widget	Componente visual de uma interface que permite apresentar informação ou até mesmo interagir com o utilizador a partir de botões, filtros, etc.
Work Log	Corresponde ao registo da quantidade de tempo que se despendeu em determinado *issue*. Um *issue* pode receber vários *work logs* até ser dado como concluído.

Acrónimos

Acrónimo	**Termo**
API	Application Programming Interface
BI	Business Intelligence
DOM	Document Object Model
IIS	Internet Information Services
KPI	Key Performance Indicator
SPA	Single Page Application

Lista de Anexos

	Documento
Anexo A	Documento de Requisitos de Alto Nível
Anexo B	Documento de Requisitos Detalhados
Anexo C	Documento de Arquitetura
Anexo D	Planeamento do Projeto – Diagrama de Gantt
Anexo E	Plano de Testes

Capítulo 1
Introdução

O capítulo inicial pretende dar a conhecer ao leitor o contexto do estágio "*Aplicação web e móvel para apoio e decisão na área das operações"*. Este estágio encontra-se inserido no âmbito da disciplina de estágio do Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, decorrendo atualmente na empresa ISA [1] sob a supervisão do professor Doutor Carlos Fonseca, orientador do Departamento de Engenharia Informática, e Engenheira Ana Guimarães, orientadora da empresa ISA.

1.1. Contexto

A ISA – Intelligence Sensing Anywhere é uma empresa especializada em telemetria e *Machine-to-Machine* sediada em Coimbra. Contando com um grande reportório de clientes como a BP, Galp, Primagaz e Butagaz, a ISA apresenta-se como a líder de mercado de telemetria em petróleo e gás [1]. Esta empresa, em crescimento constante, oferece soluções mundialmente reconhecidas com vista a otimizar processos na área da energia, gás e outros combustíveis. Isto permite aos seus clientes a recolha de dados em tempo real com extração de indicadores relevantes que auxiliam os seus gestores na tomada de decisões de negócio, bem como na otimização de processos já existentes. Grande parte destes processos começam com a instalação de equipamentos com sensores que facultam a monitorização em tempo real do estado de reservatórios de gás ou de outros combustíveis. A informação recolhida por estes dispositivos é paralelamente analisada através de *softwares* especializados também desenvolvidos pela ISA, e que fornecem ao cliente um conjunto de indicadores, auxiliando-o na gestão mais eficiente na produção, distribuição, transporte e armazenamento desses combustíveis.

Graças a esta filosofia, a ISA apresenta um elevado estatuto a nível mundial, sendo bastante requisitada por grandes nomes da indústria petrolífera. Com esta responsabilidade, a empresa tem de fornecer e instalar grandes quantidades de equipamentos pelos quatro cantos do mundo. Esta situação implica que, por sua vez, tenha de existir mão-de-obra para efetuar a manutenção e reparação de equipamentos avariados. Desta forma, seria insustentável para a empresa fazer todas estas operações por conta própria vendo-se, assim, necessitada a recorrer a mão-de-obra externa (subcontratados) para realizar grande parte destas intervenções.

A gestão e análise de custos com subcontratados e respetivos clientes finais é um processo mensal extremamente moroso que envolve vários colaboradores do departamento de operações da ISA. São utilizadas folhas de cálculo Excel nas quais os colaboradores introduzem, de forma manual, enormes conjuntos de dados provenientes de dois sistemas de informação usados pela empresa, o Primavera ERP e o MaisGás. Destas duas plataformas provêm registos de intervenções efetuadas por subcontratados e por técnicos da ISA, pagamentos aos subcontratados, entre outras informações relacionadas com clientes. Alguns desses dados têm de ser pré-processados noutras folhas de cálculo antes de serem incluídos no ficheiro Excel principal. A extração dos mesmos encontra-se, também, dependente de diferentes colaboradores com acesso aos dois sistemas em questão. Existe um responsável por recolher dados de intervenções, outras duas pessoas com a função de extrair dados associados a custos com subcontratados e por fim existem vários colaboradores que organizam e

introduzem toda essa informação num ficheiro Excel que permite visualizar os KPIs desejados.

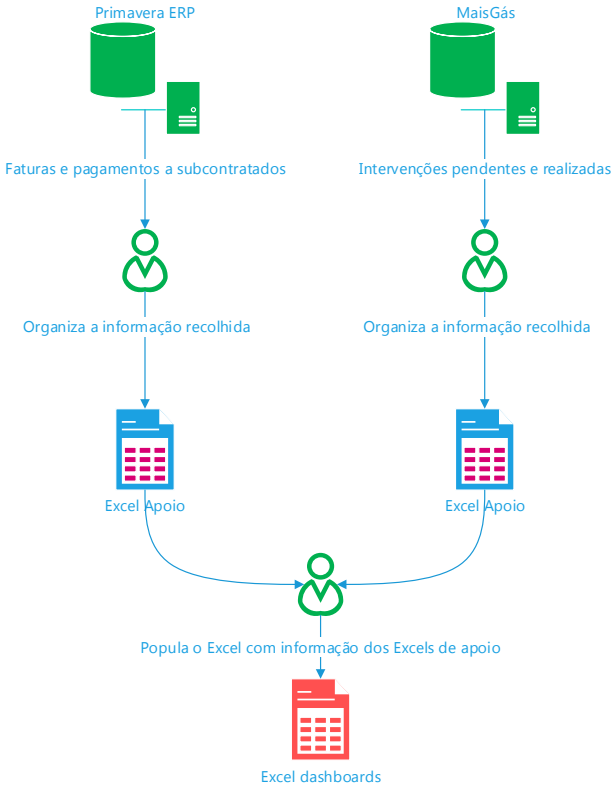

Figura 1 - Processo seguido pelos colaboradores do departamento de operações da ISA.

Este é um procedimento complexo, dependente e bastante suscetível à introdução de erros. Desta forma, surge a necessidade de agilizar todo este processo através da implementação de um sistema colaborativo e automatizado que permita extrair indicadores de negócio de forma contínua, economizando assim um total de cerca de 50 horas mensais despendidas pelos colaboradores neste processo.

1.2. Motivação

Apesar de a empresa utilizar sistemas como o Primavera ERP e o MaisGás para registar grande parte das interações com clientes e subcontratados, existe a necessidade de manter estes dados cruzados e de analisar os *KPI*s obtidos pela interpretação destas enormes quantidades de registos. Em proporções semelhantes, essas necessidades crescem à medida que o negócio se vai desenvolvendo, requerendo mecanismos cada vez mais complexos para a empresa se manter a par dos seus próprios indicadores e oportunidades. É sobre este paradigma que o mercado do *software* de *Business Intelligence* se tem vindo a basear e a crescer, apresentando inúmeras soluções para gerir o negócio do cliente alvo.

Motivada pela ineficiência dos processos seguidos atualmente para a ajuda no apoio e decisão na área das operações, o departamento de operações da ISA conferiu a necessidade de uma plataforma que permitisse agilizar todo este conjunto de procedimentos. Este estágio foca-se exatamente nisso, visa desenvolver um sistema à medida das necessidades da empresa, livre

de custos e que permita a integração com os vários sistemas usados na ISA. O produto final, resultante deste estágio, é totalmente direcionado para a própria ISA, não havendo planos de comercialização do mesmo. Desta forma, o cliente do produto é a própria empresa, em que os principais *stakeholders* deste projeto são os colaboradores pertencentes ao departamento de operações.

1.3. Estrutura do Documento

O presente documento encontra-se estruturado da seguinte forma:

- **Capítulo 2 - Apresentação do estágio**

  É feita uma apresentação dos objetivos principais deste estágio, do planeamento de tarefas, bem a apresentação dos riscos e desvios associados ao projeto.

- **Capítulo 3 – Estado da arte**

  São apresentadas as principais soluções existentes no âmbito de *Business Intelligence* de forma a ser estudado o que já existe na área mas também para ajudar na assimilação de pontos fortes a ter em conta no desenvolvimento de um sistema com os mesmos princípios.

- **Capítulo 4 – Requisitos**

  É elaborado um resumo dos requisitos do sistema partindo das necessidades apresentadas pelo cliente do produto.

- **Capítulo 5 – Arquitetura**

  É feita uma apresentação dos componentes constituintes do sistema, tecnologias utilizadas no seu desenvolvimento e o correspondente modelo de dados necessário.

- **Capítulo 6 – Implementação**

  São esclarecidas as principais soluções implementadas ao longo de todo o processo de desenvolvimento e demonstrados os resultados finais do produto.

- **Capítulo 7 – Controlo de Qualidade**

  São apresentados alguns processos seguidos no sentido de validar junto do cliente o produto desenvolvido.

- **Capítulo 8 – Conclusões**

  É feita uma análise geral relativamente aos objetivos atingidos neste estágio.

Capítulo 2
Apresentação do Estágio

Iniciado a 16 de setembro de 2013, o presente estágio decorreu sob o regime de *part-time* durante o primeiro semestre, sendo cumpridas semanalmente 16 horas de trabalho. A 10 de fevereiro de 2014 iniciou-se o segundo semestre e consequentemente foi estabelecido um plano de 40 horas semanais – regime *full-time*. A equipa deste projeto foi composta pela engenheira Ana Guimarães, como gestora de projeto, e pelo estagiário João Martins.

Neste capítulo pretende-se dar a conhecer ao leitor os principais objetivos definidos para o corrente ano letivo, bem como explanar as metodologias seguidas e os riscos e desvios associados ao projeto desenvolvido.

2.1 Objetivos

O presente estágio centrou-se no desenvolvimento de um sistema colaborativo para análise de dados relativos ao departamento de operações da ISA. O principal objetivo deste projeto foi agilizar um processo seguido pelos colaboradores da ISA que utilizava folhas de cálculo Excel para monitorizar um conjunto de indicadores de negócio relativamente às instalações e reparações levadas a cabo pelos subcontratados e técnicos da empresa.

O sistema desenvolvido no contexto deste estágio fornece aos colaboradores da empresa uma *web application* multiutilizador constituída por um conjunto de *dashboards* interativas com indicadores extraídos automaticamente dos vários sistemas de informação utilizados internamente na empresa e necessários ao processo – MaisGás e Primavera ERP.

De forma sucinta, este estágio tem como objetivo disponibilizar aos colaboradores do departamento de operações da ISA um sistema que permita:

1. Monitorizar as despesas com subcontratados e técnicos da ISA;
2. Monitorizar os países de ação dos subcontratados e a evolução das intervenções efetuadas nesses países;
3. Monitorizar a evolução de instalações e reparações de equipamentos efetuadas pelos subcontratados e técnicos da empresa;
4. Monitorizar a evolução de reparações e instalações nos clientes da ISA;
5. Definir alertas que notificam o utilizador quanto ao estado de indicadores de negócio monitorizados;
6. Analisar projetos da empresa;
7. Analisar o progresso individual de cada colaborador.

Na enumeração acima, o leitor poderá constatar a inclusão da análise de projetos e da análise de colaboradores, nos dois pontos finais, que não são objetivos diretamente relacionados com o processo a agilizar no departamento de operações. Foi também proposto pelos *stakeholders* a inclusão de mecanismos para monitorizar os projetos e colaboradores do departamento. Estes procedimentos requerem a integração com outro sistema utilizado pela ISA no planeamento e gestão de tarefas em projetos – o JIRA[i]. Pretende-se recolher dados deste sistema para analisar, essencialmente, a alocação dos recursos[ii] em projetos através do processamento dos *work logs* associados a *issues* dos mesmos. Toda a especificação de requisitos

[i] O leitor poderá ler mais sobre este sistema na secção 2.3.1 do presente documento.
[ii] Colaboradores.

e de arquitetura tem em conta a inclusão deste conjunto de objetivos associados a projetos e colaboradores.

2.2. Tarefas Realizadas

Inicialmente, o planeamento do projeto foi instaurado sofrendo, posteriormente, ligeiras alterações à medida que foi tomando rumo. Sendo um projeto sem qualquer *background*, todo este teve de ser delineado de raiz tendo no primeiro semestre passado por fases como, por exemplo, a recolha de necessidades junto dos *stakeholders,* a definição de requisitos de alto nível e detalhados, o estudo de soluções de arquitetura e a definição da própria arquitetura do sistema. O segundo semestre foi exclusivamente dedicado à implementação da plataforma desenvolvida e à realização de testes de aceitação.

Mês	**Tarefas Realizadas**
Setembro	• Estudo dos processos internos da empresa; • Apresentação dos objetivos do estágio; • Reuniões com *stakeholders* para recolha de necessidades.
Outubro	• Escrita de documento de requisitos de alto nível a partir das necessidades recolhidas; • Análise de aplicações a integrar o estado da arte; • Elaboração de *mockups* do sistema.
Novembro	• Escrita de documento de requisitos detalhados; • Estudo de abordagens para a arquitetura do sistema.
Dezembro	• Escrita de documento de arquitetura; • Estudo de *frameworks* de desenvolvimento; • Escrita destado da arte.
Janeiro	• Redefinição da arquitetura do sistema; • Elaboração de relatório de estágio.
Fevereiro	• Implementação do *scheduler* do sistema; • Integração com o MaisGás; • Implementação da API do sistema em Service Stack.
Março	• Desenvolvimento estrutura da *web application*; • Integração com o JIRA.
Abril	• *Design* da *web application;* • Alteração da arquitetura da API – migração para ASP .NET Web API 2 [2]; • Desenvolvimento de *dashboards* de intervenções, projetos e colaboradores; • Desenvolvimento de filtros temporais.
Maio	• Integração com sistema LDAP; • Implementação do módulo de autenticação e segurança de rotas; • Sistema de notas; • Desenvolvimento de mecanismos de exportação de dados; • Implementação de módulo de alarmes e sistema de notificações.
Junho	• Integração com o Primavera ERP; • Execução de testes de aceitação.

Tabela 1 – Tarefas realizadas ao longo do ano lectivo

A tabela acima representa as principais tarefas realizadas ao longo de todo o estágio. O trabalho planeado e realizado encontra-se também detalhado no anexo D.

2.3. Metodologia

Apesar de em grande parte dos projetos da ISA ser seguido um modelo ágil de desenvolvimento de *software,* mais concretamente Scrum [3], este projeto seguiu um caminho relativamente diferente desta metodologia. Tendencialmente os processos Scrum não evidenciam um planeamento futuro das várias fases de um projeto e não existe a necessidade de especificação de documentos de requisitos, havendo para isso um *product backlog* com os itens que necessitam de ser implementados. No entanto, no caso deste estágio, foram seguidos percursos algo distintos, tanto porque o projeto não tinha qualquer *background* e foi necessário produzir documentos de requisitos, mas também porque o primeiro semestre seguiu um regime de *part-time*. Este último ponto é talvez o mais relevante para haver o afastamento no modelo Scrum. Isto deve-se ao facto deste regime não garantir uma presença diária na produção de conteúdo para o projeto, bem como ao facto de existir a necessidade de se alargar a duração de *sprints* e aniquilar a existência de *daily meetings*.

Contrariando esse pressuposto, foram produzidos documentos detalhados de forma a estruturar e planear uma grande parte do projeto. Apesar disso, ao entrar em vigor o regime de *full-time* pode-se considerar que a metodologia a seguir passou a ser ágil. Não Scrum na totalidade pois não existiu um *product backlog,* mas foram definidos *sprints* de duas semanas com o intuito de desenvolver determinados itens, neste caso requisitos detalhados especificados no ponto 4.2. É claro que, seguindo este princípio, passaram a ser regulares também os agendamentos de *demos* do projeto com os *stakeholders* de forma a serem apresentados os avanços no projeto e a recolher *feedback* das funcionalidades produzidas. Deste aspeto surgiu, também, a necessidade de uma constante adaptação e modificação dos requisitos e arquitetura à medida que o projeto foi sendo implementado.

2.3.1. JIRA

O controlo sobre as tarefas (*issues*) a realizar neste projeto de estágio para cada *sprint* foi feito através da utilização da plataforma JIRA. Este sistema, permitiu que se mantivesse um histórico das atividades realizadas, mas ajudou, também, na organização de todo o planeamento de esforço/horas de trabalho e registo de problemas a resolver no sistema ao longo das iterações de trabalho - *sprints*.

De certa forma a especificação de *issues* associados aos requisitos da aplicação ou a problemas que se encontram por resolver, acabam por funcionar como um *product backlog* do projeto. Em cada *sprint* são definidas tarefas mais abrangentes e no caso de ser necessário podem-se especificar sub-tarefas mais detalhadas. Quando necessário, é registado no JIRA a quantidade de horas despendidas e o estado de determinada tarefa – resolvida, adiada para a próxima *sprint* ou até mesmo cancelada em caso alteração/cancelamento de requisitos. Podem também ser adicionados comentários ao *issue* em questão e monitorizar o estado geral do *sprint* relativamente ao planeado inicialmente.

Esta metodologia foi usada com sucesso ao longo deste estágio por facilitar uma melhor organização do projeto, permitindo também avaliar constantemente o estado em que o projeto se encontra e ajustar as estimativas de horas de trabalho para tarefas semelhantes em *sprints* futuras.

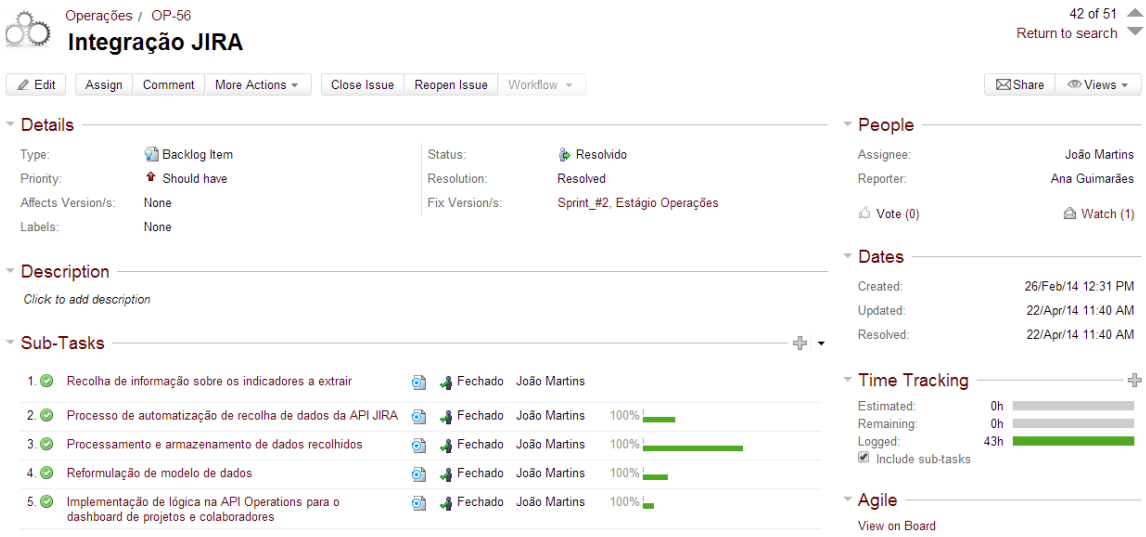

Figura 2 - Exemplo de um *issue* na plataforma JIRA e as suas sub-tarefas

2.3.2. SVN

Durante o desenvolvimento deste projeto foi utilizado o sistema de controlo de versões SVN da empresa. Foi criado um repositório especificamente dedicado ao projeto e todo desenvolvimento de código e até de documentos foi efetuada no *trunk* do mesmo até à versão 1[i] do sistema. Todos os *commits* efetuados ao repositório foram detalhadamente comentados quanto às funcionalidades implementadas e problemas resolvidos (*bugs*), de modo a que se mantivesse um registo histórico de toda a evolução do projeto e fosse possível a qualquer momento reverter o projeto ou partes do projeto para determinado ponto bem especificado na linha de tempo do mesmo.

2.3.3. Enterprise Architect

Foi utilizado o Enterprise Architect [4] para produzir os documentos de requisitos de alto nível[ii], requisitos detalhados e arquitetura. Este *software* largamente utilizado nos projetos da ISA, permite ao utilizadores modelar e planear a estrutura completa de um projeto de *software*, quer ao nível da definição de casos de uso, até mesmo ao nível da definição do modelo de base de dados. Este facilitou também o processo de rastreabilidade entre casos de uso e requisitos detalhados permitindo uma melhor organização no processo de modelação inicial do projeto.

2.4. Riscos do Projeto

Um projeto de *software* apresenta sempre riscos que nem sempre são identificados e precavidos quando este é inicialmente proposto. Numa primeira fase foram avaliados os riscos que poderiam ocorrer no projeto e as consequências que esses mesmos riscos poderiam ter levando ao insucesso deste estágio caso estes se materializassem. De seguida, são apresentados os riscos identificados.

[i] Versão com todos os requisitos implementados.
[ii] Documento que contém, essencialmente, uma visão geral das necessidades e casos de uso do projeto.

2.4.1. Interpretação de dados

O âmbito deste projeto implica que seja traduzido para um sistema informático a organização e interpretação de dados que até então são analisados através de uma folha de Excel. Estes mesmos ficheiros envolvem o cruzamento de vários tipos de dados relativamente a intervenções e a custos associados a subcontratados à empresa. A questão neste ponto não se prende com a utilização de técnicas avançadas de gestão no processamento destes dados ou com a utilização de complexas fórmulas matemáticas, prende-se sim na organização dessa informação em vários livros de Excel sem qualquer documentação da origem da mesma. Desta forma, foi identificado como um risco a má interpretação desses dados, tendo chegado a materializar-se.

Uma grande parte do tempo ao longo da especificação de requisitos e até mesmo na definição da arquitetura foi dedicado para conseguir relacionar os dados presentes no documento Excel já anteriormente referido. Estas relações de dados foram sendo assimiladas e traduzidas em algumas considerações a nível de arquitetura. No entanto, posteriormente, foram levantadas questões por parte do autor relacionadas com a origem de determinado conjunto de dados e de que forma eram obtidos. Nessa altura foram apresentados por parte dos *stakeholders* outros dois ficheiros Excel preenchidos por outros colaboradores e que eram usados no pré-processamento desses dados para que de seguida pudessem ser introduzidos na folha de cálculo final. Desta forma, esse conjunto de dados estava a ser mal interpretado por parte do autor devido à falha de comunicação criada desde o início do projeto. Este risco poderia ter originado problemas com proporções superiores se não tivesse sido identificado durante a definição da arquitetura.

Este risco foi mitigado através da interpretação conjunta dos ficheiros Excel juntamente com os *stakeholders* de modo a serem redefinidas as considerações arquiteturais tomadas.

2.4.2. Alterações durante o desenvolvimento

Todo o projeto de *software* deve identificar este risco como sendo possível em algum momento do seu desenvolvimento. Será quase impossível definir um projeto de uma ponta à outra sem ter de existir qualquer tipo de reformulações, quer ao nível de requisitos quer ao nível da arquitetura. No entanto, especialmente durante a fase de desenvolvimento do sistema, reformulações severas ao nível de requisitos e arquitetura podem constituir grandes problemas para o estado do projeto.

Este risco verificou-se no presente projeto ao nível da arquitetura, tendo existido a necessidade de se reformular alguns dos componentes do sistema inicialmente definidos. Este assunto será abordado na secção de desvios à arquitetura inicialmente proposta do capítulo cinco.

Este risco pôde ser colmatado através da metodologia ágil usada durante a fase de desenvolvimento. Os requisitos e questões arquiteturais foram constantemente reavaliados nas reuniões de *sprint* permitindo colmatar os problemas encontrados e explorar soluções alternativas.

2.4.3. Integração com outros sistemas

A dificuldade da integração da plataforma a desenvolver com os sistemas de informação da ISA foi também um risco identificado para o projeto deste estágio. A falta de documentação e a falta de conhecimento do autor em relação ao funcionamento das APIs e bases de dados destes sistemas poderia dificultar a implementação dos objetivos inicialmente propostos.

Para amenizar este risco foram efetuadas sessões de trabalho com o *solutions architect* da ISA – Engenheiro Luís Carvalho.

2.4.4. Curva de aprendizagem

A implementação da solução proposta necessita de integrar vários serviços e sistemas de informação da ISA, seguindo a arquitetura especificada no capítulo cinco. Esta arquitetura identifica a utilização de *frameworks* de desenvolvimento com que o autor não se encontrava totalmente familiarizado, como é o caso da linguagem C# (usada na API e *scheduler* do sistema) e AngularJS [5] sugeridas pela empresa. Este ponto ao mesmo tempo tornou-se um desafio e uma possibilidade de enriquecimento pessoal.

Apesar destas propostas terem sido aceites com grande entusiasmo por parte do estagiário é natural que este facto seja identificado como sendo um risco. A aprendizagem destas *frameworks* levou a processos mais morosos na conceção das funcionalidades requeridas, especialmente nas primeiras semanas de desenvolvimento.

Este ponto poderia ter trazido desvios ao nível do planeamento do projeto, no entanto, de forma a minimizar este risco foi necessário ao longo deste estágio levar a cabo um estudo permanente das *frameworks* utilizadas bem antes de se iniciar a produção de código em si.

Capítulo 3
Estado da Arte

O levantamento do estado da arte constitui uma fase de grande importância em qualquer projeto de engenharia de *software*. É através deste estudo que se podem descrever e avaliar as soluções atualmente existentes e, analisar o que estas oferecem. Deste modo, é fulcral examinar os pontos em que estas soluções carecem de perfeição, bem como tirar lições das características de sucesso que as representam. Com isto, pode-se compreender o que já existe neste campo de investigação e, ao mesmo tempo, indagar sobre o que é que projeto a desenvolver terá de se sobressair para se obter um produto final inovador e de qualidade.

3.1. Introdução

> *"A dashboard is a rich computer interface with charts, reports, visual indicators, and alert mechanisms that are consolidated into a dynamic and relevant information platform."* [6]

Em finais do século vinte, ferramentas como o Microsoft Excel eram a solução ideal para a maioria dos analistas, permitindo manipular, armazenar e analisar dados à medida das suas empresas [6]. Mas, geralmente, a frustração torna-se na principal criadora de inovação e, para satisfazer as necessidades emergentes em mundos empresariais tão competitivos, este tipo de análise de negócio tem vindo a ser posto de lado ao longo da última década. Um novo tipo de interpretação de dados utilizando *dashboards* colaborativos e mecanismos de recolha e extração de informação relevante, tem vindo a emergir substancialmente. A este conjunto de metodologias foi dado o nome de *Business Intelligence.*

O conjunto de produtos de *software* na área de *Business Intelligence* visa trazer ao ambiente empresarial soluções colaborativas de análise de dados para que o rumo do negócio seja facilmente compreendido e sejam exploradas novas oportunidades. Desta forma, ferramentas como o Microsoft Excel têm vindo a tornar-se obsoletas ao não cobrirem as necessidades cada vez mais exigentes no mercado empresarial. O trabalho colaborativo nesse tipo de gestão é extremamente suscetível à ocorrência de erros. Digamos que existe um ficheiro Excel partilhado por todos os colaboradores. Facilmente o leitor pode identificar que a adição, edição e leitura de dados em simultâneo do mesmo ficheiro poderá induzir imediatamente na extração de indicadores de negócio incoerentes. Para além disso, é notável que toda a introdução de dados terá de ser feita manualmente e que toda a definição de regras para extração de informação desses dados é controlada pelos próprios colaboradores resultando, uma vez mais, numa enorme probabilidade de apresentação de resultados erróneos. Este último aspeto prende-se com a dificuldade de integração de folhas de cálculo com outras ferramentas, tipicamente *data sources* necessários para à produção de indicadores de negócio.

Apesar dos obstáculos que os colaboradores têm na gestão destas folhas de cálculo, na generalidade, os motivos mais influentes para a utilização de sistemas de *BI*, são a boa usabilidade e a capacidade dos utilizadores identificarem novas oportunidades de negócio de forma simples. São poupadas horas de trabalho e são planeadas estratégias de negócio eficazmente. Neste sentido, ferramentas de folhas de cálculo como o Microsoft Excel, Apple Numbers, OpenOffice Calc e Google SpreadSheets não serão abordadas nesta análise por serem exatamente o tipo de *software* que se pretende substituir.

3.2. Análise de Produtos de Business Intelligence

O presente estado da arte descreve a análise efetuada às principais propostas de ferramentas de *Business Intelligence* existentes no mercado. Pretende-se, por fim, fazer não só um pequeno comparativo geral entre essas ferramentas mas também salientar as principais características que poderão influenciar positivamente o trabalho a ser realizado ao longo deste estágio.

As soluções abordadas ao longo deste capítulo foram exploradas utilizando versões *trial* disponibilizadas nos respetivos *web sites* durante o mês de dezembro de 2013. Para analisar os componentes e as características distintivas de cada produto foi estabelecido um modelo de testes. Foi criada uma folha de cálculo Excel com informação fictícia de despesas pessoais de um dado agregado familiar e, paralelamente, foi construído um modelo relacional simples para MySQL constituído por duas tabelas (pessoa e despesa) com informação semelhante à do ficheiro Excel. Deste modo, para cada uma das ferramentas, procedeu-se à importação de dados a partir destas *data sources* e foram efetuados os processos necessários no sentido de manipular esses dados para produzir *dashboards* com indicadores relevantes, como a evolução anual de despesas, gastos acumulados por pessoa, gastos por rúbrica, etc. Neste contexto, foram avaliadas ao detalhe as funcionalidades dos sistemas em análise seguindo essencialmente os seguintes critérios definidos pelo autor:

- **Usabilidade do sistema**

  Facilidade com que os utilizadores conseguem criar painéis integrantes dos *dashboards* e configurar funcionalidades associadas à plataforma.

- ***Data Sources* suportados**

  Origem de dados dos quais o sistema poderá recolher informação para disponibilizar nos *dashboards*. Por exemplo: motores de bases de dados MySQL ou SQL Server, ficheiros Excel, ficheiros CSV, APIs REST, SAP, etc.

- **Manipulação de dados extraídos**

  Opções disponibilizadas para o tratamento sobre os dados extraídos das *data sources* assim como a facilidade com que se especificam os indicadores desejados. Neste critério são fortemente valorizadas opções avançadas no tratamento de informação, como a aplicação de expressões matemáticas no relacionamento entre dados, a seleção específica de intervalos de elementos e o controlo oferecido na disponibilização dos indicadores resultantes dessa informação.

- **Qualidade de grafismo**

  Qualidade visual dos componentes criados para integrar os *dashboards*. Neste ponto é também tido em consideração a capacidade dos gráficos produzidos permitirem fazer *drill down* para mostrar informação mais detalhada.

- **Capacidade de elaboração de relatórios**

  Conjunto de mecanismos que permitam exportar resumos dos dados presentes nos *dashboards*. São incluídos quer serviços de exportação manual por parte do utilizador, como a exportação para PDF e ficheiros Excel, quer mecanismos de notificação automática por correio eletrónico sob a forma de *daily digests* ou mesmo a possibilidade de subscrever determinados indicadores.

- **Sistema colaborativo**

  Conjunto de funcionalidades que permita ao sistema integrar grupos utilizadores podendo atribuir-se ou não diferentes tipos de permissão aos vários integrantes para consultar ou editar os *dashboards* disponíveis.

- **Acesso**

  Capacidade do sistema poder ser, ou não, acedido interna ou externamente depois de efetuado o seu *deployment*. Normalmente este processo está diretamente relacionado com o facto de ser um sistema que permita a interação com vários utilizadores. Neste contexto é interessante analisar se esse acesso por múltiplos utilizadores é feito através de *mobile application*, de *web application* acedida através de um *browser* ou a partir de uma clássica *desktop application*.

É certo que a avaliação segundo os critérios supramencionados pode ser considerada um pouco pessoal e subjetiva, no entanto, tentou-se seguir rigorosamente os mesmos tipos de análise para cada uma das ferramentas em cada um dos critérios.

3.2.1. Tableau Desktop e Tableau Server

A Tableau Software, empresa americana de desenvolvimento de *software* fundada entre 1997 e 2002, é uma das principais gigantes na área de *Business Intelligence* e consequentemente com alguns dos produtos mais sonantes neste ramo. Com clientes como a Intel, eBay, Yahoo ou até mesmo a Google, esta companhia oferece aos seus clientes essencialmente cinco propostas de ferramentas de análise de dados, entre elas o Tableau Desktop [7], Tableau Server [8], Tableau Public, Tableau Reader e Tableau Online.

No contexto do presente estado da arte, a proposta ideal será o Tableau Desktop 8.1 Professional por se tratar da ferramenta mais completa e mais focada na criação de *dashboards* das apresentadas acima. Apesar destar apenas disponível para sistemas operativos Windows e Windows Server, esta trata-se de uma solução extremamente simples de usar e completa o suficiente para organizar e extrair informação de enormes quantidades de dados. A conexão a *data sources* como MySQL, SQL Server, PostgreSQL, SAP, Microsoft Excel e Access, Hadoop e muitas outras, torna-se imediatamente num enorme ponto a favor no que toca a características essenciais neste tipo de *software*. A partir destas *data sources*, a aplicação mantém sempre os dados constantemente atualizados, tratando de detetar automaticamente alterações nos seus conteúdos e atualizando imediatamente os componentes dependentes dessa informação. A partir de qualquer uma destas conexões, o utilizador consegue escolher, com rapidez, de forma extremamente intuitiva os dados que pretende incluir para formar uma planilha que consiste, essencialmente, num painel sob a forma de gráfico, *widget* ou tabela que juntamente com outras planilhas formam um *dashboard*. O Tableau oferece também opções mais avançadas de manipulação desses dados, permitindo relacionar informação de diferentes tabelas entre si e disponibilizando indicadores a partir de expressões matemáticas à medida do utilizador. Desta maneira é possível construir não só *dashboards* que por si só apresentam um *design* apelativo ao utilizador como, também, produzir *dashboards* interativos muito poderosos com indicadores criteriosamente selecionados.

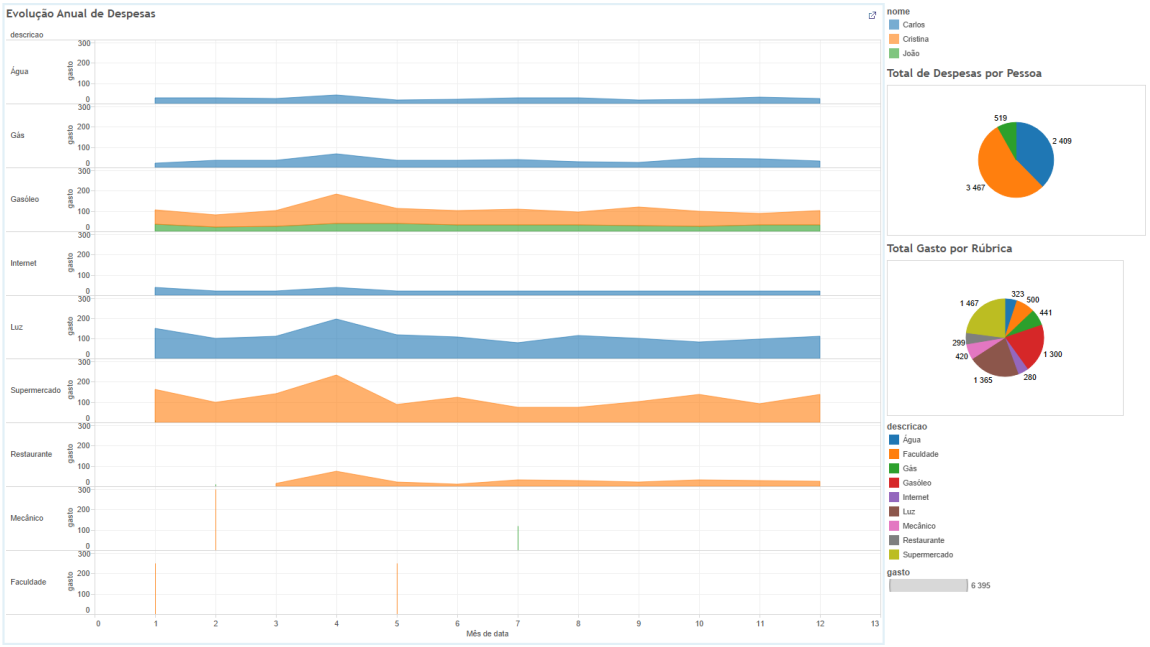

Figura 3 - Dashboard criado com o Tableau Desktop

Ao visualizar determinado *dashboard*, o utilizador poderá pormenorizar a informação que pretende ver apresentada em determinado componente através da utilização de filtros. Uma das características valorizadas que surge deste processo de filtragem, é que todos os outros componentes que sejam dependentes dessa informação filtrada são automaticamente atualizados permitindo ao utilizador analisar a influência que esse filtro tem nos restantes indicadores. A qualquer altura o utilizador poderá também exportar os presentes indicadores sob a forma de ficheiro PDF, folha de cálculo Excel ou até mesmo sob a forma de imagem. De forma a partilhar essa informação, o utilizador pode ainda disponibilizar *online* o seu *dashboard* através da ferramenta Tableau Public, ficando disponível para outros colaboradores uma versão *view-only* mas interativa[i]. Ao longo do estudo desta ferramenta foi sendo notória a facilidade com que se obtém todo um conjunto de funcionalidades implementadas graças à excelente usabilidade e organização do próprio editor de *dashboards,* permitindo justificar uma das frases que caracteriza o produto, "*anyone can become an analytics expert*".

No entanto, apesar deste produto apresentar muito boas referências, existem algumas características menos boas. A grande variedade de *data sources* apresentada não cobre ainda a crescente necessidade de integrar *software* de *BI* com outras aplicações do seio empresarial que forneçam dados através de APIs REST, algo já suportado por outras soluções ao longo deste capítulo. Outro entrave prende-se com o facto deste produto ser uma *desktop application* e como tal apenas permite a criação e utilização de ambientes interativos de *dashboards* na própria aplicação em si não permitindo haver a interação multiutilizador. Para colmatar este último obstáculo, a Tableau Software fornece também o Tableau Server. Este serviço permite fazer o *deployment* de um conjunto de *dashboards online* ou numa rede local, facultando a interação com vários utilizadores e permitindo a utilização das *mobile applications* para Android e iOS. Com o Tableau Server são também trazidas novas características à própria aplicação em si, nomeadamente, a possibilidade de fazer subscrições de indicadores de modo serem enviados alertas via correio eletrónico aquando a existência de flutuações dos valores definidos. Isto vem trazer aos colaboradores a capacidade destarem constantemente informados dos indicadores de negócio, permitindo efetuar importantes tomadas de decisão.

[i] Dashboard disponbilizado com o Tableau Public

3.2.2. Zoho Reports

O Zoho Reports [9] é um serviço de *BI online* disponibilizado pela Zoho que oferece um conjunto interessante de soluções para gestão de negócio. É facilmente percetível que este modelo *online* pode apresentar grandes vantagens no que toca a infraestruturas necessárias para o *deployment* de conjuntos de *dashboards*. No entanto, neste ramo de *software* grandes comodidades como estas implicam elevados custos mensais de manutenção, nomeadamente na versão empresarial em que são cobrados 495 dólares somados aos 6 dólares por cada utilizador ativo e 200 dólares se a organização necessitar de 10 milhões de registos de dados extra.

A partir deste princípio no uso de *thin clientes* verifica-se que esta é uma solução muito boa do ponto de vista colaborativo, visto que a acessibilidade por qualquer dispositivo é garantida através de um simples *web browser*, independentemente do sistema operativo e localização do colaborador.

A nível de *data sources* suportadas o Zoho Reports apresenta uma enorme variedade de sistemas de bases de dados, entre eles MySQL, SQL Server, PostgreSQL e Access mantendo constantemente a informação atualizada. O utilizador poderá também criar um modelo de base dados, sendo alocados os recursos necessários no próprio Zoho Reports. É possível, ainda, a importação quer local quer em localizações *online* de folhas de cálculo Excel, Google Spreadsheet e de ficheiros CSV. Por fim, esta solução apresenta um enorme ponto forte ao nível da importação de dados, pois disponibiliza a recolha de dados em formato JSON a partir de localizações na *web,* garantindo desta forma alta compatibilidade com possíveis integrações com serviços RESTful.

A criação de *dashboards* segue processos relativamente avançado, não sendo propriamente um modelo intuitivo para o utilizador comum. Daí, obter componentes gráficas com indicadores desejados não é um trabalho trivial, tornando-se até um pouco confuso de definir quais os dados que são utilizados e de que forma se relacionam. Apesar disso, a qualidade gráfica e a interação com os *widgets* criados é relativamente interessante permitindo fazer a exportação de relatórios para PDF e até mesmo para ficheiros HTML. O utilizador poderá também definir intervalos de tempo em que o serviço irá produzir relatórios que são enviados por correio eletrónico, ainda assim, não está disponível a funcionalidade de definir alertas específicos para indicadores presentes no *dashboard*.

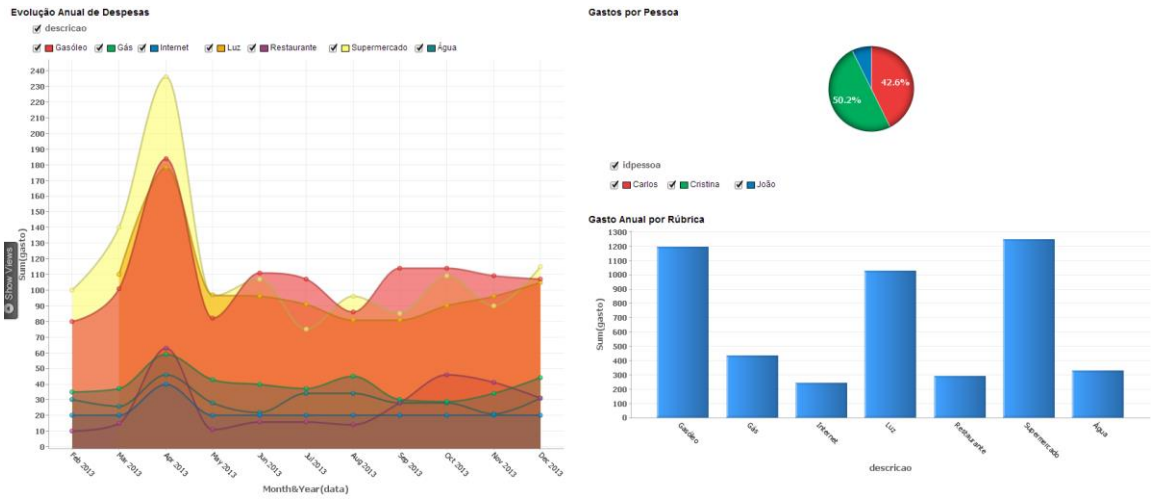

Figura 4 - Dashboard criado com o Zoho Reports

Mesmo existindo uma boa interação com os componentes gráficos que constituem os *dashboards,* a aplicação de filtros em *widgets* não afeta diretamente os outros componentes. Desta forma o utilizador não poderá analisar a influência que a filtragem tem sobre outros indicadores presentes, tornando-se talvez no principal ponto negativo em termos de usabilidade do Zoho Reports.

3.2.3. Visokio Omniscope

A Visokio [10], empresa de desenvolvimento de *software,* foi fundada em 2002 com o propósito específico de produzir um produto de *BI.* É só em 2005 que é lançada no mercado a primeira versão do Omiscope, solução que viria rapidamente a ganhar um leque considerável de clientes. Daí até aos dias de hoje a ferramenta tem vindo a evoluir bastante devido à colaboração muito próxima com os seus clientes, tentando satisfazer as suas necessidades.

Atualmente na sua versão 2.8, o Omniscope Desktop trata-se de uma *desktop application* disponível para Windows, Mac OS X e Linux. É ainda disponibilizado um *thin client* em Java que permite ao utilizador fazer uso de um simples editor de *dashboards* sem ter de instalar a aplicação. De forma a garantir um serviço colaborativo de sempre disponível, a Visokio oferece ainda a solução Omniscope Server que permite a autenticação de vários utilizadores, a definição de agendamento de relatórios sumarizados de *dashboards* e até mesmo o estabelecimento de ações a executar em *event driven.* Brevemente, na versão 2.9 será também lançada uma versão Mobile Server de modo a que utilizadores Android e iOS possam aceder através dos seus smartphones ou tablets aos *dashboards* alojados no servidor. De notar que a versão Server não foi testada ao longo da abordagem a esta ferramenta.

A nível de *data sources,* o Omniscope suporta grande parte dos habituais conectores para sistemas de bases de dados MySQL, SQL Server e Oracle. Tem ainda a particularidade de permitir importar e analisar dados provenientes de folhas de cálculo Excel, de ficheiros CSV, de ficheiros XML e de *cloud* - como o Google Analytics, Facebook API e Salesforce. Para sistemas mais exigentes é, também, possível ser feita a integração com *web services* permitindo ao Omniscope trocar dados com servidores remotos.

A manipulação dos dados extraídos é bastante simples, tornando-se extremamente fácil para o utilizador comum relacionar vários tipos de dados para extrair indicadores de negócio. Os elementos gráficos resultantes destas operações são de admirável qualidade porque permitem a interação com a maior parte dos seus componentes, *drill downs* muito bem apresentados e a filtragem de informação que influencia todos os outros *widgets* disponíveis no *dashboard.*

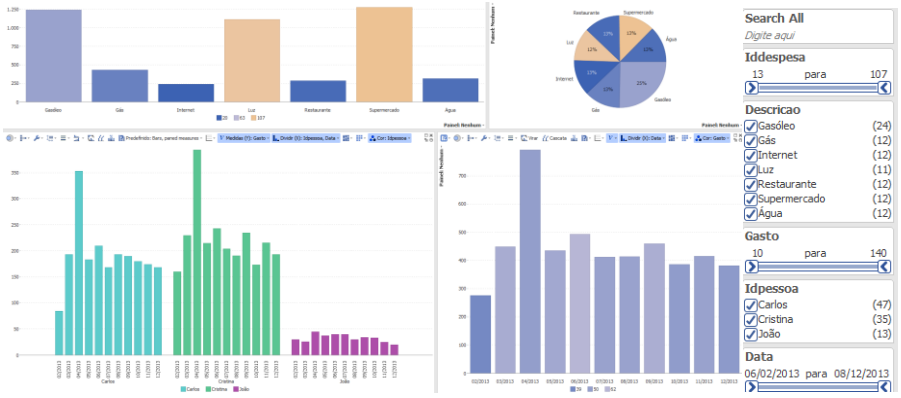

Figura 5 - Dashboard criado com o Visokio Omniscope

Outro ponto interessante a nível de usabilidade no próprio *dashboard* é a disponibilização automática de um painel lateral sob a forma de *faceted search* permitindo ao utilizador detalhar

a informação que é disponibilizada nos *widgets* presentes. Para utilizadores avançados, esta ferramenta permite ainda a utilização de JavaScript e HTML5 de modo a serem definidas *web views* completamente personalizadas, fazendo do Omniscope uma *"hybrid desktop/web application"* [11]. O utilizador final poderá a qualquer momento exportar relatórios dos dados filtrados ou do *dashboard* original para ficheiro PDF, PowerPoint ou até mesmo para uma imagem alojada *online*.

Na generalidade o Omniscope mostra-se uma ferramenta extremamente completa mas ao mesmo tempo muito intuitiva de utilizar. O seu preço base não se encontra apresentado no *web site* da aplicação.

3.2.4. Klipfolio Dashboard

O Klipfolio Dashboard [12], disponibilizado pela empresa de desenvolvimento de *software* canadiana Klipfolio Inc., trata-se de uma plataforma de criação de *dashboards online*. Desta forma, com a utilização de *responsive layout* pode ser utilizada sob qualquer plataforma. No caso de ser utilizado um *web browser* numa máquina fixa, a informação contida no *dashboard* será toda apresentada de forma tradicional como mostra a seguinte figura:

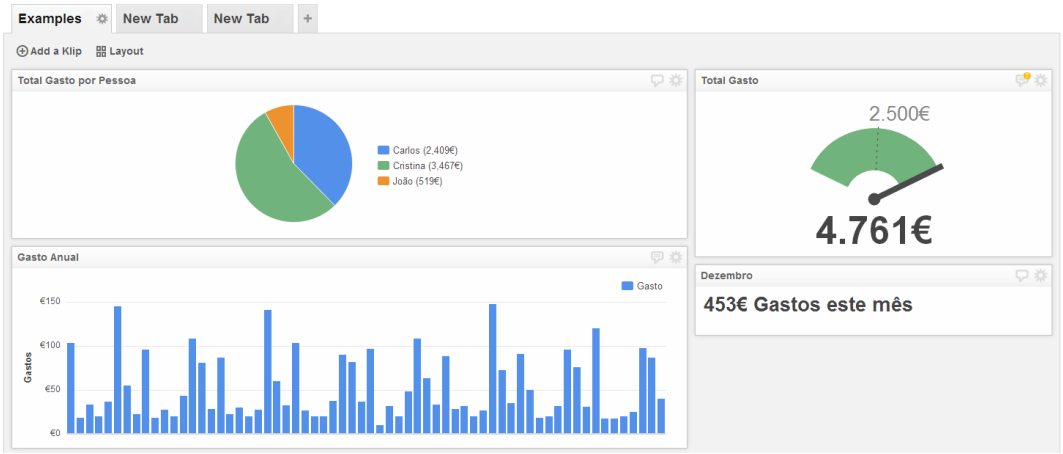

Figura 6 - Dashboard criado com o Klipfolio Dashboard

Na visualização do mesmo projeto num dispositivo *mobile,* os *widgets* presentes no *dashboard* são apresentados individualmente, não podendo ser feita uma análise geral a todos os indicadores. No entanto, será de notar que desta forma a informação é apresentada de forma nítida, algo que nem sempre é possível em *mobile web applications*.

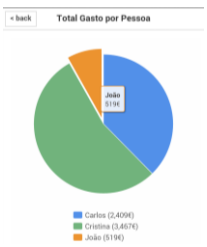

Figura 7 - Responsive layout usando o Klipfolio Dashboard num dispositivo mobile.

No geral, a qualidade gráfica desta solução é muito boa, fornecendo boa interação com os *widegets* criados - klips. Estes podem ser organizados segundo vários *layouts* predefinidos e fazer *drag and drop* desses elementos de modo a organizar o *dashboard* da melhor maneira. A informação que irá produzir indicadores relevantes sob a forma desses klips pode ter origem

em vários *data sources*. Para além da importação local das tradicionais folhas de cálculo, ficheiros CSV e XML, esta ferramenta permite também a interação com ficheiros remotos:

- Google Spreadsheet;
- Google Analytics;
- Facebook API;
- Folhas de cálculo alojadas na cloud Dropbox e Box;
- Salesforce;
- Ficheiros alojados em servidores FTP.

Outro ponto extremamente positivo, prende-se com o facto de ser possível estabelecer a recolha de dados disponibilizados por serviços RESTFul, retornando conteúdo em formato JSON. Como já foi falado, esta é uma característica de elevado valor num sistema de *BI,* especialmente em ambientes empresariais exigentes em que é necessário analisar dados crus provenientes de outros serviços. Neste caso em particular, o Klipfolio permite ao utilizador especificar com algum detalhe os parâmetros, o método a utilizar (POST ou GET) e a autenticação necessária para construir determinado pedido REST. Por último, o Klipfolio apresenta um défice no que toca a integração com motores de base de dados. Esta ferramenta permite apenas que se executem *queries* isoladas a bases de dados SQL em servidores remotos acessíveis ao exterior da rede em que se encontram.

A nível da manipulação de dados recolhidos das *data sources* acima mencionadas, o Klipfolio não se trata de uma plataforma muito intuitiva para ser usada por utilizadores comuns. Todos os dados têm de ser manualmente tratados e devem ser especificadas, com bastante cuidado, as expressões matemáticas para obter determinados indicadores. Por exemplo, a criação de gráficos de barras cumulativos torna-se algo complexo demais quando comparado com a forma trivial com que se executa o mesmo processo noutras plataformas. Desta forma, esperava-se um pouco mais de simplicidade na criação de klips básicos, podendo haver, se o utilizador necessitasse, opções mais avançadas no controlo e manipulação de dados.

O Klipfolio Dashboard permite o trabalho colaborativo entre vários utilizadores facilitando a tomada de decisões e análise de dados por diferentes colaboradores. Esta ferramenta disponibiliza também a criação de grupos de utilizadores com diferentes tipos de privilégios, garantindo o controlo sobre os acessos e edições doas *dashboards* existentes. As principais atividades executadas pelos utilizadores são registadas num *log* da aplicação.

Por fim, em termos de custos associados a este serviço, é cobrado um valor mensal que ronda entre os 4 e 17 dólares por utilizador, dependendo do número total de utilizadores associados a determinada companhia. Não é cobrado um valor fixo mensal para utilização do serviço, como é feito noutras aplicações *BI online.*

3.2.5. Datazen

O Datazen [13] trata-se de um novo paradigma de aplicações de *BI.* Este *freeware* mostra-se como uma solução simples e intuitiva para análise de dados em ambientes empresariais pouco exigentes. Encontra-se disponível na *Store*, como *Windows 8 App*, para criação, visualização e edição de *dashboards*. Também disponível para Android e iOS, o Datazen nestas versões *mobile* permite visualizar os *dashboards* previamente criados que estejam associados a determinada conta de utilizador.

A criação de painéis é feita a partir de modelos predefinidos podem ser arrastados para o *dashboard* principal. Posteriormente são configurados os *data sources* que irão fornecer informação a determinado painel. O Datazen permite apenas a importação de dados de

ficheiros Excel 2007+, SQL Server e Sharepoint Server, sendo bastante limitado neste campo de estudo.

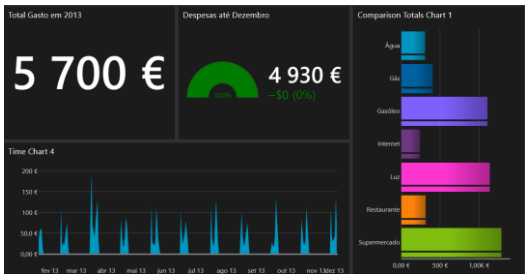

Figura 8 - Dashboard criado com o Datazen

O tratamento de dados obtidos a partir dos *data sources* mencionados é muito pobre, permitindo ao utilizador fazer poucas operações sob a informação recolhida. Além disto, o editor de *dashboards* nem sempre permite ao utilizador visualizar imediatamente os *widgets* com os indicadores selecionados, ficando apenas disponíveis em modo de visualização.

A nível de exportação de relatórios feitos a partir da análise de dados, o Datazen permite apenas gerar folhas de cálculo Excel 2007+, sendo que alguns dados, como datas, ficam corrompidos com esta exportação. O Datazen também carece de qualquer sistema de notificações/alertas aplicadas aos indicadores produzidos, ficando restrito à informação que o utilizador pode consultar utilizando a aplicação. Outro ponto menos positivo a nível de usabilidade, pelo menos para utilizadores pouco familiarizados com o Windows 8, é o facto desta ferramenta ser uma aplicação *full screen*. Desta forma, o utilizador se quiser cruzar dados com outras aplicações para possíveis tomadas de decisão, terá de andar constantemente a trocar de ecrãs de trabalho.

Na globalidade o Datazen é uma ferramenta muito simples mas carece de muitas funcionalidades importantes essenciais neste ramo de *software*. Além disso, a qualidade gráfica na apresentação e *drill down* de conteúdo não apresenta características suficientemente boas para se tornar numa aplicação de elevado valor para o utilizador.

O Datazen tem também disponível uma versão paga que permite a interação com vários utilizadores, no entanto, esta não foi explorada ao longo da análise efetuada.

3.2.6. Quadbase Enterprise Reporting

O Quadbase Enterprise Reporting [14] é uma plataforma de *BI* comercializada pela Quadbase Systems Inc., empresa norte americana sediada em Santa Clara – Califórnia. O Enterprise Reporting encontra-se disponível para Windows 7/8, Mac OS X e Linux mas também para dispositivos móveis Android e iOS. Estas últimas *mobile applications* permitem ao utilizador consultar em qualquer lugar os *dashboards* previamente produzidos.

Esta solução suporta a importação de dados de vários tipos de sistemas de bases de dados, entre elas MySQL, SQL Server mas também permite a integração com sistemas Java, utilizando Java Beans, ou até mesmo SOAP. O utilizador comum poderá também importar dados de folhas de cálculo. Neste campo, a Quadbase permite que o Enterprise Reporting seja ao mesmo tempo um sistema simples de instalar mas também deixa em aberto uma ampla possibilidade dos *developers* poderem construir ambientes mais complexos com a integração com outros sistemas de informação.

Esta plataforma permite trabalho colaborativo entre vários utilizadores. O *deployment* do sistema é feito através do servidor Tomcat que tem de ser necessariamente instalado numa máquina servidor. Desta forma, fica disponível uma *web application* com autenticação para os

vários grupos de utilizadores, que permitirá visualizar e interagir com os *dashboards* previamente criados.

A nível de criação dos *dashboards,* o Enterprise Reporting utiliza um mecanismo algo diferente do que é costume de observar neste tipo de *software*. Os elementos que irão constituir determinado *dashboard* são construídos individualmente através de uma *desktop application* – ERES Organizer. Daqui, o utilizador terá de utilizar a *web application* para definir toda a estrutura do *dashboard* a partir dos componentes criados.

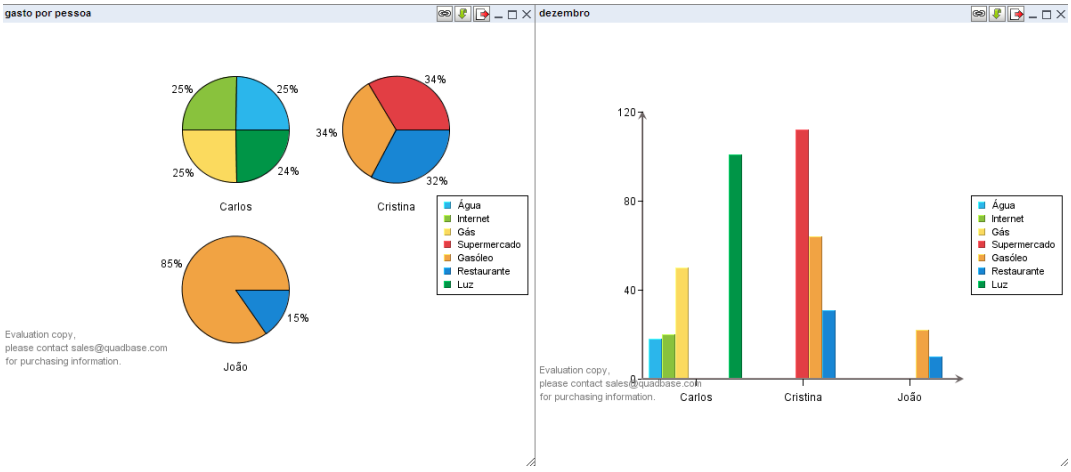

Figura 9 - Dashboard criado com o Quadbase Enterprise Reporting

O processo mais complexo ocorre na definição e manipulação da informação que irá constituir determinado elemento do *dashboard*, onde o utilizador terá à sua disposição uma enorme quantidade de opções avançadas para criar o mais simples indicador. A seleção de dados não é intuitiva e todas as operações sobre estes tem de ser manualmente criadas.

A qualidade gráfica dos *widgets* é rudimentar, não permitindo também muitas interações com os mesmos, como por exemplo *drill downs* ou até mesmo disponibilizar informação detalhada quando o utilizador coloca o rato em cima de determinado valor num gráfico. Por fim, importa também reportar o facto de a filtragem de dados não afetar outros *widgets* que dependam dessa informação, não permitindo ao utilizador tirar partido de mecanismos de análise de vários indicadores dependentes entre si.

Resumidamente, o Enterprise Reporting trata-se uma ferramenta extremamente avançada em termos de implementação de meios empresariais complexos conseguindo ser até bastante simples de configurar. No entanto, peca em termos de usabilidade tanto a nível do editor de *dashboards* como na *web application* ao ser usada pelos utilizadores finais. O preço da versão completa desta solução não se encontra disponível, sendo apenas indicado através do contacto com a Quadbase.

3.2.7. Arcplan Enterprise

O Arcplan Enterprise 8 [15], desenvolvido pela Arcplan, trata-se de uma solução de *BI* disponível na sua versão completa para plataformas Windows e em versões *mobile application* para Android e iOS apenas para consulta de *dashboards* previamente criados. No entanto, o *deployment* destes permite que os utilizadores da organização possam consultar esses *dashboards* a partir de qualquer *web browser* com suporte para HTLM5, garantindo compatibilidade de utilização final com qualquer sistema operativo.

O Arcplan Enterprise possibilita a importação de dados provenientes de mais de vinte *data sources* diferentes, incluindo folhas de cálculo Excel, plataformas SAP, sistemas de bases de dados que usem conectores ODBC (MySQL inclusivé), Oracle SQL, SQL Server e até mesmo *web services*. Apesar de a recolha de dados ser relativamente simples, a criação de *dashboards* é todo um processo muito mais complexo. Esta solução de *BI* não é dirigida sob a forma de modelos predefinidos, tendo de ser criado um projeto e de serem escritos através de *scripts* (.Net e HTML5) praticamente todos os componentes que irão constituir a aplicação. Pode-se dizer que o Arcplan Enterprise funciona na verdade como uma biblioteca de desenvolvimento de *software* que fornece aos utilizadores, neste caso *developers,* processos e recursos para a criação minuciosa de sistemas à medida da empresa. Posto isto, a análise a esta ferramenta seguiu um processo diferente ao serem explorados os modelos fornecidos pela própria Arcplan em vez de ser construído um *dashboard* de raíz.

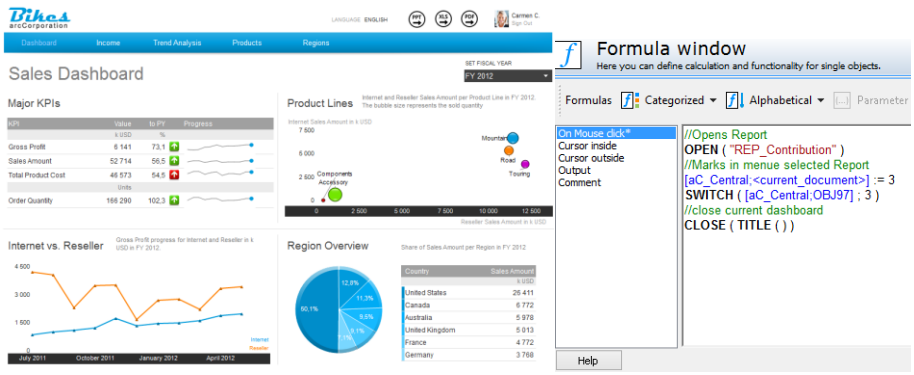

Figura 10 - Dashboard exemplo e script exemplo utilizando o Arcplan Enterprise

Todas as interações com os componentes integrantes de um *dashboard* são também criadas através de *scripts*. Por exemplo, para definir um simples *drill down* de informação ao passar com o cursor em cima de um mês do gráfico, terá de ser definido um processo como o que se encontra apresentado à direita na figura 8.

Nos exemplos disponíveis com o Arcplan, foi possível verificar a elevada importância que é dada à interação com os *widgets*. *Drill downs*, filtros que influenciam outros indicadores, entre outras interações gráficas são mecanismos que apresentam enormes níveis de qualidade nesta plataforma. A qualidade gráfica também não fica atrás, e o Arcplan disponibiliza uma grande variedade de *widgets* visualmente cuidados que os utilizadores podem implementar no seu projeto.

Foi também possível verificar que esta solução permite a definição de grupos colaborativos de utilizadores fazendo uso de perfis com autenticação para acesso aos *dashboards*. Dentro destes é possível também definir-se o *layout* específico para cada utilizador, podendo até restringir-se a visualização de determinado indicador a um grupo de utilizadores. Qualquer conjunto de indicadores pode também ser exportado para ficheiro Excel, PDF ou até mesmo PowerPoint permitindo aos utilizadores finais analisar dados *offline*.

Resumidamente o Arcplan Enterprise mostra-se como uma solução extremamente robusta e personalizável, no entanto, o nível de complexidade imposto para a criação de projetos mais simples pode ser um ponto que afaste o utilizador comum da compra deste produto. O preço de venda não se encontra disponível a não ser por contacto com a Arcplan.

3.2.8. Logi Info

Desenvolvido pela Logi Analytics, o Logi Info [16] trata-se de uma solução de *BI* disponível para os seus utilizadores finais sob a forma de *web application*. Encontra-se apenas disponível para Windows, fornecendo o Logi Studio, *software* que permite a criação de *dashboards*. O Studio permite a criação de *dashboards* através da construção de uma árvore[i] de componentes que representa uma página da *web application*. Através disto o utilizador poderá construir praticamente qualquer elemento que seja suportado por um *web browser,* isto é possível porque o Logi Studio funciona como uma biblioteca de recursos que assenta sob a criação de um projeto Java ou .Net. Pode-se assim perceber que o Logi Info se trata de uma solução mais focada para projetos com requisitos empresariais mais exigentes. Não é, de forma alguma, uma aplicação para o utilizador comum que não tem conhecimentos de desenvolvimento de *software*.

O *deployment* do sistema é feito com o Servidor Tomcat no caso de o projeto estar concebido em Java ou através de IIS do próprio Windows se desenvolvido usando .Net. Desta forma, fica disponível uma *web application* com os *dashboards* definidos permitindo o acesso seguro de múltiplos colaboradores através do Logi Secure Authentication. O facto de ser concebido e disponibilizado num ambiente de desenvolvimento, permite ao Logi Info incorporar todo um conjunto de código personalizado que seja necessário à aplicação, como ficheiros CSS, JavaScript e HTML.

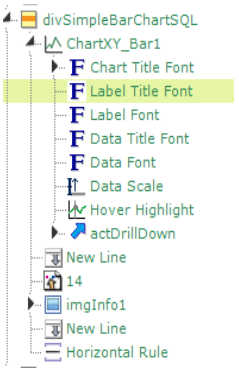

Figura 11 - Árvore de componentes de um projeto Logi Info

A importação de dados pode ser feita a partir de uma grande quantidade de *data sources* diferentes, como SQL Server, MySQL, serviços RESTful, ficheiros Excel e CSV, entre outros. A seleção dos dados a utilizar nos componentes criados não segue um processo muito simples, especialmente na integração com motores de bases de dados. Em vez do acesso direto e da especificação das tabelas de bases de dados que se pretende manipular, o Logi Info exige que o utilizador produza *queries* SQL para obter determinado conjunto de dados.

A qualidade gráfica dos *widgets* disponíveis por *default* com o Logi é bem cuidada permitindo também efetuar *drill downs* de dados com algum detalhe e mostrar informação relevante. Já o aspeto geral do próprio Logi Studio aparenta um *design* um pouco ultrapassado fazendo com que a usabilidade deste editor não seja propriamente uma referência neste ramo *software*. A criação de relatórios a partir de *dashboards* é também um processo pouco simples de definir, seguindo um modelo de *tasks* que têm de ser criadas manualmente especificando todos os elementos que irão ser exportados.

Para testar esta ferramenta foram utilizados *templates* de *dashboards* fornecidos com o Logi Studio. O preço de venda do produto não se encontra disponível.

3.2.9. Pentaho

O Pentaho [17], desenvolvido pela Pentaho Corporation desde 2004, trata-se de um dos nomes mais ouvidos na área de *BI*. Esta ferramenta *open source* desenvolvida em Java encontra-se disponível para sistemas operativos Windows, Linux e Mac OS X. Como se trata de um serviço disponível através de *web browser,* depois de feito o seu *deployment*, o Pentaho fica disponível para qualquer tipo de dispositivo - incluindo *smartphones* e *tablets*.

[i] Semelhante a um DOM de uma página HTML

A autenticação no User Console, serviço que permite a gestão e acesso aos *dashboards* e relatórios interativos, pode ser feita através do mecanismo de segurança disponibilizado pelo próprio Pentaho ou pode ser integrado com sistemas LDAP presentes em ambientes empresariais. Neste ponto central os utilizadores podem, segundo as permissões de cada um, consultar, criar ou editar *dashboards* e relatórios.

A usabilidade desta ferramenta é muito boa, sendo bastante intuitivo para o utilizador padrão conseguir aceder às principais funcionalidades em apenas um clique. Para além disso, o Pentaho tem disponível *online*, e na própria ferramenta, um repositório de documentação muito completo, disponibilizando também um conjunto de vídeo-tutoriais sobre como utilizar as funcionalidades principais da ferramenta. Como a construção de *dashboards* no Pentaho implica processos um pouco complexos, especialmente na integração das *data sources* e especificação de dados a tratar, esta ferramenta foi explorada utilizando os *templates* exemplo disponíveis na própria aplicação.

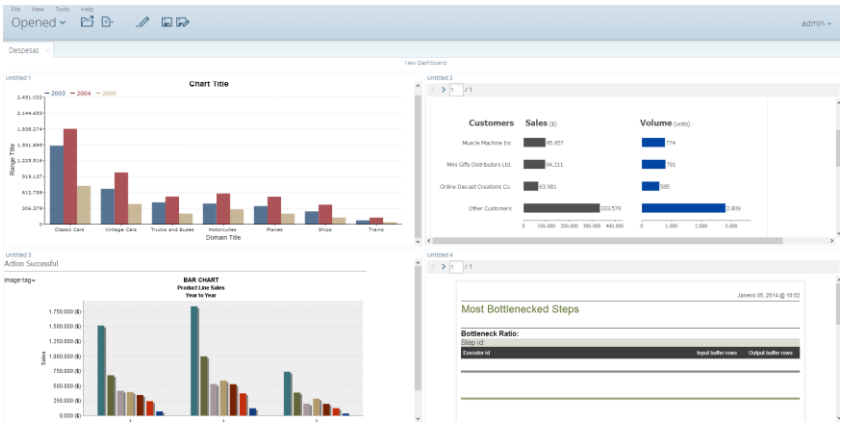

Figura 12 - Dashboard exemplo do Pentaho

O Pentaho permite a integração com um grande conjunto de *data sources* diferentes. Para tal é necessário utilizar o Pentaho Data Integration, que se trata de uma das aplicações que fazem parte do pacote Pentaho. É possível a importação de dados alojados em bases de dados SQL Server, MySQL, PostgreSQL, Hadoop, MonetDB e também diretamente de folhas de cálculo ou ficheiros CSV. Nesta ferramenta é possível definir através de diagramas o *workflow* necessário para produzir um conjunto de dados. Trata-se de um processo interessante do ponto de vista de simplificar e repartir as ações necessárias para gerar determinado resultado. No entanto, para mecanismos simples como a importação de uma tabela de um ficheiro Excel, torna-se trabalhoso demais quando comparado com a facilidade trazida por outras soluções de *BI*. A definição de alertas para gerar notificações via correio eletrónico também é feita através da criação de *jobs* no Data Integration que permitem definir intervalos de tempo em que os colaboradores irão ser notificados.

Esta ferramenta permite também a exportação de relatórios para o formato Excel e para HTML. A exportação para ficheiros PDF também é possível, no entanto, é necessário integrar bibliotecas como o PhantomJS para implementar este mecanismo.

Em síntese, o Pentaho trata-se de uma solução bastante completa e com capacidade para tratar e organizar dados em sistemas de praticamente todos os graus de complexidade. Isto torna-se bastante apelativo para os utilizadores que terão de criar e desenvolver os sistemas de *dashboards,* mas também garante que o utilizador final possa ter uma fácil experiência na obtenção dos indicadores de negócio desejados. A usabilidade oferecida para os utilizadores finais também constitui um grande ponto a favor nesta ferramenta, dispondo de forma

intuitiva o conjunto de ações que estes podem usufruir, bem como boas interações gráficas com os elementos dos *dashboards*.

O preço deste produto depende das especificações acordadas entre o cliente e a Pentaho, sendo que o preço base não se encontra disponível ao público.

3.2.10. QlikView

O QlikView [18] trata-se da última ferramenta analisada em detalhe neste capítulo. Desenvolvido pela QlikTech, empresa sediada na Pensilvânia, este produto é usado como ferramenta de análise de negócio por clientes como a Shell, Toyota, Canon e Panasonic [19]. A versão testada, QlikView Personal Edition (*freeware*), trata-se de uma ferramenta que se encontra apenas disponível para Windows. Nesta distribuição o QlikView inclui um editor de *dashboards* bastante completo e intuitivo, permitindo ao utilizador sem conhecimentos de desenvolvimento obter aplicações poderosas. Os *dasbhboards* criados poderão ser distribuídos sob a forma de *web applciation* utilizando a versão Server. Independentemente do sistema operativo, qualquer utilizador com um *web browser* com HTML5 poderá usufruir de toda a tecnologia oferecida por esta ferramenta. Encontra-se também disponível para iOS uma aplicação para visualização dos *dashboards* criados, no entanto, esta não permite usufruir de todas as funcionalidades da *web application*. Para dispositivos iPad, a QlikTech incentiva a utilização da *web application* que se adapta com enorme facilidade a este tipo de ecrãs [20].

A partir do editor de *dashboards,* é possível criar uma enorme panóplia de *widgets* que na generalidade apresentam um aspeto bastante cuidado e que permitem *drill downs* bastante eficazes. Detalhar a informação disponível nos vários indicadores irá influenciar diretamente outros *widgets* dependam desses dados. Como tal, o utilizador poderá visualizar de forma intuitiva as dependências entre os vários conjuntos de informação. Este trata-se de um ponto de elevada importância na análise a tomadas de decisão de negócio e no qual o QlikView consegue destacar-se de forma muito significativa.

Figura 13 - Dashboard criado com o Qlikview

A importação de dados no QlikView também segue um *workflow* bastante simples e eficaz, que permite, rápidas seleções e manipulações dos dados. MySQL, SQL Server, SalesForce, Microsoft Excel e RESTful APIs são apenas alguns dos *data sources* suportados. Apesar da simplicidade obtida utilizando os *configuration wizards* disponíveis, o Qlikview permite ao utilizador configurar um enorme conjunto de opções avançadas relacionadas com estas *data sources,* dando também a possibilidade de manipular e relacionar dados através de expressões.

Importa também referir que os dados são automaticamente atualizados independentemente do tipo de *data source*, não ficando dependentes de intervenção dos utilizadores.

Com a utilização da versão Server [21], toda a visualização de *dashboards* fica centralizada e é acedida através do que a QlikTech chama de *access point* – *web site* que permite a autenticação segura de múltiplos utilizadores. Posteriormente à autenticação, os utilizadores possuem um ponto colaborativo de negócio onde o Qlikview oferece várias funcionalidades inovadoras:

- Gravação do estado atual do *dashboard* para futura retoma no trabalho em qualquer dispositivo;
- Partilha de *dashboards* em tempo real com outros colaboradores;
- Sistema de notas colaborativas.

Os utilizadores podem ainda configurar sistemas de *reporting* automatizados que os manterão a par de mudanças nos indicadores especificados através do envio de relatórios por correio eletrónico. Este é sem dúvida um aspeto muito importante tanto na medida de notificar utilizadores ausentes da aplicação como no facto de permitir arquivar relatórios periódicos de dados importantes.

Sucintamente, o QlikView trata-se de um produto bastante completo e fácil de incorporar em qualquer que seja a dimensão de uma empresa, com a certeza que o utilizador comum terá uma curva de aprendizagem bastante rápida na utilização deste sistema. A qualidade normalmente implica elevados preços e o QlikView não é exceção à regra. Trata-se do *software* mais dispendioso de entre os que foram analisados neste estado da arte. A versão QlikView Client (versão licenciada do QlikView Personal Edition) tem um preço de 1350 dólares por cada utilizador registado enquanto a versão Server atinge os 35000 dólares por servidor instalado e 21000 dólares pelo sistema de *reporting* [22].

3.2.11. Outras ferramentas

Todas as ferramentas descritas anteriormente foram analisadas da forma mais pormenorizadamente possível a partir das suas versões *trial* gratuitas. No entanto, ficaram por testar várias soluções interessantes de *BI* devido a um conjunto de entraves.

- Jaspersoft[i] - ferramenta com um *workflow* incompreensível com a qual não foi possível obter qualquer tipo de resultados.
- InetSoft[ii] - após instalado, este *software* nunca chegou a ser executado com sucesso. Foi testado tanto em Windows 8 como em Windows 7 não se tendo encontrado solução para o problema.
- InsightSquared[iii] - solução de *BI* apenas disponível para sistemas SalesForce.
- iDashboards[iv] - foi preenchida uma requisição de uma versão *trial*, no entanto, esta nunca chegou a ser facultada.
- Dundas Dashboard[v]- foi preenchida uma requisição de uma versão *trial*, no entanto, esta nunca chegou a ser facultada.

[i] Web site do Jaspersoft - https://www.jaspersoft.com/
[ii] Web site do InetSoft - http://www.inetsoft.com/
[iii] Web site do InsightSquared - http://www.insightsquared.com/
[iv] Web site do iDashboards - http://www.idashboards.com/
[v] Web site do Dundas Dashboard- http://www.dundas.com/dashboard/

3.3. Análise Comparativa

A seguinte tabela faz um resumo dos principais critérios de avaliação para cada uma das ferramentas estudadas ao longo deste capítulo. Foram dadas classificações qualitativas para a maior parte dos critérios para que o leitor possa analisar facilmente a aptidão de determinada ferramenta no ponto em questão. É certo que esta avaliação pode ser um pouco tendenciosa por parte do autor, no entanto, todos os produtos foram analisados focando-se o mais objetivamente possível nos pontos chaves desta área de investigação, desprovendo-se de preferências preestabelecidas.

Produto	**Usabilidade**	**Data Sources**	**Manipulação de Dados**	**Elaboração de Relatórios**	**Múltiplos Utilizadores**	**Servidor**	**Preço**
Tableau	Muito Boa	Bom	Muito Boa	Boa	Sim	Sim	Moderado
Zoho Reports	Razoável	Bom	Razoável	Boa	Sim	Online	Moderado
Visokio Omniscope	Muito Boa	Bom	Bom	Muito Boa	Sim	Sim	N.D.
Klipfolio	Boa	Bom	Razoável	Razoável	Sim	Online	Aceitável
Datazen Free	Razoável	Insuficiente	Razoável	Razoável	Não	Não	Gratuito
Quadbase Enterprise Reporting	Razoável	Bom	Razoável	Boa	Sim	Sim	N.D.
Arcplan Enterprise	Boa	Muito Bom	Bom	Boa	Sim	Sim	N.D.
Logi Info	Razoável	Muito Bom	Bom	Boa	Sim	Sim	N.D.
Pentaho	Boa	Muito Bom	Razoável	Razoável	Sim	Sim	N.D.
QlikView	Muito Boa	Muito Bom	Muito Boa	Boa	Sim	Sim	Elevado

Tabela 2 - Comparativo de produtos analisados

3.3.1. Conclusões da Análise Comparativa

A partir da visualização da tabela 2, é possível verificar que ficam positivamente evidenciados três produtos: o Tableau, o Visokio Omniscope e o QlikView. Tratam-se de soluções bastante poderosas e personalizáveis à medida de qualquer empresa, independentemente do seu tamanho. Dispõem de grande parte das funcionalidades necessárias para transpor qualquer necessidade de análise de dados de uma empresa num conjunto de *dashboards,* serviços de *reporting* e de monitorização. Na generalidade apresentam bons níveis de usabilidade, quer para o utilizador final quer para os programadores do sistema. Os seus grandes entraves passam essencialmente pelos mesmos campos. O primeiro pretende-se com a dificuldade de implementação de sistemas mais exigentes, sendo necessário conhecimentos avançados em desenvolvimento de *software*. O segundo e último passa pelas quantias avultadas exigidas nos licenciamentos destes produtos, especialmente no QlikView.

O Quadbase Enterprise Reporting, o Arcplan Enterprise, o Logi Info e o Pentaho tratam-se de possíveis alternativas para as soluções apresentadas acima. São também bastante personalizáveis e permitem definir cenários empresariais complexos. No entanto, são produtos que não se destacaram da mesma maneira em termos de usabilidade, manipulação de dados provenientes dos *data sources* ou na elaboração de relatórios.

Já o Datazen destaca-se pela negativa, não apresentando avaliações deslumbrantes em nenhum ponto a não ser no facto de ser uma aplicação gratuita. Este *software* é bastante simples e apresenta um escasso número de funcionalidades, sendo mais indicado para interpretação de dados presentes em ficheiros Excel para utilizadores pouco exigentes.

De entre as propostas de *software BI online*[i] não existe na verdade uma que se destaque mais em relação a outra. Ambas apresentam falhas e alegam boas funcionalidades que trazem valor para o produto. No entanto, são soluções que devem ser aconselhadas para pequenas ou micro empresas uma vez que carecem da personalização e extensibilidade proporcionadas por outras aplicações analisadas.

3.4. Aplicação no Contexto do Projeto

Como foi possível analisar durante o presente capítulo, existe uma enorme oferta no mercado ao nível de soluções de *BI*. Contudo, a falta de soluções gratuitas e a complexidade associada a plataformas mais completas deram luz à iniciação do presente estágio e à construção de raiz de um sistema à medida, capaz de responder às necessidades e requisitos impostos pelos *stakeholders* do projeto. É de relembrar também que foi imposta a condição de produzir um sistema livre de custos, quer no desenvolvimento, quer ao nível de manutenção desta solução. Desta forma, é possivel verificar que os elevados custos dos produtos apresentados afastaram a possível implementação de uma solução existente no mercado de *BI*. Existindo estas condições *a priori* e tendo a ISA um departamento de desenvolvimento de *software,* faz todo o sentido que a solução pretendida seja produzida internamente e que seja construída de acordo com as necessidades do projeto.

[i] Klipfolio e Zoho Reports

Capítulo 4
Requisitos

O presente capítulo descreve os principais requisitos da solução a ser desenvolvida ao longo deste estágio. Para o leitor analisar de forma mais pormenorizada os assuntos explanados neste ponto deve consultar os documentos de requisitos de alto nível e requisitos detalhados que figuram em anexo. Estes, depois de concluídos, foram reavaliados junto do *stakeholders* de modo a ajustar falhas de interpretação por parte do autor.

Este projeto surge de um conjunto de necessidades apresentadas pelo departamento de operações da ISA, não estando, à partida, definido qualquer tipo de documentação, arquitetura ou outro documento de gestão de projeto.

De forma a fazer o levantamento de requisitos foram necessárias várias reuniões com os *stakeholders* do departamento de operações da ISA. Foi compreendido o problema em questão através da recolha das necessidades gerais e foram traçados os objetivos do projeto, descritos no ponto 2.1.

Serão de seguida apresentados os atores do sistema e um resumo dos requisitos detalhados associados a este projeto.

4.1. Atores do Sistema

Os atores constituem os intervenientes no sistema a desenvolver, quer sejam utilizadores finais ou serviços utilizados para o funcionamento do mesmo. Serão de seguida apresentados os atores identificados aquando a especificação do documento de requisitos de alto nível.

Ator	**Descrição**
Diretor	Utilizador com todos os privilégios sobre o sistema, estando destinado a ser usado pelo atual diretor do departamento de operações da ISA.
Utilizador Suporte/Administração	Utilizador com privilégios suficientes para consultar grande parte dos dados fornecidos pelo sistema. Destinado aos cargos de suporte e administração do departamento, responsável operacional e responsável da engenharia.
Técnico	Utilizador com privilégios para apenas consultar o *dashboard* de colaboradores[i]. Trata-se do nível mais baixo de acesso às funcionalidades do sistema.
Sistema MaisGás	Sistema utilizado para recolher informação de intervenções efetuadas.
Sistema Primavera	Sistema utilizado para recolher informação de custos associados a clientes.
JIRA	Sistema utilizado para recolher informação de projetos e colaboradores que participam nesses projetos.

Tabela 3 - Tabela de atores do sistema

Os casos de uso associados a estes atores e à sua participação no sistema a desenvolver podem ser consultados no anexo A no ponto 1.3.

[i] Tem apenas acesso à informação do próprio utilizador e que engloba os requisitos RD42, RD34 e RD48. Não poderá aceder à listagem de todos os colaboradores, nem à informação detalhada de outros colaboradores.

4.2. Requisitos Detalhados

Neste subcapítulo, são especificados sumariamente os requisitos detalhados deste projeto. Estes são divididos por várias secções como apresentado no diagrama de requisitos do anexo B no ponto 1.2, sendo que estas estão associadas a várias partes do sistema a desenvolver ou a conjuntos de funcionalidades gerais. O leitor poderá também visualizar nesse documento os *mockups* associados a cada uma dessas secções e analisar com maior detalhe as relações entre os vários requisitos e entre os casos de uso do documento de requisitos de alto nível.

4.2.1. Área de Login

Requisito	**Descrição**
RD01 Acesso ao sistema	O sistema deverá permitir a autenticação dos colaboradores. O mecanismo de autenticação poderá ser integrado com o próprio sistema de utilizadores e grupos LDAP presente na ISA.
RD02 Validação do acesso	O sistema terá de se certificar que o utilizador existe no sistema LDAP e em caso positivo verificar as permissões que este terá no sistema. Em caso negativo o utilizador permanecerá na área de *login* e poderá voltar a tentar autenticar-se.
RD03 Manter sessão ativa	Ao introduzir os dados da sua conta com sucesso, o utilizador passará a ser sempre reencaminhado para a página inicial da aplicação sem ter de introduzir as suas credenciais cada vez que aceder ao endereço do *web site* da mesma. A sessão manter-se-á válida durante 24 horas ou enquanto o utilizador não fizer *login* num dispositivo diferente durante essas 24 horas. Para cancelar uma sessão basta o utilizador fazer *logout* no sistema.

Tabela 4 - Requisitos da área de login

4.2.2. Sistema em Geral

Requisito	**Descrição**
RD04 Logout do sistema	O utilizador depois de autenticado poderá em qualquer altura fazer *logout* do sistema. Deste modo, para utilizar novamente o sistema terá de ser efetuada uma nova autenticação.
RD05 Sistema de navegação	Ao se autenticar, será apresentada ao utilizador uma *homepage* onde este poderá decidir para que funcionalidade da aplicação pretende navegar. Dentro de cada página a que o utilizador acede, este terá sempre à disposição uma barra de navegação que lhe permitirá navegar entre os componentes essenciais (de acordo com o seu nível de privilégio) da aplicação - *dashboards*, gestão de projetos e colaboradores, área do utilizador, administração do sistema e área de notificações.
RD10 Filtrar conteúdo	Em qualquer um dos *dashboards* de intervenções em que exista uma evolução temporal, ou seja, dados que estão dependentes de datas[i], o utilizador poderá utilizar um filtro disponível nesse *dashboard* que permitirá filtrar todo o conteúdo apresentado de acordo com o intervalo de tempo selecionado. Todos os gráficos e indicadores disponíveis deverão automaticamente adaptar-se ao filtro definido. Se o intervalo temporal for inferior a dois meses, os gráficos mensais passarão a disponibilizar o total semanal durante esse intervalo em vez do total mensal.

[i] Por exemplo a evolução de intervenções efetuadas por determinado subcontratado ao longo de um período de tempo.

RD20 Exportação de dashboard	O utilizador poderá a qualquer momento exportar para ficheiro PDF a informação contida no *dashboard* que se encontra a visualizar. O conteúdo a ser exportado estará também diretamente relacionado com os filtros escolhidos nesse *dashboard*. O utilizador poderá ainda exportar individualmente cada um dos gráficos interativos apresentados nesse *dashboard*. Como o nome indica, a exportação de *dashboard* só estará disponível nas áreas de *dashboard*.

Tabela 5 - Requisitos gerais do sistema

4.2.3. Área de Notificações e Alarmes

Requisito	**Descrição**
RD06 Lista de notificações	Na área de notificações o utilizador terá à disposição uma lista com todas as notificações associadas à sua conta. A partir desta listagem poderá selecionar as notificações que pretende marcar como lidas ou mesmo proceder à sua eliminação.
RD08 Gerir alertas	O utilizador poderá consultar os alertas que definiu até ao momento. Nesta listagem o utilizador poderá apagar individualmente determinado alerta mas também poderá visualizar quantas notificações determinado alerta gerou.
RD09 Sistema de notas	O utilizador poderá manter um registo de notas pessoais no sistema. Esta funcionalidade consiste em poder adicionar e eliminar pequenos lembretes que lhe sejam úteis para o uso da aplicação. Este sistema possuirá também um conjunto de *tags* que podem ser incluídas nas notas criadas.
RD21 Envio de notificações por email	Ao ser gerada uma notificação a partir dos alertas definidos, esta terá sempre de ficar disponível na lista de notificações do utilizador. No entanto, se este tiver ativada a opção de receber notificações por correio eletrónico, terá de ser enviado por correio eletrónico o conteúdo correspondente à notificação gerada.
RD13 Definir notificações por email	O utilizador poderá definir se pretende receber via correio eletrónico notificações correspondentes aos alertas que definiu. Mesmo sem ter esta funcionalidade selecionada, o utilizador será sempre notificado na central de notificações da aplicação.

Tabela 6 - Requisitos da área de notificações e área de alarmes

4.2.4. Área de Administração

Requisito	**Descrição**
RD11 Configurar intervalos de atualização de dados	O administrador poderá, nesta área, definir o horário a que o *scheduler* do sistema irá diariamente atualizar os dados necessários do MaisGás, Primavera e JIRA.
RD12 Listagem de logs do sistema	O administrador do sistema terá acesso a uma lista com entradas de *logs* da aplicação. Estes *logs* serão agrupados segundo as seguintes categorias: 1. Informação relativamente a atualizações/recolha de dados dos sistemas externos (MaisGás, JIRA e Primavera); 2. Informação de exceções ocorridas no sistema. Os *logs* serão constituídos por um ID, data/hora de criação e uma breve descrição textual.

Tabela 7 - Requisitos da área de administração

4.2.5. Dashboard Geral de Intervenções

Requisito	**Descrição**
RD22 Evolução geral de intervenções	O utilizador poderá visualizar um histograma com a evolução mensal do total de intervenções efetuadas e por realizar associadas a todos os subcontratados da empresa. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD23 Estatísticas gerais de intervenções	É mostrado ao utilizador a média de intervenções efetuadas por mês, por semana e o número de intervenções efetuadas na última semana. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*.
RD24 Intervenções realizadas e pendentes	O utilizador tem à sua disposição três gráficos de barras clicáveis (*drill down*) com o número total de intervenções, total de intervenções realizadas e total de intervenções pendentes. Ao clicar numa das barras, o utilizador poderá visualizar os clientes com maior influência para esses números. Em cada um dos gráficos existe a distinção entre as intervenções relativas a reparações e a instalações de equipamentos. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD25 Evolução de intervenções realizadas	O utilizador terá à disposição um gráfico de linhas que mostra a evolução das intervenções realizadas por todos os subcontratados da ISA. É possível no mesmo gráfico filtrar entre instalações e reparações. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD26 Evolução de intervenções pendentes	O utilizador poderá visualizar um gráfico de linhas que mostra a evolução das intervenções pendentes associadas a todos os subcontratados da ISA. É possível no mesmo gráfico filtrar entre instalações e reparações. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.

Tabela 8 - Requisitos do *dashboard* geral de intervenções

4.2.6. Dashboard de Intervenções por País

Requisito	**Descrição**
RD27 Distribuição por país	O utilizador poderá interagir com um gráfico circular com a distribuição de intervenções por país. É apresentado o valor em percentagem sendo que o utilizador poderá também consultar numa listagem paralela com o número total de intervenções para cada país. Cada país listado terá uma hiperligação que desencadeará a apresentação de um gráfico com a evolução das intervenções nesse país ao longo de determinado período de tempo. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD28 Evolução de intervenções por país	Se for selecionado um país na listagem de países do atual *dashboard*, será apresentado um gráfico com a evolução das intervenções nesse país ao longo de determinado período de tempo.

	O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual dashboard. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD29 Evolução de intervenções em todos os países	Será apresentado ao utilizador um histograma mensal com o total de intervenções realizadas por país. Neste gráfico é apresentado o acumulado por mês de todo o tipo de intervenções (realizadas, pendentes, instalações e reparações) efetuadas por todos os subcontratados nesse país. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual dashboard. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.

Tabela 9 - Requisitos do *dashboard* de intervenções por país

4.2.7. Dashboard de Análise de Custos com Subcontratados

Requisito	**Descrição**
RD31 Total adjudicado	O utilizador poderá visualizar um gráfico de barras correspondente ao total adjudicado por subcontratado. Será também disponibilizado o total adjudicado por todos os subcontratados. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD32 Faturas recebidas	O utilizador poderá visualizar um gráfico de barras correspondente ao total de faturas recebidas por subcontratado. Será também disponibilizado o total de faturas recebidas por todos os subcontratados. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD33 Pagamentos efetuados	O utilizador poderá visualizar um gráfico de barras correspondente ao total de pagamentos por subcontratado. Será também disponibilizado o total pagamentos por todos os subcontratados. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD35 Evolução financeira	O utilizador poderá visualizar um gráfico de linhas com a evolução mensal de adjudicações faturas recebidas e pagamentos efetuados. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.

Tabela 10 - Requisitos do *dashboard* de análise de custos com subcontratados

4.2.8. Dashboard de Clientes

Requisito	**Descrição**
RD36 Lista de clientes	Será apresentado ao utilizador uma lista com todos os clientes e o número de intervenções associadas a cada cliente. É apresentado também o país de cada cliente. Nesta lista cada nome de cliente é clicável apresentando detalhes das intervenções realizadas no mesmo. Os dados apresentados estão diretamente relacionados com o filtro temporal definido para o atual *dashboard*.
RD37 Evolução de intervenções por cliente	O utilizador poderá selecionar um cliente presente na lista e analisar a evolução das intervenções associadas ao mesmo. Esta evolução é apresentada sob a forma de um histograma mensal com o total de intervenções, total de instalações e reparações.

	O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD38 Estatísticas do cliente	Depois de selecionado um cliente da lista será apresentado ao utilizador um painel com informação do total de intervenções desse cliente, distribuição de reparações/instalações e a quantidade de intervenções que se encontram pendentes para o mesmo. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*.

Tabela 11 - Requisitos do *dashboard* de clientes

4.2.9. Dashboard de Subcontratados

Requisito	**Descrição**
RD40 Intervenções por subcontratado	O utilizador poderá analisar um gráfico de barras com o total de intervenções realizadas e pendentes associadas a cada subcontratado. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.
RD41 Seleção de subcontratado	O utilizador terá à sua disposição um *dropdown* com o qual poderá selecionar um subcontratado para analisar mais pormenorizadamente a evolução das intervenções associadas ao mesmo.
RD30 Lista de técnicos por subcontratado	Depois de selecionar um subcontratado, será apresentada ao utilizador uma lista com todos os técnicos associados a essa empresa (subcontratada) bem como o número total de intervenções feitas individualmente. Os dados apresentados estão diretamente relacionados com o filtro temporal definido para o atual *dashboard*.
RD39 Evolução de intervenções por subcontratado	O utilizador poderá visualizar um histograma semanal com o total de intervenções realizadas pelo subcontratado selecionado. Neste mesmo gráfico, se o utilizador tiver clicado num dos técnicos na listagem de técnicos desse subcontratado, será disponibilizada paralelamente a evolução de intervenções desse técnico. O intervalo de dados apresentado estará diretamente relacionado com o filtro temporal definido para o atual *dashboard*. O utilizador poderá ainda exportar individualmente este componente para ficheiro PDF, PNG ou CSV.

Tabela 12 - Requisitos do *dashboard* de subcontratados

4.2.10. Dashboard de Projetos

Requisito	**Descrição**
RD43 Seleção de projeto	O utilizador poderá utilizar um *dropdown* disponível no *dashboard* para escolher o projeto que pretende analisar. Esta escolha desencadeará a apresentação de informação detalhada do projeto selecionado.
RD44 Detalhes do projeto	Será apresentado ao utilizador um painel com informação relativamente ao número total de horas registadas por todos os colaboradores desse projeto, o número de tarefas resolvidas/por resolver e o número de colaboradores que participam no mesmo.
RD45 Lista de colaboradores do projeto	O utilizador poderá consultar uma lista com todos os colaboradores que participam no projeto selecionado. É disponibilizado o número de horas registadas por cada colaborador e a sua alocação no projeto (percentagem de horas nesse projeto). Cada colaborador terá uma hiperligação que reencaminhará o utilizador para outro *dashboard* onde poderá analisar com maior detalhe os projetos em que esse colaborador participa e as horas de trabalho registadas.

RD46 Evolução de horas de trabalho registadas	Será apresentado ao utilizador um gráfico de áreas com o total de horas registadas por semana por todos os colaboradores. O utilizador poderá, no entanto, alternar o modo de visualização deste gráfico para disponibilizar o total mensal de horas registadas.

Tabela 13 - Requisitos do *dashboard* de projetos

4.2.11. Dashboard de Colaboradores

Requisito	**Descrição**
RD47 Lista de colaboradores	É apresentada ao utilizador uma lista de todos os colaboradores registados nos projetos do sistema. Para cada colaborador é apresentado o número total de horas registado e o número de projetos associados ao mesmo. O utilizador poderá clicar na hiperligação do nome do colaborador e será apresentado um *dashboard* com informação mais detalhada.
RD42 Lista de projetos	Ao selecionar um colaborador, ou no caso de o utilizador estar a visualizar o seu próprio perfil de colaborador, será apresentada uma lista de todos os projetos em que o colaborador participa. Para cada projeto é possível ver o número de horas de trabalho registadas, sendo que o utilizador poderá clicar numa hiperligação para analisar a evolução de horas de trabalho registadas nesse mesmo projeto por parte do atual colaborador.
RD34 Evolução de horas registadas	Será apresentado no atual *dashboard* um histograma com o total de horas registadas mensalmente pelo colaborador.
RD48 Evolução de horas registadas por projeto	Quando o utilizador clica sobre a hiperligação de determinado projeto na lista de projetos do atual colaborador, é apresentado um histograma com o total de horas semanais registadas pelo colaborador nesse projeto. No mesmo gráfico é comparado o total de horas de cada semana registadas nesse projeto com o total de horas semanais em todos os projetos. Desta forma o utilizador poderá comparar semanalmente a alocação do colaborador em determinado projeto com o total registado em todos os projetos.

Tabela 14 – Requisitos do *dashboard* de colaboradores

Capítulo 5
Arquitetura

O presente capítulo pretende explanar a arquitetura do sistema desenvolvido neste estágio. Serão abordadas as decisões tomadas quanto ao *deployment* do sistema e a estrutura arquitetural das duas partições deste projeto – *front-end* e *back-end*.

5.1. Considerações Iniciais e Desvios

5.1.1. Sharepoint

No início da definição da arquitetura, foram levadas a cabo várias reuniões entre o *solutions architect* da empresa e o estagiário no sentido de estabelecer uma estrutura base para o arranque do desenho do sistema. Apresentados os requisitos do sistema, nomeadamente a necessidade de acesso à plataforma dentro da *intranet* da empresa, decidiu-se que este projeto seguiria o modelo apresentado na proposta de estágio – uma *web application*.

Foi sugerida a possibilidade de desenvolver todo o sistema em Microsoft SharePoint 2013 [23] com vista a poder integrar esta solução no SharePoint Server já existente na *intranet* da empresa que permite gerir as férias dos empregados. O principal objetivo desta decisão era que a empresa passasse a ter todos os novos serviços *web* centralizados numa só plataforma. Apesar de não existir qualquer conhecimento técnico, por parte do autor, a nível de SharePoint, foi investigada esta proposta de modo a analisar se todos os requisitos identificados poderiam ser implementados.

O estudo [24] [25] desta solução acabou por tomar algum tempo na fase de definição de arquitetura, chegando-se à conclusão que não seria possível desenvolver todo o sistema usando apenas SharePoint. As seguintes tabelas definem as propriedades identificadas utilizando um sistema puramente com desenvolvido em SharePoint.

Módulos	**Descrição**
Reporting Services [26]	Neste projeto existe a necessidade de recolher informação de três bases de dados Microsoft SQL Server utilizadas no sistema MaisGás e outra base de dados associada ao ERP Primavera. Utilizando Reporting Services não é necessário a criação de um modelo de base de dados para servir os dados destes sistemas à aplicação para extrair indicadores de negócio. Ou seja, toda a informação provém diretamente do servidor de base de dados, podendo definir-se o processamento automático da mesma de forma a extrair KPIs. A informação está sempre atualizada, não havendo necessidade de definir processos de recolha de dados através de um *scheduler*.
Sistema de alertas	Este serviço permite definir *workflows* que especificam quando devem ser geradas notificações para determinado utilizador no sistema.
Sistema de autenticação	O SharePoint Server permite integrar facilmente a autenticação de utilizadores a partir de um servidor LDAP, sendo possível gerir as permissões de grupos de utilizadores na área de administração.

Tabela 15 - Vantagens de utilizar uma solução SharePoint

Módulos	Descrição
Reporting Services	De forma a implementar este serviço é necessário ter permissões totais de controlo sobre as bases de dados de onde será recolhida a informação. Para usar este serviço é também necessário a instalação de um *plug-in* no próprio servidor de base de dados. Desta forma, o Reporting Services torna-se um serviço intrusivo trazendo enormes responsabilidades na integridade da informação nas bases de dados do sistema MaisGás e do ERP Primavera.
Desenvolvimento	O modelo de desenvolvimento de *web parts* em SharePoint segue princípios complexos e pouco adaptáveis às necessidades do projeto, sobretudo na integração com outros módulos/bibliotecas externas necessários ao projeto. O modelo Model-View-Controller seria o indicado para o desenvolvimento de uma *web application* com os requisitos identificados.
Sistema de alertas	O presente sistema do SharePoint não permite definir alertas para automatismos associados a Reporting Services. É apenas possível especificar o envio de notificações quando existem alterações em ficheiros em determinada biblioteca de ficheiros no SharePoint Server. O envio de notificações é só, aparentemente, possível através de correio eletrónico, não existindo um painel de notificações dentro do próprio Sharepoint.

Tabela 16 - Desvantagens de utilizar uma solução SharePoint

Apesar de ter sido descartada a implementação de um sistema totalmente desenvolvido em SharePoint, este iria ter, no entanto, um papel essencial na arquitetura deste projeto. Esta plataforma iria ser utilizada como ponto de acesso e de estrutura à aplicação a ser desenvolvida ao longo deste estágio. Seria reaproveitado o sistema de autenticação já existente do atual serviço SharePoint da *intranet* da empresa e o *user interface/design* do mesmo. Este aspeto tinha, como já foi referido, o propósito muito simples de unificar todos os serviços da *intranet* num único ponto de acesso e garantir uma experiência de utilização idêntica independentemente da aplicação/serviço da empresa que o utilizador fosse usar. Pretendia-se então utilizar o Sharepoint essencialmente como servidor de *front-end* e que permitisse ao mesmo tempo fornecer integração com o serviço de autenticação de utilizadores LDAP disponível na rede da empresa.

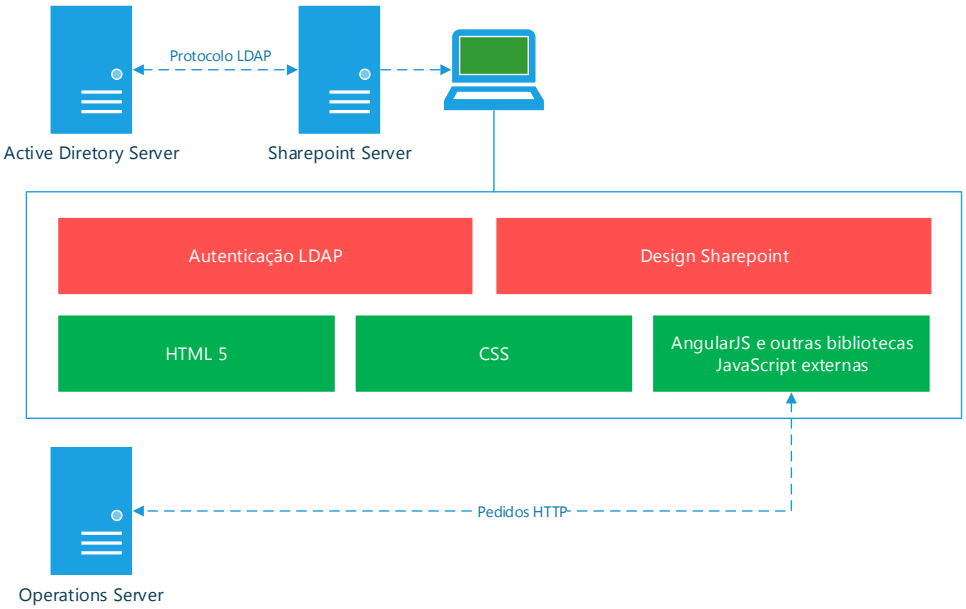

Figura 14 - Contexto de utilização do Microsoft Sharepoint

Desta forma, as páginas *web* fornecidas por este servidor iriam conter também uma parte lógica desenvolvida AngularJS que ficaria responsável por fazer a comunicação com o *back-end* do sistema[i], tratar os dados fornecidos pelo mesmo e fazer o controlo sobre qualquer tipo de comportamento lógico que a aplicação de *front-end* devesse ter[ii].

Esta decisão arquitetural manteve-se inalterada até meados de março de 2014. Contudo, acabou por ser posta de parte devido à inexistência de licenças de Sharepoint para ambiente de desenvolvimento enquanto o presente estágio decorreu. Este desvio ao planeamento trouxe consigo novos encargos ao nível da necessidade de implementação de um módulo de autenticação e segurança no projeto. Para além deste entrave, toda a parte de *design* da aplicação passou a ser da responsabilidade do estagiário uma vez que não estariam disponíveis as *guidelines* de Sharepoint. Esta alteração ao nível da arquitetura acabou por não ter consequências no trabalho desenvolvido até ao momento de identificação deste impedimento pois ainda decorriam os primeiros passos na implementação do *back-end* do sistema.

Posto este contratempo em análise, foi reformulada a solução para a arquitetura geral do *front-end* decidindo-se que seria utilizado o servidor *web* IIS para fazer o *deployment* do mesmo e que toda a parte de autenticação seria processada no espectro da API do projeto.

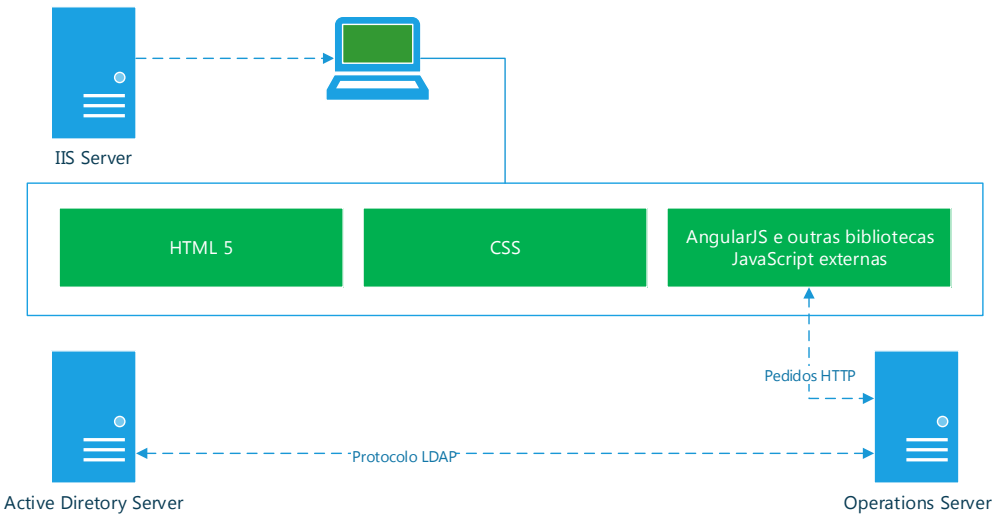

Figura 15 - Solução para *deployment* do *front-end*

5.1.2. ERP Primavera

A nível de integração com o ERP Primavera foram postas em cima da mesa duas possíveis soluções. Uma que passava pela utilização dos *interops* deste sistema interagindo diretamente com o mesmo para recolher dados necessários relativamente a faturas e a pagamentos de subcontratados. E por último uma solução que implicaria o acesso direto à base de dados deste ERP. Apesar de se ter planeado inicialmente seguir a primeira abordagem, por ser uma solução mais limpa do ponto de vista de desenvolvimento, existiram alguns contratempos que levaram à implementação da segunda opção.

Durante o mês de abril de 2014 a empresa mudou de instalações levando ao atraso em inúmeras semanas da disponibilização de um servidor de desenvolvimento com uma cópia do

[i] Neste caso a API do projeto.
[ii] Por exemplo: exportação de *dashboard,* filtros temporais, funcionalidades *event-driven,* entre outros.

ERP Primavera de produção. Este servidor havia sido pedido para que não tivesse de haver interação com a plataforma de produção que faz a gestão de dados críticos para a empresa, não colocando em risco a integridade dos mesmos. Devido aos atrasos sucessivos, foi pedido, por parte do estagiário, uma cópia da base de dados do ERP em produção e foi feita toda a integração à volta desta. Esta solução acabou por ser um trabalho mais moroso e abstruso do que o planeado pois não existe qualquer tipo de documentação da base de dados do sistema, havendo cerca de 1250 tabelas nas quais os dados necessários[i] ao projeto se poderiam encontrar e relacionar. No fim, este caminho acabou por se revelar como a melhor decisão tomada na altura uma vez que o servidor requisitado só foi disponibilizado durante a última semana de estágio.

5.2. Estrutura Geral do Sistema

Nesta secção é apresentada, sinteticamente, a estrutura do sistema repartida em duas partes, *front-end* e *back-end,* para que o leitor possa ficar familiarizado com os assuntos que serão abordados, com mais detalhe, nos próximos subcapítulos.

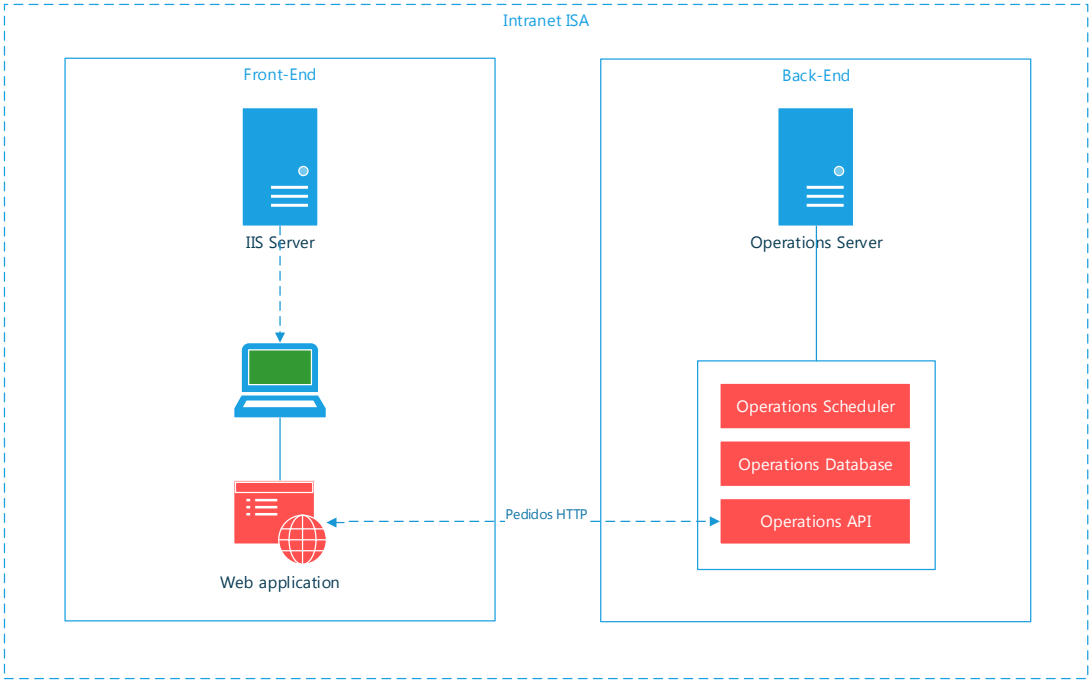

Figura 16 - Estrutura geral do sistema

De uma forma geral, a *web application* fornecida pelo IIS Server para o *web browser* do utilizador é o principal constituinte do *front-end* do sistema sendo, essencialmente, a única fonte de valor para o mesmo em todo este projeto. Trata-se do meio pelo qual o utilizador pode analisar dados que anteriormente eram processados de forma manual e disponibilizados em folhas de Excel.

A partir da figura acima, o leitor pode ainda verificar que existe apenas um canal de comunicação entre as duas camadas do sistema. Este canal é estabelecido com a Operations API, que tem o papel de controlar e fornecer todos os dados para o *front-end.* Todos os dados fornecidos para alimentar indicadores e gráficos da *web application* provêm, primariamente, do *scheduler* do sistema antes de serem fornecidos pela API. Este *scheduler* tem a obrigação de

[i] Faturas e pagamentos.

diariamente recolher dados dos três sistemas necessários ao processo – MaisGás, Primavera ERP e JIRA. Estes dados são armazenados e atualizados na Operations Database.

5.3. Front-End

Como já foi referido, o *front-end* do sistema é disponibilizado a partir de um servidor IIS. O utilizador acede a partir do seu *web browser* ao endereço[i] da aplicação dentro da rede da empresa ou através de VPN e rapidamente lhe será disponibilizada uma página da aplicação. O conteúdo desta *web application* é transferido para o *web browser* como qualquer outro *site*. É feito o pedido HTTP GET da primeira página HTML, neste caso uma página de *template* layout.html, que por sua vez, ao ser analisada pelo *web browser* irá fazer os pedidos para os restantes componentes da mesma – CSSs, JavaScripts e imagens.

Para além de HTML e CSS como constituintes de uma página desta *web application*, é usado AngularJS para tratar de toda a parte lógica do *front-end*. Esta *framework* de JavaScript, desenvolvida pela Google, tem vindo a ganhar grande notoriedade na comunidade de desenvolvimento *web* devido às suas elevadas potencialidades. Trata-se de uma *framework* bastante leve que traz para o *front-end* de uma aplicação um novo paradigma de controlo quer do DOM, quer dos dados recebidos em *runtime*. Esta metodologia permite definir, do lado do utilizador (*web browser)*, controladores que podem ficar responsáveis por processar todo o tipo de necessidades do programador, poupando recursos do lado dos servidores, aumentando a escalabilidade do sistema. Para além disso, dispõe de uma comunidade amplamente em crescimento, existindo cada vez mais *directives* [27] disponíveis para aplicar às necessidades de um projeto. Estas funcionam como extensões da aplicação, permitindo declarar no DOM *tags* que em *runtime* irão ser processadas para definir determinado comportamento ou a disponibilizar algum tipo de conteúdo. A utilização desta *framework* revelou-se com toda a certeza uma mais-valia neste projeto quer na estruturação da aplicação durante o desenvolvimento, quer ao nível dos mecanismos oferecidas pela mesma para atingir com conforto as funcionalidades pretendidas.

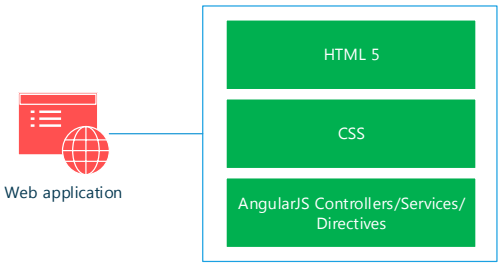

Figura 17 - Componentes de uma página da *web application*

5.3.1. Estrutura do Projeto de Front-End

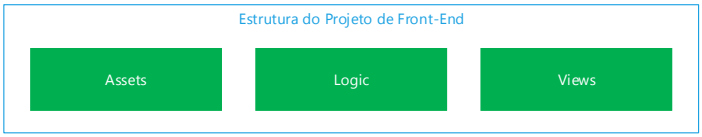

Figura 18 - Estrutura do projeto de *front-end*

[i] http://operationsdashboards.isa.pt/views/layout.html#/

Apesar de alguns componentes já terem sido apresentados, interessa conferir a estrutura geral do projeto de *front-end*. Tal como a figura acima demonstra, este encontra-se dividido em três diferentes partições:

- Assets – conjunto de ficheiros CSS, imagens e bibliotecas JavaScript da aplicação (incluíndo a distribuição de AngularJS) que serão usados.
- Logic – conjunto de controladores, serviços e *directives* de AngularJS. São essencialmente toda a camada que processa operações no *front-end,* elabora os pedidos/respostas efetuados ao *back-end* e realiza toda uma série de tarefas relacionadas com a geração de gráficos, exportação de *dabhboards* ou validação de rotas.
- Views – conjunto formado pela o *template* layout.html e *partial views* da *web application.*

5.3.2. Templating

Um dos mecanismos mais atrativos na utilização de AngularJS é o facto de toda a estrutura de uma *web application* poder ser centralizada num modelo SPA (Single Page Application). Isto quer dizer que toda a aplicação se fomenta num template, ou seja, numa única página que servirá como contentor de todas as outras páginas. A figura seguinte explica de forma simples a estrutura do template *layout.html* definido nesta *web application.*

Figura 19 - *Templating* usado na *web application*

No *template* usado neste projeto são definidas quatro áreas principais. No topo do documento é feita a declaração de todos os ficheiros de estilo usados na aplicação enquanto no fundo é feita a importação de todos os controladores/serviços/*directives* AngularJS entre outras bibliotecas externas de JavaScript[i]. Esta estruturação segue os princípios estabelecidos por Steve Souders no livro "Even Faster Web Sites" [28] em que este defende, corretamente, que duas das principais soluções para otimizar a performance de um *web site* são a declaração dos ficheiros CSS no top da página e dos JavaScript no fundo. A primeira prende-se com o facto de que se todos os ficheiros de estilo forem carregados logo no início do processamento da página que os componentes da mesma dependentes desses estilos irão carregar progressivamente já com os estilos aplicados. A segunda é defendida pelo facto de a transferência dos ficheiros JavaScript ser bloqueante. Ou seja, enquanto estão ficheiros

[i] Serão abordadas algumas dessas bibliotecas ao longo do capítulo de implementação.

JavaScript a serem transferidos do servidor, os componentes da página não irão ser apresentados enquanto estes não terminarem a transferência. Este ponto para além de trazer a sensação de demora na apresentação da página para o utilizador, pode levar a tempos mais elevados na transferência de todo o conteúdo dessa página devido ao bloqueio de transferências paralelas de outros tipos de ficheiros (imagens, ficheiros CSS e até mesmo outros ficheiros HTML usados na injeção do *template*).

A terceira zona em destaque neste documento é a estrutura visualmente estática da aplicação, constituída pela barra lateral, barra de navegação e *footer*. Tratam-se de três estruturas constantes ao longo de todas as páginas do projeto e como tal não existe a necessidade de carregar sempre este conteúdo a cada pedido efetuado para uma nova página. Na realidade, é neste ponto que entram as vantagens de uma estrutura SPA. Existe uma quarta zona definida no ficheiro *template* que servirá como um contentor de HTML em que será injetado todo o código pertencente a determinada rota da aplicação, ou seja, todo o código dinâmico. Por exemplo, pretendendo-se aceder à área de projetos da aplicação com a rota "/projects", a *framework* AngularJS irá injetar o código HTML definido numa *partial view* denominada "projects.html" na zona definida para esse fim no *template*. Esta zona é definida através da *directive* `data-ng-view` do AngularJS especificando que é nessa `<div>` que o código dinâmico irá ser embutido.

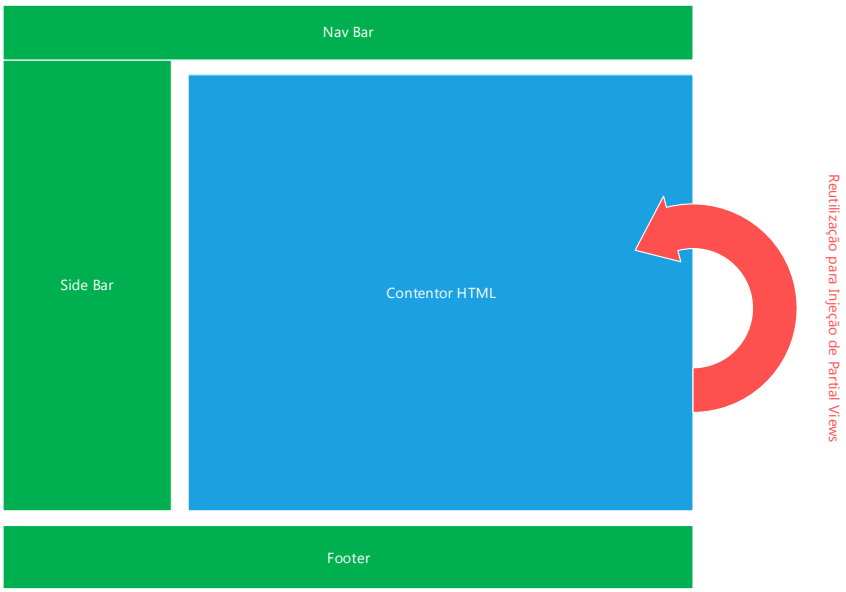

Figura 20 - Utilização de *templating* para injeção de *partial views*

O uso desta metodologia permite que se evite a repetição de código estático em cada página que se pretende apresentar e, principalmente, leva a que o carregamento das páginas seja bastante mais rápido. Este último ponto prende-se com o facto de os componentes de toda a aplicação serem carregados apenas na primeira página que o utilizador consulta. Seguidamente, para todas as rotas requisitadas, é apenas transferida a *partial view* e imagens necessárias à página em questão. É certo que no primeiro acesso são descarregados todos os ficheiros CSS e JavaScripts (*assets)* que serão utilizados independentemente da zona da aplicação que o utilizador aceda. No entanto, em deterioramento do tempo de resposta de carregamento da primeira página, os acessos a páginas seguintes são praticamente instantâneos, dando ao utilizador a perceção de que não existiu a troca de rota não havendo o recarregamento/desaparecimento momentâneo do conteúdo estático.

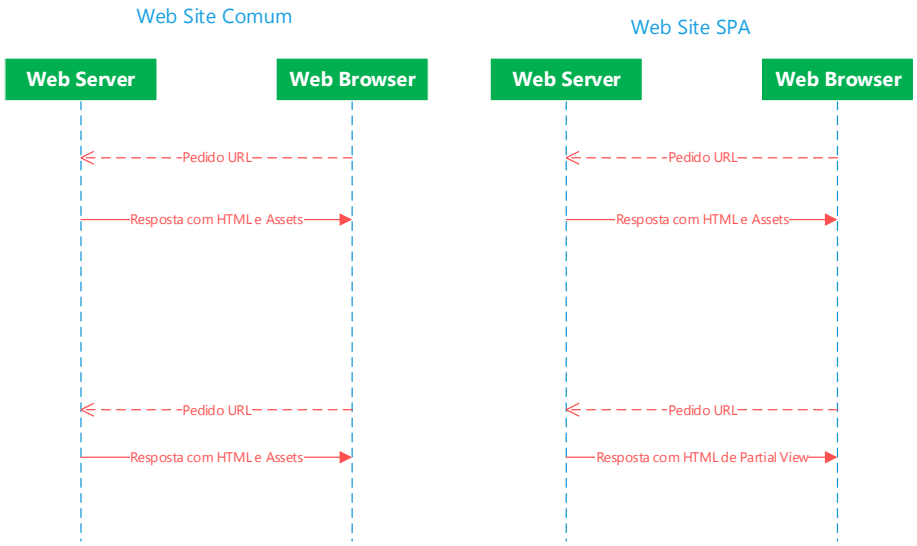

Figura 21 - Diagrama de sequência de pedidos feitos pelo *web browser* a um *web server*

5.3.3. AngularJS Application

A arquitetura do *front-end* ou *web application* está inteiramente dependente dos princípios seguidos pela *framework* AngularJS. Como já foi visto até agora, toda a aplicação está relacionada com um *template* para efetuar todo o processo de injeção de código para cada página requisitada. Este ficheiro não tem só o papel de fornecer o *template* da aplicação mas também de iniciar a aplicação em si. A *tag* `data-ng-app="operationsapp"` declarada no topo do HTML deste ficheiro indica que a *framework* irá inicializar o módulo "operationsapp", que consiste na verdade no controlador central de toda a aplicação.

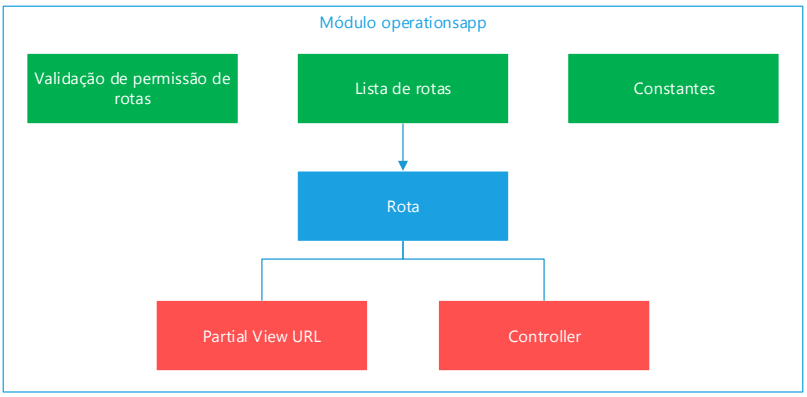

Figura 22 - Componentes do controlador central da *web application*

A partir da figura acima o leitor pode verificar que este módulo/controlador central desempenha três funções:

- Validação de rotas – sempre que é requisitada uma nova rota da aplicação, esta ação é processada neste controlador e são verificadas as permissões do utilizador no acesso à mesma. Esta verificação é feita tanto ao nível do *front-end* como no *back-end* e poderá ser melhor analisada no capítulo de implementação na secção de autenticação e segurança.

- Lista de rotas – é também neste controlador que são definidas todas as rotas disponíveis na *web application*. Esta lista funciona essencialmente com um conjunto de registos, um por cada rota, em que são definidas três propriedades:
    - O próprio nome da rota. Por exemplo: "/projects";
    - O controlador AngularJS associado a essa rota. Por exemplo: "projectsController". Importa referir que uma dada página/rota pode conter mais do que um controlador associado a diferentes áreas dessa página, no entanto, o controlador definido nesta propriedade é considerado como o controlador central dessa página;
    - A *partial view* associada à rota. Por exemplo: "partials/projects.html". Trata-se do caminho para o ficheiro HTML que contém o código a ser injetado no *template* da *web application* quando a rota "/projects" é requisitada.
- Constantes – por fim, este controlador também tem é usado como local de declaração de constantes do projeto. Entre elas estão o URL da API do sistema, a versão da *web application*, conjunto de cores usadas, entre outras constantes necessárias.

Importa também constatar que o módulo central "operationsapp" também disponibiliza automaticamente um componente importante para toda a *web application* - o `$rootScope` [29]. O AngularJS funciona à base de `$scopes` para interligar e ao mesmo tempo separar a camada de HTML dos controladores. É através de `$scopes` que é feito o *binding* de toda a informação que se pretende passar dos controladores para a camada de visualização HTML[i] e vice-versa. É importante frisar que o `$scope` de cada controlador descende do `$rootScope`, sendo que este acaba por funcionar como um contentor estático ao longo de toda a aplicação que pode ser acedido em qualquer controlador ou até mesmo pelas próprias camadas de visualização HTML. O `$rootScope` e os seus conteúdos são mantidos automaticamente ao longo de toda a aplicação sem ter de haver qualquer tipo de *handling* para que este seja acedido no atual controlador.

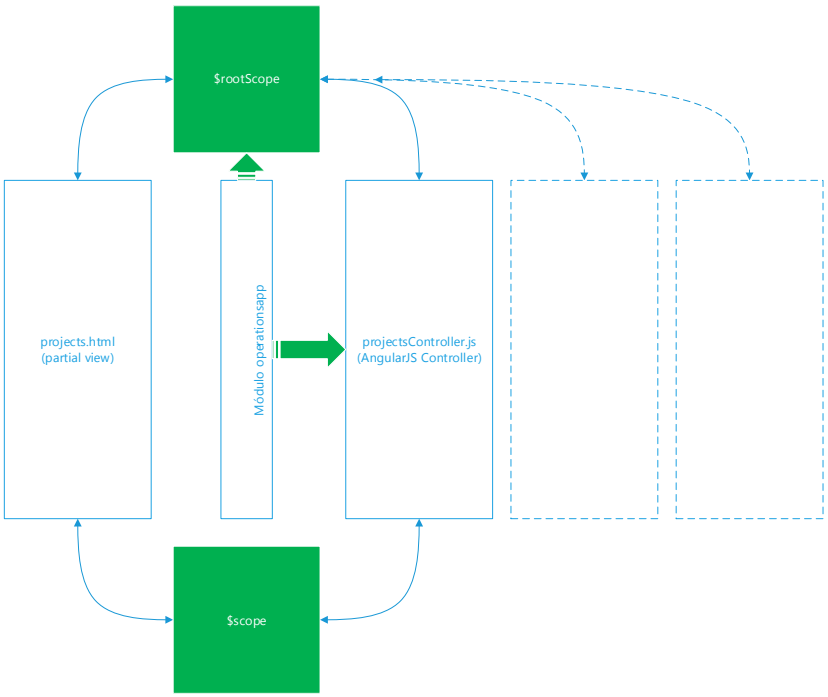

Figura 23 – Estrutura camada de visualização – controlador

[i] *Template* ou *partial views*.

5.4. Back-End

O *back-end* do sistema foi totalmente desenvolvido usando a linguagem C# culminando numa solução constituída por cinco projetos que se interligam.

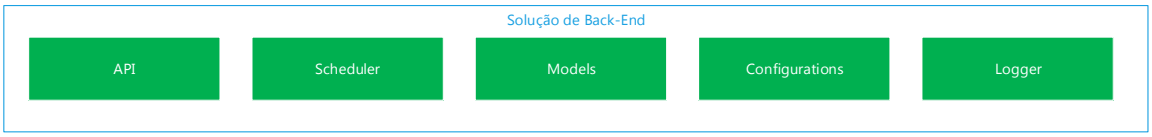

Figura 24 - Solução de *back-end*

Ao longo deste subcapítulo é apresentada uma breve descrição do modelo de dados utilizado e da tecnologia usada à volta do mesmo. Serão também abordadas as principais estruturas arquiteturais da API e Scheduler do sistema. Não será feita uma análise detalhada aos projetos Configurations e Logger uma vez que se tratam de projetos utilitários para o sistema. Como os nomes indicam, o projeto Configurations disponibiliza configurações/parâmetros gerais na solução de *back*-end, enquanto o projeto Logger permite fazer o registo para ficheiro das atividades e exceções da API e do Scheduler usando a biblioteca log4net [30]. Este último permite monitorizar o estado da aplicação já fora do ambiente de desenvolvimento, ou seja, na fase em que o *back-end* é colocado em produção.

5.4.1. Modelo de Dados

Esta subsecção apresenta uma descrição sumarizada do modelo de dados usado no *back-end* do sistema. Uma vez que a definição do documento de arquitetura levou à especificação de um diagrama de entidades detalhado, foi exportado e aplicado esse mesmo diagrama na criação de uma base de dados num servidor SQL Server 2012. Ficou-se desta maneira com um modelo de dados completamente definido. No entanto, a necessidade de trazer para o ambiente de desenvolvimento uma solução que evitasse o uso exaustivo de *queries* com múltiplas relações levou à pesquisa de uma tecnologia semelhante a JPA [31], com a qual o estagiário já havia tido contacto em projetos Java. Foi encontrada uma tecnologia com propósitos muito semelhantes ao já conhecido JPA - a Entity Framework 6 [32]. Com esta solução torna-se possível mapear todo o modelo de base de dados para classes em C#.

Foi usado, primeiramente, o processo de engenharia reversa Database First [33] para, a partir de um esquema de base dados já existente, serem criadas as respetivas classes C# com todas as relações necessárias entre si. No entanto, é sabido que um projeto de *software* está sempre sujeito a mudanças ao longo do seu desenvolvimento podendo haver a necessidade de se proceder a alterações ao modelo de dados. Este projeto não foi exceção e, como tal, para realizar essas alterações ao modelo de dados foi usado o processo Code First [34]. Este permite que se façam alterações ao nível das classes C# e que estas sejam aplicadas ao modelo existente na base de dados. Para controlar as alterações efetuadas ao esquema são usadas *migrations* [34], que registam todas as mudanças efetuadas ao nível do código das classes mapeadas. Esta metodologia permite reverter o esquema para determinada *migration* em caso de ser necessário cancelar determinado conjunto de alterações.

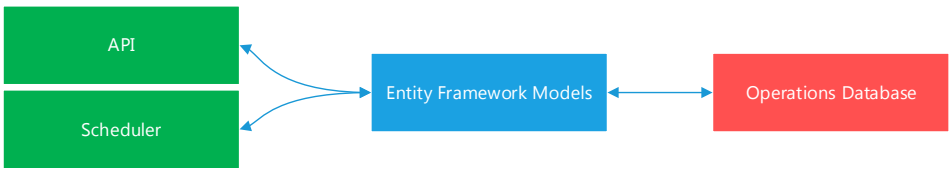

Figura 25 - Entity Framework como ponto de mapeamento da base de dados do sistema

Esta solução utilizando Entity Framework trouxe bastante dinâmica ao projeto, permitindo ter um ponto central de acesso ao modelo de dados tanto para a API como para o Scheduler do sistema.

De seguida será apresentada uma lista com uma breve descrição das tabelas usadas no sistema. O leitor poderá analisar detalhadamente o diagrama de relação de entidades (ER) no ponto 1.4 do anexo C (documento de arquitetura) para compreender as relações entre as várias tabelas e as colunas usadas em cada.

Tabela	**Descrição**
Alert	Alarmes associados a determinado utilizador.
Client	Clientes nos quais foram efetuadas intervenções.
Collaborator	Colaboradores da ISA com tarefas associadas nos projetos monitorizados.
Intervention	Instalações/manutenções feitas por subcontratado a dado cliente.
Invoice	Faturas emitidas para determinado subcontratado.
Issue	*Issues* do JIRA associados a determinado colaborador para um dado projeto.
Note	Notas do sistema em que o utilizador pode definir pequenos lembretes ou anotações relativamente a dados da aplicação.
Notification	Notificações geradas por alertas definidos pelos utilizadores.
Payment	Registo associado ao pagamento de determinada fatura a um subcontratado.
Project	Informação relativa a projetos do JIRA.
Technician	Subcontratados que efetuaram intervenções em clientes.
User	Utilizador do sistema mapeado a partir do sistema LDAP da empresa.
WorkLog	Registos associados a *issues* que permitem o controlo sobre o tempo despendido nessas mesmas tarefas.

Tabela 17 - Descrição de entidades presentes no diagrama ER

5.4.2. Operations API

A API do sistema foi totalmente desenvolvida em ASP.NET Web API 2 [35]. Trata-se de uma plataforma para construção de serviços RESTful utilizando a *framework* .NET e que permite o desenvolvimento bastante rápido de um serviço HTTP para fornecer dados ao *front-end* do sistema. Essencialmente, é esta a principal função desta API – responder a pedidos HTTP provenientes da *web application*. A distribuição deste sistema para produção é feita através de um servidor IIS, tipicamente numa máquina diferente da que distribui a *web application*/*front-end* do sistema.

5.4.2.1. Controlador e Ações

Toda a camada de serviços fornecidos pela API foi dividida em vários controladores de acordo com o tipo de dados/ações que estes têm de fornecer ou tomar. Um controlador funciona nesta plataforma como um contentor de ações em que cada ação corresponde a uma rota. Por

exemplo, existindo o controlador "Projects" e se for definido um método correspondente a um HTTP GET de nome "getAllProjects()", pode-se requisitar esta ação a partir do endereço http://apihost:portnumber/api/projects/getallprojects.

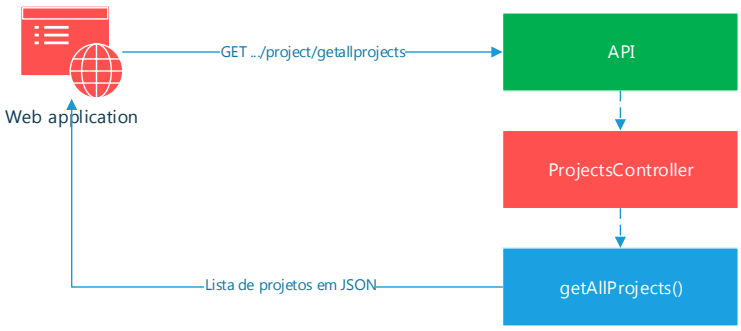

Figura 26 - Exemplo de utilização da API para obtenção de informação de projetos

O fornecimento de dados ao *front-end* é sempre feito no formato JSON, no entanto, este não é o formato por defeito da Web API 2. Uma vez que JSON é cada vez mais estandardizado e fortemente incutido pela *framework* AngularJS foi removido o formato XML usado por defeito na Web API 2, ativando automaticamente a distribuição do conteúdo em JSON.

5.4.2.2. Retorno de Objetos

Outra particularidade da Web API 2 é a possibilidade de retornar objetos de classes C# (POCO – Plain Old CLR Object [36]), sendo que estes são automaticamente transformados no formato JSON. Esta é uma propriedade bastante explorada ao longo de todo este projeto, podendo ser criadas classes específicas que são retornadas como resultado em cada uma das ações dos controladores. Continuando com o exemplo do pedido por uma lista de projetos, é efetuado um HTTP GET para a rota "/getallprojects" que corresponde à seguinte estrutura de exemplo:

```
public class ResultProjects(){

        public List<string> projectsList = new List<string>();

        public int numberOfProjects = 0;

}

[HttpGet]

public object getAllProjects(){

        ResultProjects result = new ResultProjects();

        result = getProjectsLogic();

        return result;

}
```

É primeiramente definida uma classe `ResultProjects` que servirá como um contentor de informação a ser retornada. Na ação em si[i], é criada uma instância da classe especificada e é preenchido este objeto com os dados necessários. A Web API permite que se faça o retorno imediato desse objeto e, de forma transparente para o programador, este é convertido para JSON e é transmitido para a *web application*. Este é um *design* adotado por *frameworks* como a Service Stack [37], muito semelhante a Web API 2, que permite uma melhor organização do

[i] Requisitada pela rota "http://apihost:portnumber/api/projects/getallprojects"

código, criando uma solução mais limpa, fornecendo contentores de dados bem estruturados para acesso[i] na *web application.*

5.4.2.3. CORS

Como representado na figura abaixo, todos os controladores desta plataforma descendem de um BaseController. Esta decisão prende-se com a necessidade de ativar o mecanismo CORS [38] para que as conexões entre o *front-end* e *back-end* do sistema sejam permitidas. Sem este mecanismo ativo a comunicação entre a *web application* e a API do sistema não seriam autorizadas pelo *web browser,* uma vez que a API é fornecida a partir de um domínio diferente do domínio origem dos recursos da *web application.* Na verdade as máquinas que distribuem os dois serviços[ii] até podem estar na mesma rede (mesmo domínio), neste caso a *intranet* da ISA, mas foi verificado que a própria definição de máquina é considerado pelo *web browser* como um domínio diferente, bloqueando as comunicações e obrigando ao uso de CORS.

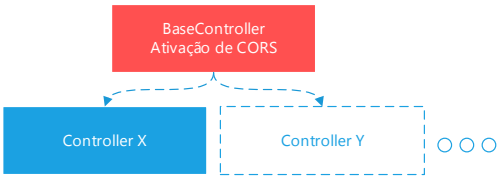

Figura 27 - Uso de um controlador "pai" para ativação de CORS

5.4.2.4. Controladores da Web Application vs Controladores da API

Existe uma relação direta entre os controladores da *web application* e a API do sistema, tal como é apresentado na figura seguinte.

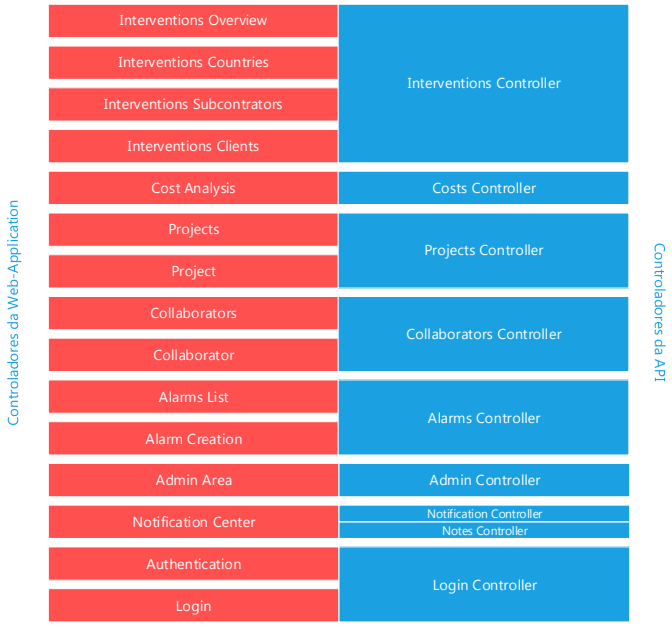

Figura 28 – Relação entre controladores da *web application* e controladores da API

[i] Acesso ao objeto JSON resultante da transformação do POCO.
[ii] Servidor IIS que fornece a API e servidor IIS que distribui a *web application*.

Na tabela seguinte é apresentada uma breve descrição dos principais controladores da API e a relação existente com a *web application*.

Controlador	**Descrição**
Interventions Controller	Fornece aos controladores da *web application* especificados dados para construção de gráficos e apresentação de indicadores relacionados com intervenções, subcontratados e clientes.
Costs Controller	Fornece ao controlador Cost Analysis da *web application* dados relativamente a faturas e pagamentos a subcontratados.
Projects Controller	Retorna dados relativamente a projetos do JIRA e a colaboradores que participam nesses projetos.
Collaborators Controller	Fornece aos controladores da *web application* especificados dados relativamente a colaboradores.
Alarms Controller	Permite a criação/eliminação de alarmes ou requisição da lista de alarmes definidos por determinado utilizador.
Admin Controller	Fornece à *web application* os mecanismos para alteração do ciclo de refrescamento de dados do Scheduler do sistema e permite a requisição da lista de *logs* da API e Scheduler.
Notification Controller	Controlador que fornece a lista de notificações gerada por alertas definidos pelo utilizador. Permite também que a *web application* faça pedidos para eliminação ou marcação como lida de determinada notificação.
Notes Controller	Fornece à *web application* a criação, eliminação e listagem de notas de determinado utilizador.
Login Controller	Disponibiliza o conjunto de todos os mecanismos de autenticação, segurança e validação de permissões ou rotas na *web application*.

Tabela 18 - Controladores usados na API do sistema

Estes controladores são ainda apoiados por um conjunto de processos e bibliotecas externas com a finalidade de processar datas, fazer a desencriptação de dados, filtrar resultados, entre outros. Alguns desses processos serão abordados no capítulo de implementação.

5.4.3. Operations Scheduler

O Scheduler do sistema trata-se de um serviço desenvolvido na linguagem C# assente na *framework* Quartz.NET [39] e que é corrido na forma de uma *console application* a partir de um ficheiro executável do Windows. Esta plataforma foi, de uma forma clara, desenvolvida para responder a dois conjuntos de necessidades, que se prendem ambas com o facto de tentarem atingir objetivos dependentes de agendamento. A primeira prende-se com a necessidade de recolher dados periodicamente dos três sistemas[i] que fornecem informação à Operations Dashboards. A segunda tem que ver com a criação de notificações para os utilizadores a partir dos alarmes definidos. Ambas carecem de execução periódica e bem coordenada e, como tal, foi decidido usar uma solução que permitisse fazer o agendamento e execução automáticas do conjunto de tarefas necessárias ao processo. Para tal, foi usada a Quartz.NET onde são definidas várias *jobs*[ii] cada uma com um *trigger*[iii] associado. Os *triggers* destas *jobs* são configurados a partir de datas ou intervalos de tempo.

5.4.3.1. Ciclo de Vida de uma Job

Quando um *trigger* é ativado, ou seja, quando se atinge a hora ou o intervalo de tempo definidos, é imediatamente iniciada a *job* correspondente a esse mesmo *trigger*. No momento

[i] Primavera ERP, MaiGás e JIRA.
[ii] Processos paralelos ao processo original do Scheduler que têm como finalidade executar periodicamente um conjunto de ações definidas programaticamente.
[iii] Objeto que monitoriza constantemente o agendamento de uma *job* e, caso esse agendamento se verifique, é iniciada a *job* associada a esse *trigger*.

em que a *job* conclui todo o seu processo, esta é destruída e o *trigger* retoma o estado de desativado. Apesar do tempo de execução de uma *job* poder ser variável, a próxima ativação do *trigger* ocorrerá exatamente à hora ou intervalo de tempo definido desde a primeira execução. Isto quer dizer que se existir um *trigger* de hora a hora e a *job* correspondente por alguma razão demorar mais de uma hora a concluir, ao início da segunda execução existirão duas *jobs* a executar o mesmo tipo de trabalho. Este assunto foi debatido no decorrer deste projeto e foi tomada a decisão de impedir o arranque de *jobs* do mesmo tipo concorrentemente. Para isso foi usada a propriedade `[DisallowConcurrentExecution]` para cada um dos diferentes tipos de *jobs* necessárias ao projeto. Este processo segue o ciclo definido na figura apresentada abaixo.

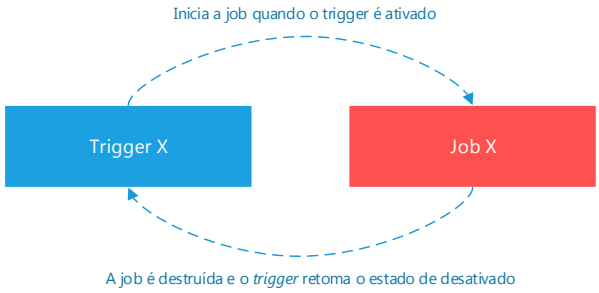

Figura 29 - Ciclo de vida de uma *job* sem concorrência

Apesar de não existirem *jobs* com períodos tão próximos entre execuções no atual projeto, este era um ponto que interessava garantir no início do projeto, com vista a não existir concorrência em *jobs* iguais na modificação simultânea dos mesmos tipos de dados. Esta preocupação prendia-se com o facto de ter sido inicialmente planeado que o sistema iria refrescar os dados em intervalos de poucas horas. No entanto, com o avançar do projeto percebeu-se que tal gasto de *performance* não era necessário uma vez que os dados, essencialmente, de *work logs* no JIRA, pagamentos e faturas no Primavera só eram registados nas plataformas no fim do dia de trabalho.

5.4.3.2. Tipos de Job

Tal como a figura abaixo apresenta, foram definidos dois grupos diferentes de *jobs* diretamente relacionados com as necessidades supramencionadas.

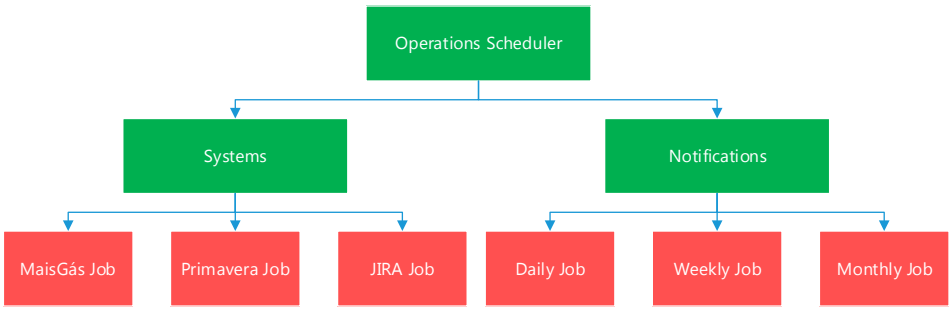

Figura 30 - Jobs definidas no Scheduler do sistema

O primeiro grupo define as *jobs* relacionadas com a obtenção automática de dados dos sistemas necessários ao processo, enquanto o segundo prende-se com a realização de tarefas como a criação de notificações e envio de *emails* aos utilizadores quando os seus alarmes forem ativados. De seguida será dada uma breve descrição sobre cada uma das *jobs* definidas.

5.4.3.3. Systems Jobs

Antes de mais é necessário esclarecer o porquê do uso de *jobs* para armazenar informação já existente nas plataformas necessárias ao processo. Esta necessidade prende-se com uma só razão – performance. O acesso e processamento de dados em tempo real, ou seja, com a requisição de informação por parte da *web application* a ser feito diretamente à base de dados do Primavera ERP, MaisGás ou JIRA, trataria tempos de resposta absolutamente incomportáveis e custos elevados de processamento aquando a obtenção destes dados. Ainda que existisse um servidor com uma API que tratasse de todo esse processo pela *web application* o problema mantinha-se.

Podem-se verificar de seguida os tempos médios atingidos pelas *jobs* na recolha de dados de cada um dos sistemas de forma a concluir o raciocínio apresentado acima.

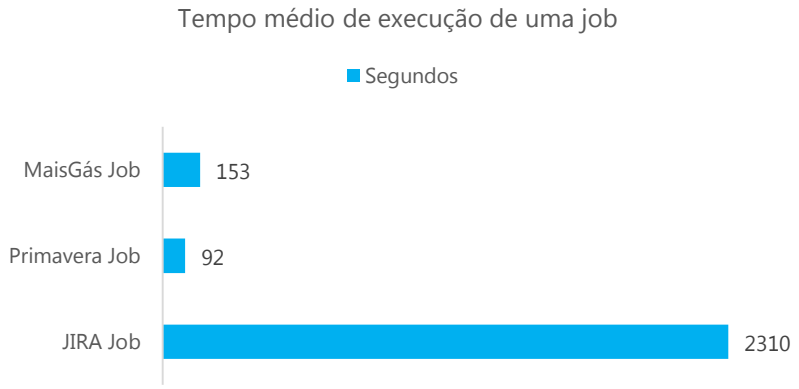

Figura 31 - Tempo médio de execução das jobs de recolha de dados

É importante referir que estas *jobs* correm em horários diferentes, descartando a possível influência de carga de processamento de uma *job* para outra. Esta decisão parte do princípio que existem dependências dos dados, em especial a informação de subcontratados recolhida do MaisGás necessária para obter faturas e pagamentos dos mesmos no Primavera ERP.

A solução mais coerente e lógica para este sistema foi, como vimos ao longo de todo este documento, Esta *job* diária, com horário passível de ser definido pelo administrador do sistema, é responsável por recolher informação de todos os projetos existentes no sistema JIRA. Este processo é desempenhado a partir da comunicação direta desta *job* com a API REST desta plataforma de gestão de projetos. É usada a biblioteca RestSharp [40] para executar os pedidos à API em questão e a biblioteca Json.NET [41] para decompor e analisar os objetos JSON resultantes dos pedidos efetuados.

O processo pode ser resumido, de forma simples, em três passos. Primeiramente são requisitados todos os projetos inseridos no sistema. De seguida, são pedidos todos os *issues* de cada projeto. Por fim, cada *issue* é processado individualmente e são requisitados todos os *work logs* associados ao mesmo. Este último passo prende-se com o facto de ser necessário estabelecer uma linha temporal dos registos de trabalho efetuados em projetos e por colaborador. A informação devolvida num *issue* fornece o número total de horas registadas, no entanto, para garantir que é analisada a repartição dessas horas registadas é necessário armazenar cada um dos *work logs* desse *issue*.

Para além de a comunicação com a API do JIRA ser extremamente lenta, por não ser uma API exclusivamente dedicada a fornecer grandes quantidades de dados, todos os registos relativamente a projetos, colaboradores, *issues* e *work logs* na Operations BD são apagados diariamente na atualização feita por esta *job*. Esta é uma operação estritamente necessária para

manter estes dados confiáveis. Este processo é efetuado devido à inexistência de identificadores únicos nos *issues* ou nos próprios *work logs* do JIRA quando estes são fornecidos pela sua API REST. Assim, torna-se uma tarefa praticamente impossível a de sincronizar fiavelmente estes dados, uma vez que o próprio sistema JIRA permite aos seus utilizadores a alteração de praticamente todos os campos de *issues* e *work logs*. Este último ponto engloba a alteração de datas de *work logs,* número de horas registadas num *work log* ou, ainda mais grave, a alteração do colaborador associado ao *issue*. Desta forma é possível compreender que a manutenção deste tipo de dados seria certamente inconsistente e poderia apresentar resultados erróneos. Contudo, a comunicação extremamente lenta com a API do JIRA acaba por ser o fator mais negativo para os longos tempos de execução desta *job*. Apresentando um exemplo para a análise de um *issue*, cerca de 85% do tempo de análise do mesmo é gasto na obtenção do objeto JSON proveniente da API do JIRA. O restante tempo é despendido na decomposição do objeto, na criação do novo objeto Issue[i] e no armazenamento na base de dados.

5.4.3.4. MaisGás Job

O MaisGás *job* trata-se de outra *job* diária com a principal responsabilidade de recolher informação relativamente a intervenções, subcontratados e clientes diretamente da base de dados do MaisGás. Na realidade são três os motores SQL Server do MaisGás a que esta *job* se tem de ligar. Esta dependência deve-se ao facto de a base de dados deste sistema ser distribuída por vários servidores.

O funcionamento desta *job* difere um pouco da *job* do JIRA. Nesta são criadas, inicialmente, três conexões a três bases de dados diferentes usando a biblioteca nativa SQLClient [42] da *framework* .NET e são efetuadas *queries* SQL de modo a obter registos de intervenções. É a partir destas intervenções que se obtém também a informação dos subcontratados e clientes a que as mesmas estão associadas.

A partir desta *job*, é possível manter os dados de intervenções da Operations Database atualizados. Existem identificadores únicos para cada intervenção, permitindo atualizar os campos modificados entre execuções da *job*. Por exemplo, se o estado de uma intervenção passar de "pendente" para "realizada"[ii] durante o dia de trabalho, na execução seguinte desta *job*, a intervenção em questão será atualizada. Para isso foi definido um processo que analisa cada intervenção minuciosamente, ao ponto de verificar se ocorreram também alterações ao subcontratado ou ao cliente associado, procedendo às devidas correções. Assim, se existir uma intervenção a ser efetuada pelo subcontratado X (não tendo este mais nenhuma intervenção associada) e durante o dia de trabalho essa intervenção tiver sido vinculada ao subcontratado Y, na execução seguinte da *job* será apagado o subcontratado X da base de dados e é adicionado o subcontratado Y (caso este não exista). Esta abordagem pode ser seguida porque cada intervenção possuiu, como já referido, um identificador único, no entanto, esse facto não cobre a possibilidade de uma intervenção ser eliminada podendo levar à apresentação de dados incorretos. Na verdade uma intervenção nunca é eliminada neste sistema. No caso de ser cancelada, o campo numérico para o estado da intervenção passa a ser 0, levando a que a informação dessa intervenção deixe de ser considerada nos dados a apresentar.

5.4.3.5. Primavera ERP Job

A *job* de recolha de dados do Primavera ERP funciona segundo a lógica apresentada para a *job* do MaisGás. É estabelecida a conexão a uma só base de dados onde são feitas *queries* no sentido

[i] A partir do Modelo de Dados mapeado para Operations BD.
[ii] O estado da intervenção é identificado com um campo numérico de 0 a 5.

de obter dados relativamente a faturas e pagamentos de subcontratados. Esta é uma das principais razões pela qual a *job* do MaisGás tem de executar sempre primeiro – é necessário existir informação de subcontratados para que se possam pesquisar faturas associadas aos mesmos.

Nesta *job,* um dos principais desafios foi a de incluir na pesquisa de faturas do Primavera o nome dos subcontratados já existentes na Operations Database. Isto deve-se ao facto de o nome dos subcontratados poder não ser exatamente igual ao existente na base de dados Primavera. Como tal, antes de se efetuarem as *queries* necessárias são levadas a cabo algumas operações para retirar acentos das palavras, abreviaturas, entre outros tipos de palavras que levavam a pesquisas inconclusivas. Além disso, são muitas vezes introduzidas faturas ligadas aos técnicos de determinado subcontratado e não à própria empresa subcontratada, levando à necessidade de pesquisa de faturas por cada técnico para que se possa fazer essa associação.

Outro dos grandes desafios da integração com este sistema foi fazer a ligação de faturas para os pagamentos em si. Na realidade não existe qualquer identificador que relacione estas duas tabelas, como tal, foi feita correlação através do valor da fatura e o identificador da entidade associada ao pagamento (subcontratado). Desta forma existe a possibilidade de neste ponto se recolherem dados incorretos. Tendo em conta determinado pagamento de valor X para a entidade A associado a uma fatura Y, facilmente o leitor pode identificar que se for criada uma nova fatura Z de valor X para a mesma entidade, seguindo este processo está a ser recolhida informação de um pagamento (para a fatura Z) que ainda pode não ter ocorrido. Para colmatar este problema são tidas em conta informação de datas. Em cada fatura existe o prazo de pagamento da mesma, analisando-se, para o caso apresentado acima, se o pagamento se encontra dentro desse mesmo período, de forma a evitar o registo de pagamentos que ainda não foram feitos e que têm o mesmo valor pesquisado. É claro que este é um método que não pode garantir fiabilidade na totalidade, no entanto, para o caso em estudo[i] até a presente data do relatório, todos os dados apresentados foram garantidos como corretos ao serem analisados um a um manualmente. Importa também referir que normalmente é emitida uma fatura por mês para cada subcontratado ativo e que o prazo limite de pagamento é de 60 dias.

O processo de atualizações de dados segue o mesmo princípio da *job* do MaisGás. Uma vez que cada fatura e cada pagamento possui um identificador único, é possível comparar as alterações em cada execução e fazer a sua atualização diretamente na Operations Database.

5.4.3.6. Notifications Job

O conjunto de *jobs* associado a notificações efetua o processo definido mas com periocidade diferente. Apesar disso, são definidos três *triggers* diferentes que levam à criação de três instâncias descendentes da mesma *job* "pai". Para dentro da execução destas *jobs* é passado apenas o indicador da sua periodicidade[ii] de forma a analisar apenas os alarmes que lhe correspondem. Estas *jobs* podem executar em paralelo e foram definidas para executar às 8 da manhã de acordo com a sua periodicidade.

O facto de se permitir que o utilizador crie alarmes para monitorizar diariamente, semanalmente ou mensalmente determinado indicador, levou à decisão de se criar estas três *jobs* idênticas. Se tal não fosse feito iria ser necessário existir um mecanismo para controlar e monitorizar se determinado alarme já ativou notificações durante a atual semana ou mês, trazendo algumas dificuldades ao processo. Usando o processo escolhido, é garantido que determinado alarme semanal ou mensal irá enviar uma única notificação[iii] nesse período de

[i] Dados com data superior a janeiro de 2013.
[ii] Diária, semana ou mensal.
[iii] Se as condições para ativar o alarme se demonstrarem verdadeiras.

tempo pois as *jobs* responsáveis só irão executar, como é claro, semanal e mensalmente. Na definição dos *triggers* que ativam estas *jobs* é incluído o agendamento que desencadeará na verdade a ativação destas *jobs:*

- `CronScheduleBuilder.MonthlyOnDayAndHourAndMinute(int dayOfMonth, int hour, int minute))`
- `CronScheduleBuilder.WeeklyOnDayAndHourAndMinute(int dayOfWeek, int hour, int minute)`
- `CronScheduleBuilder.DailyAtHourAndMinute(int hour, int minute))`

Este último ponto é particularmente usado na definição dos *triggers* associados às *jobs* diárias do MaisGás, Primavera ERP e JIRA.

Relativamente ao processo de análise de alarmes para geração de notificações, este é feito recorrendo à própria Operations API. Para cada alarme, a *job* responsável irá fazer um pedido a um controlador[i] da API enviando a informação desse mesmo alarme. Neste controlador é analisado o tipo de alarme, as condições impostas e é devolvida uma resposta à *job* em espera. Uma vez que todo o processamento de informação para obter indicadores ou evoluções de indicadores é feito na API do sistema, decidiu-se manter essa lógica constante também aqui neste ponto. A comunicação da *job* para a API é feita usando novamente RestSharp sendo que as respostas obtidas da API levam à criação ou não de notificações. Estas notificações são especificadas por defeito para cada um dos tipos de alarme, em que são apenas introduzidos as condições variáveis que acionaram o alarme. Estas notificações são imediatamente disponibilizadas na *web application* e, no caso de o utilizador associado ao alarme em questão possuir o serviço de *email* ativo, é também enviada a mesma notificação através deste meio.

[i] AlarmProviderController

Capítulo 6
Implementação

No presente capítulo pretendem-se apresentar algumas das principais decisões tomadas ao nível da implementação da *web application* e das interações com a API do sistema de forma a atingir um produto robusto e ao mesmo tempo capaz de responder às necessidades dos seus utilizadores. Serão abordados individualmente aspetos ao nível de segurança, apresentação de dados e exportação de conteúdo do sistema Operations Dashboards.

6.1. Design e Responsividade

O *design* da *web application* tem os alicerces montados no conhecido Bootstrap [43]. Esta biblioteca de CSS e JavaScript trouxe para a *web application* excelentes mecanismos de disponibilizar conteúdo, de organizar o próprio *layout* da aplicação mas, também, define o estilo para praticamente qualquer necessidade do programador ao nível de visual da página. A partir desta estrutura foram feitas várias alterações de modo a produzir um *design* mais específico para o âmbito deste projeto. Foram escritas cerca de mil linhas de código CSS em que a maior parte são *override* ao estilo definido originalmente pelo Bootstrap.

A utilização desta poderosa biblioteca veio também facilitar um dos pontos importantes de qualquer *web application* moderna – a responsividade do *layout*. Esta prende-se com a capacidade que uma *web application* tem em se adaptar visualmente sem que o tamanho de ecrã do dispositivo em que corre interfira. Todo o *layout* da Operations Dashboards foi construído sobre este princípio, tendo havido o cuidado de definir corretamente o comportamento de cada painel em cada *dashboard*.

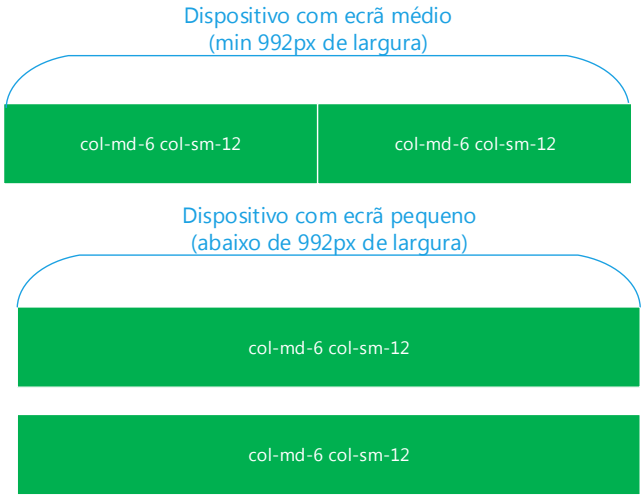

Figura 32 - Exemplo de uso do Bootstrap para se obter um *design* responsivo

A figura acima representa um exemplo da definição da estrutura de dois painéis responsivos. Por convenção o Bootstrap considera a largura total de uma zona de uma página com o número 12. Ao se definir uma `<div class="col-md-6">` tem-se que esta zona em questão ocupará metade da `<div>` pai[i] quando o dispositivo que corre a aplicação tiver no mínimo 992 pixéis de largura. O mesmo se aplica à `<div>` do lado direito da imagem, definindo a mesma classe, esta ocupará a outra metade da página. No entanto, pode-se verificar que é também

[i] Neste caso é o HTML `<body>` .

definida outra classe em cada uma das `<div>`. A classe `col-sm-12` especifica que quando o dispositivo tiver um ecrã inferior a 992 pixéis esta `<div>` deverá ocupar toda a largura da página. O mesmo se aplica à `<div>` que fica à sua direita em ecrãs maiores. Esta passa estende-se em toda a largura da página ficando por baixo da primeira `<div>` definida.

É muito importante manter todo este processo coerente ao longo do desenvolvimento de interfaces de modo a se obter um *layout* percetível no máximo de dispositivos possíveis. Podem usadas as classes responsivas `col-xs` ou `col-lg` para definir a apresentação de conteúdo em dispositivos com ecrãs inferiores a 768 pixéis de largura ou superiores 1200 pixéis respetivamente. Ao longo deste projeto foram sempre usadas as classes `col-sm` e `col-md` por conseguirem responder corretamente a qualquer tipo de ecrã. Importa referir que a aplicação foi testada em vários *web browsers* de dispositivos móveis como iPad, iPhone ou Asus Eee PC, mas também em ecrãs de grandes e médias dimensões. Todos os testes levaram à correção de falhas encontradas no reajustamento dos painéis conseguindo-se, no final, obter uma aplicação capaz adaptável a todos esses tipos de equipamentos.

A nível da *side bar e nav bar* da aplicação, ao ser detetado que o dispositivo tem um ecrã inferior a 768 pixéis, esta também se confina numa só barra clicável. Ao clicar nesta barra responsiva serão apresentados os itens que constituíam as duas barras da aplicação.

As figuras seguintes representam pequenos excertos do aspeto visual do sistema desenvolvido e da responsividade dos painéis para diferentes tipos de dispositivos.

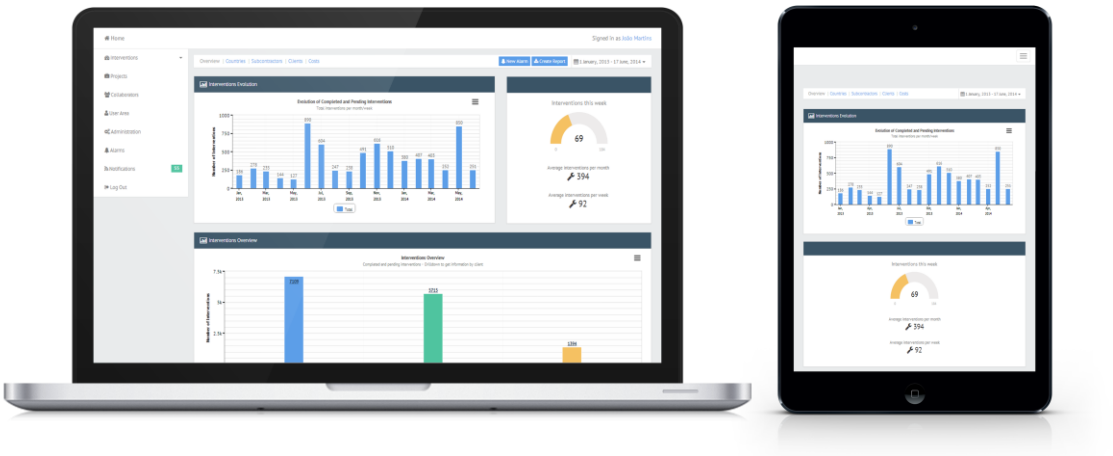

Figura 33 - *Dashboard* de intervenções da *web application* versão desktop e mobile

Figura 34 - *Dashboard* de progresso de um colaborador

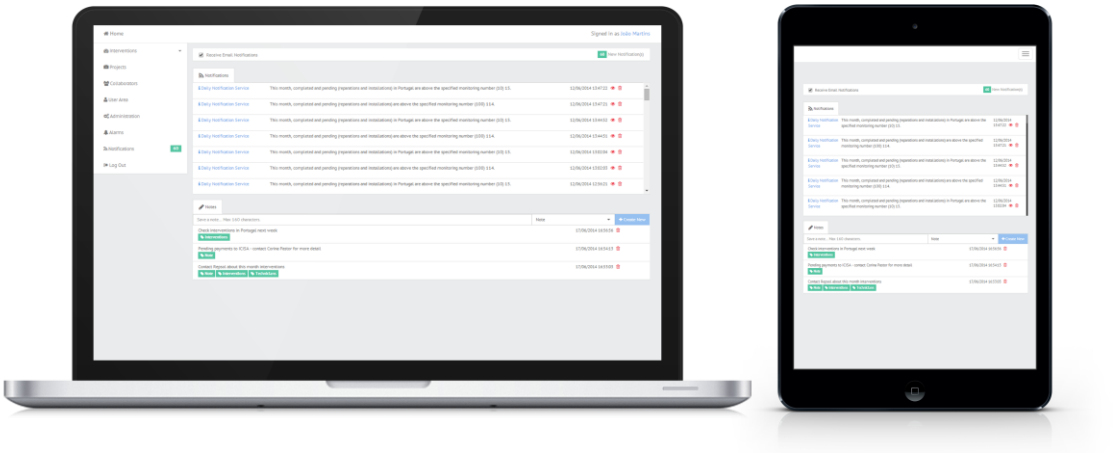

Figura 35 – Central de notificações e sistema de notas

6.2. HighCharts

É notável que toda a Operations Dashboards gira à volta da apresentação de dados. Uma das melhores representações possíveis de grandes quantidades de dados, especialmente dados que evoluem ao longo do tempo, é através de gráficos. Foram inicialmente estudadas várias propostas de bibliotecas JavaScript para criação de gráficos no contexto da *web application* (*cliente-side)*:

- Charts.js - http://www.chartjs.org/
- jqPlot - http://www.jqplot.com/
- ZingChart - http://www.zingchart.com/
- D3js - http://d3js.org/

No entanto, rapidamente ferramenta que chamou mais à atenção, tanto por ser completamente gratuita, mas principalmente por permitir um nível de personalização inigualável, foi a biblioteca HighCharts [44]. Mais vantagens existiam quando se verificou que era a única (em fevereiro de 2014) que possuía uma *directive open-source* para AngularJS – a highcharts-ng [45]. Esta *directive* permite definir rapidamente, no corpo de uma página HTML, um novo gráfico usando simplesmente a *tag* `<highchart id="chartX" config="chartXConfig" ></highchart>`. Este gráfico, depois de definidas as propriedades do mesmo num *array* `chartXConfig[]` no controlador dessa mesma página, passa a ser apresentado na página e passível de ser alterado em *runtime* da forma que se pretender que todas as modificações serão apresentadas automaticamente. Por exemplo, se um gráfico estiver a apresentar determinado conjunto de valores e for feita uma nova injeção de dados, todo o processo de recarregamento de dados é transparente ao programador. Usando apenas a biblioteca JavaScript original seria necessário efetuar um conjunto de operações de modo a criar uma nova instância do gráfico e a iniciar essa mesma instância. Para além desta *directive* trazer a simplificação na criação e atualização de gráficos, estabelece também uma melhor organização do código pois não existe a necessidade de criar métodos para iniciar ou atualizar esses mesmos gráficos. Simplesmente são alteradas as propriedades desejadas no *array* de configurações e a própria *directive* irá tratar de alocar os recursos novamente.

Foi criado um tema[i] que se aplica a todos os tipos de gráficos usados ao longo do projeto. Ao tema foi aplicada uma palete de cores específica, assim como o tamanho e tipo de letra que

[i] Conjunto de estilos CSS.

são usados no resto do projeto. Esta necessidade prendeu-se com o facto de ter sido pedido pelos *stakeholder*s que os gráficos fossem coerentes com o resto da interface da *web application*.

6.2.1. Carregamento de Dados

O carregamento dos dados apresentados em cada os gráfico é feito com recurso à Operations API. É a partir do controlador da página em que determinado gráfico se insere que são efetuados os pedidos pela informação a disponibilizar. Tipicamente estes dados são devolvidos pela API sob a ordem pela qual têm de ser disponibilizados no gráfico em questão, sendo apenas necessário formar um *array* com esta informação, que será injetado na propriedade `series` do gráfico.

Figura 36 - Exemplo de pedido efetuado para obtenção de dados que alimentam um gráfico

Os métodos dos controladores da *web application* responsáveis por fazer os pedidos à API são, normalmente, chamados *onDocumentReady,* ou seja, quando estão carregados todos os elementos do DOM, como é o caso do método getChartXData() apresentado como exemplo na figura acima. Estes podem também ser invocados para renovar dados dos gráficos correspondentes – este processo está normalmente associado ao uso dos *date filters* da aplicação, ou da troca de período em análise[i].

6.3. Datas

Para cada *dashboard* é definido um *date filter* com vista à possibilidade de o utilizador especificar o intervalo de dados apresentado. É definido para cada *date filter* um objeto *listener* que ficará responsável por detetar alterações ao intervalo de dados escolhido. Este intervalo é por defeito desde o primeiro dia do ano corrente até à presente data. Ao ser detetada uma modificação ao intervalo são imediatamente invocados todos os métodos existentes nesse controlador que são responsáveis por recolher dados da Operations API. Desta forma, ao ser modificado o intervalo de dados o utilizador poderá analisar de uma vez todos gráficos nesse novo intervalo e retirar possíveis conclusões sobre a influência entre os vários indicadores presentes no *dashboard*.

Todo o sistema desenvolvido tem a particularidade de estar bastante dependente de datas. No entanto, especialmente ao nível de informação de intervenções, estas incluem unicamente a semana do ano, o mês e o ano em que a mesma foi ou irá ser realizada. Apesar de ser usada a biblioteca Moment.js [46] para gerir datas na *web application*, ao serem efetuados os pedidos à API pelos dados para determinado gráfico, são sempre passadas as datas completas[ii] do intervalo em questão[iii]. Isto permite que, no lado da API, sejam feitas as operações necessárias, usando a biblioteca Time Period .NET [47], para determinar as semanas do ano do intervalo requisitado e para analisar cada intervenção de forma a verificar se esta se encontra na extensão desse mesmo intervalo. Este último ponto envolve a verificação de se o mês da intervenção

[i] Troca da representação da informação semanal por mensal, ou vice-versa.
[ii] Formato Dia/Mês/Ano
[iii] É feita a conversão num só lado pois as duas bibliotecas usadas (na *web application* e na API) seguem normas diferentes no que toca a cálculo de semanas do ano.

pertence ao intervalo e, caso este se verifique, é finalmente validado se a semana do ano dessa intervenção intersecta também o intervalo. São tidas em conta estas verificações pois existe a possibilidade de determinada semana pertencer a dois meses diferentes, fornecendo resultados incorretos caso um desses meses não se encontre dentro do intervalo especificado.

Importa também referir que o sistema MaisGás, por defeito, considera que a semana começa ao domingo. Posto isto, para manter a coerência dos dados, foi mantido este princípio e foram calculadas as semanas com início ao domingo, contrariando o que especifica o ISO 8601 [48].

Por fim, ainda relativamente a datas, importa referir que todo o retorno de informação para gráficos com evoluções temporais é feito em milissegundos desde 1970, isto é, todas as datas são, depois de feitas as filtragens, convertidas para esta notação. Este facto tem dois propósitos:

- Colher totais semanais/mensais da informação pretendida usando simples objetos `KeyValuePair`. Para cada semana ou mês do intervalo requisitado tem-se um objeto com uma chave (data em milissegundos) e o total semanal/mensal como o valor dessa chave. É feita a criação de uma lista com este tipo de objetos formando o que se pode chamar de um dicionário, onde a atualização dos valores é feita com acesso direto à chave do objeto;
- Inserir os dados em gráficos HighChart de forma universal para qualquer tipo de datas processadas.

6.4. Exportação de Dashboards

Existem duas formas de exportar conteúdo apresentado nos *dashboards*. O primeiro está diretamente relacionado com a biblioteca HighCharts em conjugação com o *plugin* exporting.js. Esta associação permite que o utilizador exporte individualmente cada um dos gráficos HighCharts apresentados no *dashboard* para ficheiro .pdf, .png, .jpeg e .csv. A segunda forma de exportar conteúdo é através de mecanismos desenvolvidos com o intuito de exportar para ficheiro .pdf toda a informação relevante do *dashboard* em causa. Estes mecanismos envolvem a utilização da biblioteca jsPDF [49] juntamente com canvg [50]. Com a primeira é possível definir a localização pixel a pixel que cada elemento a inserir irá tomar numa página do ficheiro .pdf. Este é um processo bastante manual tendo até que se definir quando é necessário acrescentar uma nova página ao ficheiro, obtendo-se uma estrutura bastante dependente do que foi definido inicialmente. A segunda biblioteca permite converter os gráficos existentes no *dashboard* de SVG para canvas [51]. É realizada esta operação no sentido de usufruir do mecanismo de conversão de canvas para imagem disponibilizado pela biblioteca jsPDF. Assim, torna-se possível não só introduzir dados numéricos ou listas com texto no .pdf a exportar, mas também incluir os próprios gráficos apresentados no *dashboard*.

Como se pode perceber os mecanismos de exportação definidos para cada um dos *dashboards* são praticamente não reutilizáveis uma vez que a biblioteca jsPDF só permite definir estruturas bastante específicas. Contudo, esta foi a melhor solução *client-side* encontrada para criação de ficheiros .pdf à medida do programador.

6.5. Autenticação e Segurança

6.5.1. Primeiro Acesso e Permissões

A autenticação na aplicação é feita com o uso de qualquer conta associada ao serviço Active Directory da empresa. *A priori* não existe qualquer conta de utilizador registada na Operations Database. No entanto, são definidos, *hardcoded,* no ficheiro web.config da Operations API os utilizadores com permissões mais elevadas e o seu respetivo nível de permissão[i]. Isto é feito uma vez que o número de utilizadores com permissões elevadas é bastante reduzido. Os restantes utilizadores não existentes neste ficheiro de configurações à medida que efetuam o primeiro *login* na *web application,* ficam registados na base de dados com o nível 3 de permissões, ou seja, têm apenas acesso ao seu próprio *dashboard* de colaborador. A verificação de que a conta de utilizador existe na Operations Database é sempre feita independentemente de a conta estar ou não definida no ficheiro web.config. Se a palavra-passe e conta de utilizador forem validadas com sucesso junto do serviço Active Directory será então criada a conta na base de dados local e são garantidas as devidas permissões e acesso ao utilizador em espera na *web application*.

6.5.2. Autenticação na Aplicação e Validação de Permissões

Primeiramente, importa referir que a palavra-passe do utilizador, ao se autenticar na aplicação, nunca é guardada na base de dados. A verificação é sempre feita junto do serviço Active Directory da empresa validando ou não a autenticação efetuada.

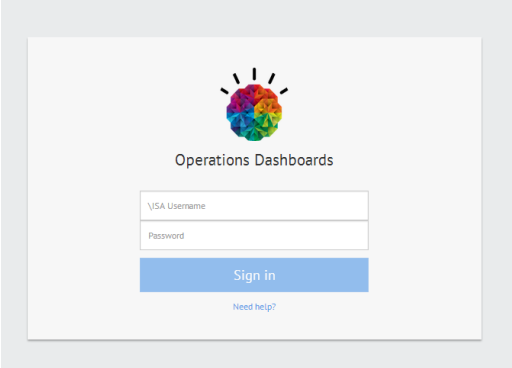

Figura 37 - Página de login da *web application*

Todo o processo de autenticação começa a partir do excerto apresentado na figura acima. O utilizador introduz os dados da sua conta e ao clicar "Sign in" irá desencadear um processo de validação da conta. Primeiro, é usada a biblioteca CryptoJS [52] para fazer a encriptação da palavra-passe através da cifra TripleDES e com o uso de uma *hash* MD5 resultante de uma de uma mensagem secreta comum à *web application* e à Operations API. Esta encriptação é feita com vista a proteger a leitura direta[ii] da informação da conta aquando efetuado o pedido de validação da mesma entre a *web application* e a API.

Ao ser recebida a informação de autenticação encriptada no controlador LoginController da API, a primeira ação passa por desencriptá-la. É usada a técnica inversa ao especificado acima, ou seja, é desencriptada com o desencriptador TripleDES sendo dada a *hash* MD5 da chave secreta partilhada entre as duas partes. Deste processo resulta uma *string* com a palavra-passe

[i] Não implica que os utilizadores fiquem criados na base de dados. Fica apenas especificado que quando estes fizerem *login* pela primeira vez que a estes lhes serão dadas as permissões adequadas.
[ii] No caso de existir a tentativa de *packet-sniffing* na rede interna da empresa.

desencriptada e que é validada juntamente com o nome de utilizador através do serviço Active Directory. É usada a classe nativa PrincipalContext [53] para efetuar a validação segura destes dados junto do servidor Active Directory da empresa. Deste processo é apenas indicado de forma binária à Operations API se a conta é válida ou não.

Como já foi referido na secção anterior, depois de validada a conta é verificado se esta já existe nos registos da Operations Database. Se não existir é criado um novo registo, guardando o nome de utilizador e as permissões adequadas sendo que, de seguida, é sempre efetuado o processo de criação de um *access token*. Este funciona como forma de validação da sessão de um utilizador e tem a validade de 24 horas. Resumidamente, enquanto este for válido ou enquanto o utilizador não iniciar sessão noutro dispositivo, será possível aceder à *web application* sem voltar realizar o *login* no sistema. A criação deste *access token* é feita através da concatenação de um `byte[]` de um identificador único gerado automaticamente e de um `byte[]` da data em milissegundos do momento requisitado. É por fim convertido o *array* resultante para *string* através do método `ToBase64String(byte[])` nativo da *framework* .NET. Daqui resulta um *token* único que é associado ao utilizador em questão.

Feito todo este processo é devolvido à *web application,* em espera pela validação da conta, o *access token* correspondente e as permissões concebidas. Estas propriedades são mantidas em *cookies* de forma a serem acedidas mesmo depois do *web browser* ser fechado. Poderiam ser mantidas no `$rootScope` da aplicação, no entanto, essa informação seria perdida assim que o utilizador visitasse outro *web site*. Se o utilizador aceder a partir de outro *web browser* ou deoutro dispositivo não existirá a informação destes *cookies,* o que leva a uma nova reautenticação na aplicação mesmo que o *token* guardado na base de dados ainda seja válido. Desta forma, será gerado um novo *access token* e a sessão passará a ser válida apenas usando esse mesmo *token.*

Estas propriedades são conservadas de modo a garantir que o utilizador mantém uma sessão válida e a garantir que este tem permissões para aceder a determinada rota. Mesmo que o *cookie* de permissões seja modificado maliciosamente, do lado da API existe sempre a verificação de permissões para cada acesso que o mesmo tente efetuar. Este processo ocorre pois foi definido um controlador na *web application* que, ao ser pedida uma nova página, faz sempre um pedido à API do sistema de modo a verificar se o utilizador tem as permissões indicadas.

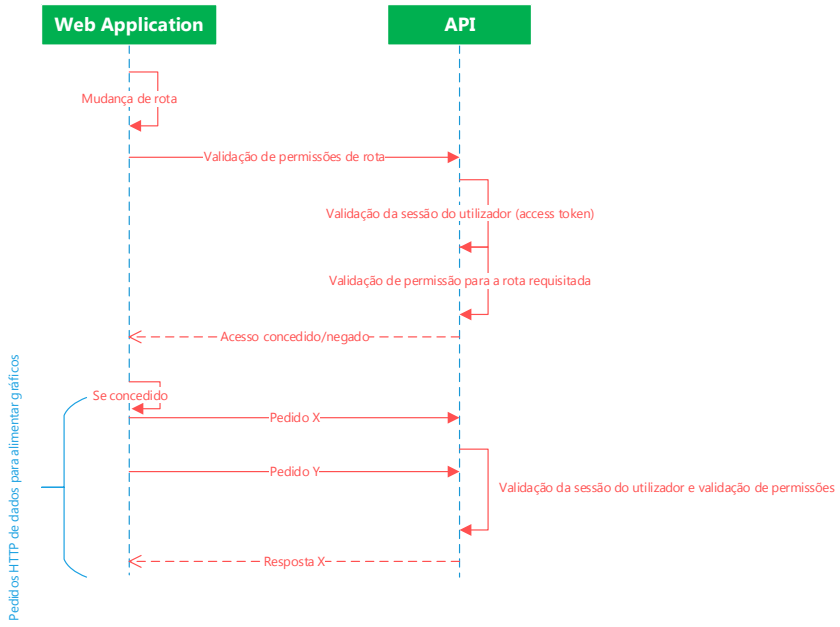

Figura 38 – Validação de rotas e permissões

Para além do que foi referido ao nível da segurança da rota em si, pode-se verificar pela figura acima que estas propriedades têm outro propósito mais ao nível de permissões no acesso a dados. Os pedidos efetuados a rotas da API utilizando um simples HTTP *requester* podem divulgar informação confidencial a utilizadores maliciosos e sem permissões. Desta forma, todas as rotas que divulgam este tipo de dados foram protegidas contra este tipo de acessos indesejados, ao requerem o nome da conta de utilizador e respetivo *access token*.

6.6. Caching

Como secção final deste capítulo, importa também referir que foram utilizados mecanismos de *caching* ao nível da API do sistema. Uma vez que os dados existentes neste sistema se mantêm inalterados durante longas horas no decorrer do dia e alguns pedidos de dados podem ser bastante pesados em termos computacionais para a API, decidiu-se fazer uso de uma biblioteca de *caching* ao nível das ações/rotas dos controladores. Foi usado o projeto *open-source* OutputCache V2 [54] para implementações em Web API 2 de forma a manter em *cache* os resultados de pedidos iguais durante períodos de tempo especificados nas configurações da API. Esta biblioteca também permite adicionar o *header* Cache Control às respostas dos pedidos efetuados especificando, em segundos, o tempo máximo que o *web browser* deve manter em cache a resposta ao pedido efetuado. Todo este mecanismo tem o propósito de libertar carga no servidor mas também diminuir o tempo de resposta total de apresentação de uma página.

Capítulo 7
Controlo de Qualidade

O desenvolvimento e cumprimento de todos os requisitos de um projeto não garante, por si só, a qualidade e sucesso do mesmo. No sentido de validar o cumprimento dos objetivos e expectativas associadas ao projeto, foram conduzidos um conjunto de processos de forma a garantir que o resultado final de todo este percurso está de acordo com as necessidades do cliente (departamento de operações da ISA).

7.1. Testes de Aceitação da Web Application

Uma vez que a *web application* é o ponto central da conjugação de todos os mecanismos implementados ao longo deste projeto, esta aplicação é, no fundo, o item de maior valor para o cliente. Como tal, antes dos desenvolvimentos, foi definido um plano de testes de aceitação de forma a validar as funcionalidades desta parte do sistema. Podem-se consultar no anexo E os testes definidos, juntamente com os resultados esperados no fim de cada teste e a validação ou não dos mesmos. É de referir que o plano de testes final foi conduzido em ambiente de pré-produção pelo próprio estagiário por indisponibilidade da equipa de testes da empresa na altura em que se terminou a implementação de todo o sistema.

Na tabela seguinte pode-se observar um pequeno apanhado dos resultados dos testes efetuados.

	Número de Testes	**Percentagem de Testes OK**
Autenticação	12	100%
Página Inicial e Sistema	8	100%
Área de Notificações	8	100%
Área de Alarmes	2	100%
Área de Administração	3	100%
Dashboards Intervenções e Pagamentos	30	100%
Lista de Colaboradores	2	100%
Dashboard de Projetos	4	100%
Dashboard de Colaborador	2	100%
Total	71	100%

Tabela 19 - Resultados de testes de aceitação executados na *web application*

Pode-se concluir que todos os testes finais foram executados com sucesso, garantindo a fiabilidade dos comportamentos esperados da *web application.*

7.2. Validação com Stakeholders

Ao longo deste estágio, de forma a ir de encontro às necessidades do cliente, foram elaboradas várias reuniões. Nestes compromissos foram validados todos os documentos de especificação do sistema (requisitos e arquitetura) e, ao longo da fase de implementação, foram efetuadas várias sessões com vista a recolher *feedback* do caminho tomado pelo projeto. Especialmente este último ponto permitiu corrigir algumas questões ao nível da usabilidade do sistema e levou à discussão de ideias sobre a forma como disponibilizar determinado conjunto de dados. As várias reuniões de *sprint* com a gestora de projeto engenheira Ana Guimarães também são consideradas um importante ponto na validação do trabalho efetuado, avaliando ao mesmo tempo os riscos e caminhos a tomar nas seguintes iterações, com o intuito de se atingir um

produto final completo e a desempenhar corretamente todas as funcionalidades impostas. Desta forma, há que realçar, de facto, que o forte contributo gerado pelo acompanhamento constante por parte dos *stakeholders* se relevou um importante fator tanto na interpretação dos próprios requisitos associados aos processos manuais seguidos no passado, como no desenvolvimento de funcionalidades à medida das necessidades apresentadas.

7.3. Formação

No fim deste estágio foi efetuada uma sessão para apresentar o trabalho produzido no âmbito do projeto Operations Dashboards. Foi feita uma demonstração de todas as funcionalidades da *web application* junto dos *stakeholders* sob a forma de formação, cobrindo cada um dos requisitos especificados inicialmente para o projeto. Neste contexto, esta sessão serviu também como elemento final de validação dos objetivos cumpridos neste projeto. Todos os requisitos foram cumpridos com sucesso sendo, esta afirmação, corroborada e constatada pelos elementos pertencentes ao departamento de operações da ISA.

7.4. Documentos

Foram produzidos dois documentos finais com especificações relativamente ao sistema desenvolvido. O primeiro determina todos os passos na utilização da *web application* do ponto de vista do utilizador. Este serve, essencialmente, como um guia de utilizador. O segundo trata-se de um registo de todas as etapas necessárias para fazer o *deployment* do *front-end* e *back-end* do sistema. Este especifica também como proceder para alteração de parâmetros estáticos da aplicação e como adicionar ou remover utilizadores com permissões elevadas no ficheiro *web.config* da Operatios API.

Capítulo 8
Conclusões

Dando por concluído o percurso traçado para este estágio, importa tecer algumas considerações tanto ao nível dos objetivos do projeto em si, como ao nível da experiência proporcionada ao autor.

No que toca aos objetivos propostos para este estágio, pode-se afirmar que todos foram concluídos com sucesso. Apesar dos vários entraves, atrasos e mudança de instalações da empresa, foi aplicado um esforço extra para conseguir cumprir todas as expetativas criadas à volta deste projeto. Daqui resultou um sistema capaz de automatizar todo o trabalho que anteriormente era feito de forma manual por vários colaboradores e que apresentava altas probabilidades de introdução de erros. Este sistema consegue, para além de evitar a introdução de erros, poupar cerca de 50 horas de trabalho mensais despendidas pelos vários colaboradores na elaboração do processo de monitorização das operações da empresa. Trata-se de um ganho extremamente importante que traz consigo a comodidade de uma *web application* para acesso aos indicadores que anteriormente eram consultados em folhas de Excel partilhados entre os vários colaboradores do departamento de operações. Ao fornecer um *design* cuidado e responsivo, torna-se simples para os utilizadores finais analisarem os indicadores disponíveis em qualquer dispositivo com *web browser* e em qualquer lugar utilizando a rede VPN da empresa. Desta forma, pode-se concluir, claramente, que esta plataforma traz algo de valor para o cliente, tendo sido manifestado uma enorme estima com o trabalho desenvolvido.

O desenvolvimento deste sistema também permite abrir portas a futuros projetos de aplicações *web* na *intranet* da ISA. Com um Scheduler e uma API capazes de recolher e fornecer dados de três importantes sistemas usados na empresa, estas duas plataformas poderão ser usadas para integrar outras aplicações que necessitem de tipos de dados semelhantes. Na verdade, o sistema desenvolvido encontra-se atualmente a ser usado como estrutura de outro projeto de estágio na empresa. Trata-se de um serviço para gestão e criação de projetos no SVN com suporte direto no JIRA. Este não só irá fazer uso do Scheduler e da API da Operations Dashboards mas irá, também, integrar-se completamente nesta plataforma como um módulo extra. Desta forma, será acedido usando a mesma *web application* através da introdução de uma nova entrada na listagem de funcionalidades da página inicial como está representado na figura abaixo.

Figura 39 - Página inicial da Operations Dashboards

No que concerne às considerações do autor, em relação ao percurso em si, pode-se afirmar, afincadamente, que este estágio proporcionou uma excelente primeira experiência no mercado de trabalho. Os desafios superados possibilitaram a abertura de oportunidades para aumentar o conhecimento técnico, tanto na área de desenvolvimento *web,* como na área de integração de *software*. Os objetivos traçados para este projeto permitiram consolidar conhecimentos previamente estabelecidos, melhorar práticas de desenvolvimento de *software*, mas também aumentar o espectro de tecnologias conhecidas. No entanto, a lição mais importante tirada desta jornada passa pelos conhecimentos adquiridos na gestão de um projeto de *software* e na engenharia que envolve todo esse processo.

Bibliografia

1 Web site da empresa ISA. [Internet]. Disponível em: http://www.isasensing.com/pt/.

2 ASP.Net Web API 2 web site. [Internet]. Disponível em: http://msdn.microsoft.com/en-us/library/dn448365(v=vs.118).aspx.

3 Scrum Wikipedia. [Internet]. Disponível em: http://pt.wikipedia.org/wiki/Scrum.

4 Web site do Enterprise Architect. [Internet]. Disponível em: http://www.sparxsystems.com.au/products/ea/index.html.

5 Web site do Angular JS - Google. [Internet]. Disponível em: http://angularjs.org/.

6 Malik S. Enterprise Dashboards, Design and Best Practices for IT. John Wiley & Sons; 2005.

7 Web site de especificação do Tableau Desktop. [Internet]. Disponível em: http://www.tableausoftware.com/products/desktop.

8 Web site de especificação do Tableau Server. [Internet]. Disponível em: http://www.tableausoftware.com/products/server.

9 Web site do Zoho Reports. [Internet]. Disponível em: https://www.zoho.com/reports/.

10 Web site do Visokio Omniscope. [Internet]. Disponível em: http://www.visokio.com/omniscope.

11 Informação de deployment com o Visokio. [Internet]. Disponível em: http://www.visokio.com/kb/deployment.

12 Web site do Klipfolio. [Internet]. Disponível em: http://www.klipfolio.com/.

13 Web site do Datazen. [Internet]. Disponível em: http://www.datazen.com/.

14 Web site do Quadbase Enterprise Reporting. [Internet]. Disponível em: http://www.quadbase.com/business-intelligence-enterprise-reporting/.

15 Web site do Arcplan Enterprise. [Internet]. Disponível em: http://www.arcplan.com/en/solutions/enterprise/.

16 Web site do Logi Info. [Internet]. Disponível em: http://www.logianalytics.com/dashboards-reports-and-analytics.

17 Web site de especificação do Pentaho. [Internet]. Disponível em: http://www.pentaho.com/product/business-visualization-analytics.

18 Web site do QlikView. [Internet]. Disponível em: http://www.qlikview.com.

[19] Clientes QlikView. [Internet]. Disponível em: http://www.qlikview.com/us/explore/customers.

[20] Qlikview no iPad. [Internet]. Disponível em: http://www.youtube.com/watch?v=tMULqhWt03c.

[21] Visão Global do QlikView. [Internet]. Disponível em: http://www.youtube.com/watch?v=sqILcEwlHSI.

[22] Preçário QlikView. [Internet]. Disponível em: http://www.qlikview.com/us/explore/pricing.

[23] Web site do Microsoft Sharepoint 2013. [Internet]. Disponível em: http://office.microsoft.com/en-us/sharepoint/sharepoint-2013-overview-collaboration-software-features-FX103789323.aspx.

[24] Hillier S, Pattison T. Microsoft SharePoint 2013 App Development. Microsoft Press; 2013.

[25] Beckett C, Medina O, Khipple K, Zhang R. Pro Sharepoint 2013 Branding and Responsive Web Development. Apress; 2013.

[26] Web site com documentação de Reporting Services. [Internet]. Disponível em: http://technet.microsoft.com/en-us/library/ms159106.aspx.

[27] Web site de AngularJS Directives. [Internet]. Disponível em: https://docs.angularjs.org/guide/directive.

[28] Souders S. Even Faster Web Sites. O'Reilly; 2009.

[29] Web site de documentação do $rootScope. [Internet]. Disponível em: https://docs.angularjs.org/api/ng/service/$rootScope.

[30] Web site da distribuição da biblioteca log4net. [Internet]. Disponível em: https://www.nuget.org/packages/log4net/.

[31] Wikipedia de JPA. [Internet]. Disponível em: http://pt.wikipedia.org/wiki/Java_Persistence_API.

[32] Wikipedia da Entity Framework. [Internet]. Disponível em: http://en.wikipedia.org/wiki/Entity_Framework.

[33] Web site do processo Database First de Entity Framework. [Internet]. Disponível em: http://msdn.microsoft.com/en-us/data/jj206878.aspx.

[34] Web site de iniciação à Entity Framework usando Code First. [Internet]. Disponível em: http://msdn.microsoft.com/pt-br/library/hh972463.aspx.

[35] Web site da Web API 2. [Internet]. Disponível em: http://msdn.microsoft.com/en-us/library/dn448365(v=vs.118).aspx.

Wikpedia de Plain Old CLR Object. [Internet]. Disponível em: http://en.wikipedia.org/wiki/Plain_Old_CLR_Object.

Web site da framework Service Stack. [Internet]. Disponível em: https://servicestack.net/.

Wikipedia de CORS. [Internet]. Disponível em: http://en.wikipedia.org/wiki/Cross-origin_resource_sharing.

Web site da framework Quartz.NET. [Internet]. Disponível em: http://www.quartz-scheduler.net/.

Web site da biblioteca RestSharp. [Internet]. Disponível em: http://restsharp.org/.

Web site da biblioteca Json.NET. [Internet]. Disponível em: http://james.newtonking.com/json.

Web site da biblioteca SQLClient. [Internet]. Disponível em: http://msdn.microsoft.com/en-us/library/system.data.sqlclient.aspx.

Web site do Bootstrap. [Internet]. Disponível em: http://getbootstrap.com/.

Web site da biblioteca HighCharts. [Internet]. Disponível em: http://www.highcharts.com/.

Github do highcharts-ng. [Internet]. Disponível em: https://github.com/pablojim/highcharts-ng.

Web site da biblioteca Moment.js. [Internet]. Disponível em: http://momentjs.com/.

Web site de documentação da biblioteca Time Period.NET. [Internet]. Disponível em: http://www.codeproject.com/Articles/168662/Time-Period-Library-for-NET.

Wikipedia do ISO 8601. [Internet]. Disponível em: http://en.wikipedia.org/wiki/ISO_8601.

Web site da biblioteca jsPDF. [Internet]. Disponível em: http://parall.ax/products/jspdf.

Web site da bibilioteca canvg. [Internet]. Disponível em: https://code.google.com/p/canvg/.

Web site de documentação da tag Canvas. [Internet]. Disponível em: http://www.w3schools.com/html/html5_canvas.asp.

Web site da biblioteca CryptoJS. [Internet]. Disponível em: https://code.google.com/p/crypto-js/.

[53] Web site de documentação da classe PrincipalContext. [Internet]. Disponível em: http://msdn.microsoft.com/en-us/system.directoryservices.accountmanagement.principalcontext.

[54] Github da biblioteca OutputCache V2. [Internet]. Disponível em: https://github.com/filipw/AspNetWebApi-OutputCache/blob/master/src/WebApi.OutputCache.V2/WebApi.OutputCache.V2.csproj.

[55] Green B, Seshadri S. AngularJS. O'Reilly Media; 2013.