Mestrado em Engenharia Informática

Estágio

Relatório Final

iTelemetry

João Carlos Pereira Ribeiro

joaocpr@student.dei.uc.pt

Empresa:

Intelligent Sensing Anywhere

Orientador empresa:

Pedro Feio

Orientador DEI:

Pedro Furtado

Data: 12 de Julho de 2011

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

Resumo

O presente documento é um relatório de estágio no âmbito da disciplina de Dissertação/Estágio do Mestrado de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Ao longo dos últimos anos, os desenvolvimentos do negócio da telemetria motivaram a empresa ISA à construção de uma plataforma capaz de responder aos novos requisitos do mercado. Nesta empresa, já existia uma plataforma dentro deste negócio. No entanto, a evolução dos requisitos mostrou que, cada vez mais, a sua complexidade e as suas limitações a tornaram incapaz de acompanhar a competição no mercado. Surgiu, portanto, a necessidade da criação da plataforma *iTelemetry*. Este estágio, na área de Engenharia de *Software*, pretende fazer parte do desenvolvimento desta plataforma, englobando tarefas como definição de requisitos, modelação, prototipagem, implementação e testes. A partir de uma plataforma multi-negócio, o *iTelemetry* disponibiliza várias aplicações *web* e serviços *windows*.

Palavras-chave

Caudalímetro, “Engenharia de *software*”, “gestão de serviços”, *iTelemetry*, *logger*, medição, *Scrum*, sensor, telemetria, “validação de dados”.

Agradecimentos

Em primeiro lugar, tenho a agradecer ao Engº Pedro Feio, orientador da empresa receptora, por todo o empenho, disponibilidade e apoio ao longo dos dez meses de estágio.

Também agradeço à restante equipa do projecto *iTelemetry*, Engº João Lourenço e Engª Sónia Cristóvão, que se disponibilizaram a esclarecer dúvidas e a ensinar técnicas, as quais melhoraram a qualidade do trabalho efectuado.

Não menos importante, o Professor Pedro Furtado apresentou disponibilidade e sugestões na construção do documento, pelo qual também agradeço.

Tenho, também, de agradecer à Daniela Gonçalves, por todo o apoio, tanto a nível psicológico, como no que diz respeito à construção de texto para o relatório.

Por último, quero agradecer à minha família pelo apoio, não só financeiro, mas também psicológico, mostrando confiança nas minhas capacidades.

Índice

Lista de figuras

Capítulo 1 Introdução

Este capítulo inicial tem como principal objectivo contextualizar o leitor no projecto *iTelemetry*, explicitando os objectivos comuns ao projecto *iTelemetry* e ao estágio, as metodologias e tecnologias utilizadas, bem como a definição de algumas palavras ou expressões essenciais. Termina-se o capítulo expondo a forma como o documento está organizado.

1.1 Contexto

A ISA - *Intelligent Sensing Anywhere* é uma empresa especializada em telemetria e M2M (machine-to-machine), líder em diferentes mercados, sobretudo na telemetria de *Oil & Gas*. Sediada no centro de Coimbra, é já uma empresa com uma elevada notoriedade internacional, devido ao elevado número de clientes estrangeiros.

Uma das suas especializações, a telemetria, segundo a referência [1], pode ser definida como medição remota de diversas utilidades, o que, na ISA, passa, principalmente, por gás, água, electricidade e combustíveis. Estas medições são feitas por sensores, que enviam a informação para servidores, sendo estes consultados por aplicações informáticas que tratam e organizam os dados.

O *iTelemetry* é uma plataforma que disponibiliza diversas aplicações *web* e serviços *Windows*. Nalgumas destas aplicações são apresentados valores de determinadas medições, tais como níveis de gás ou de água, registadas por sensores. Uma outra aplicação, o *BackOffice*, trata, essencialmente, de configurações de sensores, de clientes e de utilizadores. Quanto aos serviços *Windows*, são responsáveis por execuções com determinadas funcionalidades, como por exemplo, sincronização de dados ou envio de notificações.

Pretende-se que a plataforma, no geral, seja composta pelos seguintes módulos:

- módulo de aquisição de dados - responsável pela aquisição de dados que tenham como origem a plataforma de comunicação de dados da ISA;
- módulo de apresentação de dados - responsável pela disponibilização dos dados, obtidos dos equipamentos de telemetria, aos utilizadores;
- módulo de administração - responsável pela administração das várias entidades do sistema, tais como utilizadores, permissões, equipamentos, locais, objectos alvo de medição, etc;
- módulo de validação do sistema - responsável por permitir a validação de dados, automática e manual;
- módulo de manutenção do sistema para técnicos no terreno - responsável por auxiliar os técnicos no terreno na instalação e manutenção de equipamentos;
- módulo de gestão de processos e agendamentos - responsável por permitir a instalação e manutenção de processos agendados.

1.2 Objectivos

Dado que o *iTelemetry* é uma plataforma já no mercado, e que dá suporte a vários clientes, os objectivos deste estágio passam pela construção de novos módulos a aplicações, para que esta responda da melhor maneira aos requisitos e exigências do mercado internacional da telemetria. Pretende-se que a plataforma evolua na ambição e na longevidade, tornando-se mais completa e podendo moldar-se de acordo com as exigências de cada cliente.

No *iTelemetry*, no momento inicial do estágio, existiam módulos ainda por desenvolver, nomeadamente: módulo de validação do sistema, módulo de manutenção do sistema para técnicos no terreno, módulo de gestão de processos e agendamentos e módulo de visualização de dados. Segue-se uma explicação sobre cada um destes módulos.

O módulo de validação do sistema é dividido em dois submódulos: a validação automática de dados e a validação manual de dados. A validação automática de dados corrige desvios incoerentes com o padrão dos dados recebidos, ou seja, perante dados que não sejam reais (resultantes de problemas na recolha ou transmissão), é feita uma correcção automática, de forma a ajustá-los à realidade. A validação manual de dados inclui, quer a aprovação dos dados já validados automaticamente, quer a correcção de valores cujo sistema não conseguiu corrigir. Para a validação manual de dados é necessário disponibilizar uma aplicação aos responsáveis pelo acompanhamento do sistema. Antes de passarem a validação, os dados necessitam de ser calibrados, para ficarem na correcta ordem de grandeza.

O módulo de manutenção do sistema para técnicos no terreno surge devido à necessidade de apoiar os técnicos quando estes realizam uma intervenção, que pode ser instalação ou manutenção de equipamentos. Antes de uma intervenção, é necessário que os técnicos conheçam os materiais e as ferramentas que poderão ter de utilizar. Além disso, é, também, importante o conhecimento da localização dos equipamentos. Este módulo fornecerá tais informações, exigindo dos técnicos, no final, um relatório dos procedimentos efectuados.

No sistema existem processamentos automáticos que precisam ser manipulados - para se definirem as suas configurações, agendados - para se definirem os momentos das suas execuções, e controlados - para se garantir a sua boa execução ou detectar possíveis falhas. O módulo de gestão de processos e agendamentos permite efectuar tal gestão.

O módulo de visualização de dados já engloba duas aplicações, dedicadas às telemetrias de água e de *racks* de botijas de gás. No entanto, existem outros negócios cujos dados são incompatíveis com estas aplicações, como é o caso da telemetria de gás. Por este motivo é necessário evoluir este módulo, criando mais aplicações dedicadas a cada um dos possíveis negócios do *iTelemetry*.

O estágio incidiu sobre a evolução de alguns destes módulos, tais como o módulo de validação do sistema e o módulo de gestão de processos e agendamentos.

1.3 Metodologia

Neste estágio foi seguida a metodologia do departamento de *software* da ISA e do projecto em questão, que segue um processo de desenvolvimento ágil denominado *Scrum*.

Uma forte característica deste processo é a existência de *sprints* de tempo fixo, que pode ser entre duas e quatro semanas. Estas representam um período de tempo para completar trabalho. No final de cada *sprint*, existe um pacote de *software* funcional.

Um *Product Backlog* é um documento dinâmico que lista os itens que devem ser completos no projecto. Quem constrói o *Product Backlog* e define as prioridades dos itens é o *Product Owner*, que é o responsável pelo produto final. Para cada *sprint*, são seleccionados itens do *Product Backlog*, formando o *Sprint Backlog*. Cada item deve ser dividido em tarefas, estimando-se o tempo de realização de cada uma. Esta estimativa deve ser acordada pelos membros da equipa na reunião de planeamento de *sprint*. Esta reunião é o ponto de partida de uma *sprint*.

Em cada dia, a equipa do projecto reúne-se durante cerca de quinze minutos, para que a equipa interaja e para cada elemento actualizar os restantes do estado das suas tarefas. A estas reuniões dá-se o nome de *Daily Meetings* ou *Daily Scrum*.

Para terminar uma *sprint*, são necessárias duas reuniões: *Sprint Review* e *Sprint Retrospective*. Na *Sprint Review*, há uma apresentação aos interessados pelo produto, detalhando o que ficou completo e o que foi adiado para outra *sprint*. Também há uma demonstração do *software* aos interessados pelo produto, para que estes possam perceber como vai ser o produto final e para que dêem sugestões de melhorias. A *Sprint Retrospective* é uma reunião entre os membros da equipa, permitindo-lhes uma reflexão conjunta para que se possa melhorar o desempenho nas *sprints* seguintes.

Para orientar a equipa e os desenvolvimentos, tem de existir um responsável que, no contexto desta metodologia, se denomina *Scrum Master*. O seu principal objectivo é garantir que a equipa é funcional e produtiva, cumprindo o *Sprint Backlog*. Quanto a esta, é importante que seja multi-facetada, para que qualquer elemento possa fazer qualquer tarefa. Assim, as *sprints* são sempre equilibradas na distribuição de tarefas pelos elementos da equipa.

A acompanhar a *sprint*, existe um *Burndown Chart*, onde é visível a evolução do esforço da equipa em relação ao tempo pré-estimado. Para que este gráfico esteja de acordo com a realidade, é necessário que cada membro da equipa de projecto aponte correctamente o tempo dispendido em cada tarefa, e defina o tempo que falta para a sua conclusão.

Os conhecimentos demonstrados neste subcapítulo foram ganhos com o dia-a-dia do estágio, e acompanhados com a referência [2].

1.4 Tecnologias e Ferramentas

O projecto *iTelemetry* é desenvolvido utilizando a linguagem C# (*c-sharp*), para a programação lógica, e ASP.NET, para a programação da interface *web*. O desenvolvimento nestas linguagens foi feito com recurso à ferramenta *Microsoft Visual Studio 2010*, que é dedicada ao desenvolvimento de *software*. Quanto à ferramenta de ORM, utiliza-se a *Entity Framework 4.0*.

O sistema de gestão de base de dados utilizado no projecto é o *Microsoft SQL Server 2008*. No desenvolvimento, utiliza-se *LINQ to Entities* para a construção do *SQL* com o qual se acede às bases de dados. Para testar as *queries* de *LINQ to Entities*, é disponibilizado o *software LINQPad*. O *SQL Server Management Studio* é a interface utilizada pelos membros da equipa deste projecto, para manipular as bases de dados.

O *Enterprise Architect*, do grupo *Sparx Systems*, é utilizado no *iTelemetry*, principalmente, para desenho e modelação. Além disso, recorre-se a esta ferramenta para organização e armazenamento dos requisitos e dos casos de teste.

Já o *NUnit* é a ferramenta utilizada para testes unitários, integrando-se no *Microsoft Visual Studio 2010*. No desenvolvimento destes testes existe, por vezes, a necessidade de simular comportamento de alguns objectos. No *iTelemetry*, recorre-se aos *Mock's*, da ferramenta *Moq*, para simular os acessos à camada de acesso a dados.

Para analisar o código desenvolvido, existem diversas ferramentas, entre as quais o *FxCop*, que é a utilizada no projecto. Uma outra ferramenta de análise é o *Fiddler*, que analisa aplicações *web* no tráfico HTTP.

São também utilizadas as ferramentas do *Microsoft Office 2010*, como o *Word,* para relatórios e documentos de especificação, o *Excel,* para planos de testes, e o *PowerPoint*, para realização de protótipos e apresentações.

1.5 Definições

Definição	Significado
Log	Registo de um acontecimento.
***Rack* de botijas de gás**	Contentores metálicos que armazenam botijas de gás.
Model-View-Presenter	Segundo a referência [3], trata-se de uma arquitectura de *software* em que: • A *View* trata da exibição dos dados e da manipulação dos eventos na interface; • O *Presenter* faz a ponte, recebendo os dados do *Model*, organizando os mesmos, e devolvendo-os à *View* para a exibição; • O *Model* é o que trata da selecção dos dados a serem mostrados.
MicroPower	Tecnologia de pequenas baterias e com tempo de vida muito grande.
Assembly	No *iTelemetry*, uma *assembly* resume-se a um ficheiro com extensão ".dll". Geralmente, diz respeito a bibliotecas compiladas, podendo representar outros tipos de ficheiro, como por exemplo ficheiros com extensão ".exe".
ORM	Mapeamento objecto-relacional (*Object-Relational Mapping*) - representa a técnica de relacionar sistemas incompatíveis na orientação a objectos.
SQL	*Structured Query Language* é a linguagem de comunicação com sistemas de gestão de bases de dados.
WCF	*Windows Communication Fundation*: segundo a referência [10], é uma parte da *.NET Framework* que fornece um modelo de programação unificado para a construção de aplicações orientadas a serviços.

Tabela 1 - Definições

1.6 Estrutura do documento

Inicialmente, é feita uma introdução ao trabalho, onde se apresentam o contexto do tema, os objectivos comuns ao projecto da empresa e ao estágio, a metodologia em que se baseou o desenvolvimento e as tecnologias utilizadas. Como forma de auxiliar o leitor ao longo da análise deste trabalho, são, ainda, apresentadas algumas definições de conceitos usados.

O segundo capítulo é dedicado ao estado da arte. Nele, são apresentadas soluções semelhantes ao *iTelemetry* pertencentes a outras empresas, mas, também, uma solução já existente na ISA. No final, é apresentada uma análise comparativa das soluções, onde se detectaram aspectos importantes para o desenvolvimento do *iTelemetry*.

O capítulo que se segue serve para apresentar do plano de trabalhos do estágio. Através de um diagrama de *Gantt*, expõem-se as tarefas principais realizadas, seguindo-se o detalhe de cada uma delas, apresentando as respectivas subtarefas.

O capítulo quarto tem como objectivo expor o projecto *iTelemetry*. Apresentam-se as arquitecturas lógica e física do sistema, assim como os principais componentes da plataforma, tais como os serviços e as aplicações.

No quinto capítulo, de acordo com o plano de trabalhos estabelecido, é feita a apresentação de todo o trabalho desenvolvido ao longo do ano. Iniciando com a evolução do *ISA Scheduler*, segue-se a construção *Service Manager*, a reformulação da arquitectura lógica e a construção do módulo de validação do sistema. Termina-se com a exposição de outras tarefas realizadas, de menor dimensão mas não menos importantes para o projecto.

O trabalho termina com uma reflexão sobre todo o trabalho realizado.

Capítulo 2 Estado da arte

No mercado internacional da área da telemetria de *Oil & Gas*, a concorrência à ISA é vasta. O principal componente dos negócios desta área está nos equipamentos físicos de aquisição de dados. No entanto, é, também, bastante importante que haja *software*, adequado aos equipamentos, que disponibilize aos clientes os dados adquiridos.

Na ISA, no que diz respeito à área da telemetria de *Oil & Gas*, a opção do fabrico dos próprios *softwares* assentou, essencialmente, na melhor resposta que é possível dar a inovações nos equipamentos e a exigências dos clientes. No entanto, não é descartada a hipótese de se comprar *software* ou serviços que satisfaçam os requisitos de módulos inexistentes nas soluções da ISA.

É, portanto, importante que se conheçam os concorrentes do *iTelemetry*, estando actualizado sobre os módulos que possam vir a ser úteis e sobre as novidades às quais o *iTelemetry* tem que apresentar alternativas.

Deste modo, são apresentadas, de seguida, outras soluções que concorrem com o *iTelemetry*, sendo que o *MaisGas* pertence à ISA e as restantes pertencem a outras empresas internacionais. É sabida a existência de outras soluções, como a *Signalix* ou a *SilentSoft*, mas não existe informação suficiente disponível para que se conheçam os pormenores e se apresentem as suas características. No último subcapítulo é feita uma análise às soluções apresentadas, comparando-as com o *iTelemetry*.

2.1 MaisGas

O *MaisGas* é a plataforma da ISA que se pretende substituir com a construção do *iTelemetry*. Prestando serviços de telemetria de gás, água e electricidade, o *MaisGas* tem como principais objectivos a recolha e a visualização de dados enviados por dispositivos de aquisição de dados. Trata-se de uma plataforma que contém várias aplicações:

- *MGWeb*: visualização de dados;
- *MGManut*: configurações de equipamentos, agendamento de intervenções no terreno, criação de relatórios de monitorização de serviços e correcção de dados;
- *MGConfig*: gestão de entidades e de alarmes;
- *MGReport*: visualização de relatórios de facturação de serviços.

Para além das aplicações, o *MaisGas* contém vários serviços de exportação de dados.

Na Ilustração 1 está representada a arquitectura física do *MaisGas*.

Apesar de ter dado e continuar a dar provas das suas capacidades ao longo dos anos, tem alguns problemas assinalados difíceis de ultrapassar.

Um problema importante é a escassa documentação da plataforma, que tardou a ser iniciada e, além disso, não contém todas as especificações do *MaisGas*. A par disto, o código está, também, pouco comentado, e é pouco escalável, o que torna as suas análise e possíveis correcções difíceis de efectuar.

Outro ponto é o facto de, tanto a sua arquitectura, como o seu modelo de dados não estarem, ao início, preparados para um grande crescimento. Com o aumento do número de clientes, de requisitos e de funcionalidades, a arquitectura tornou-se demasiado complexa e o modelo de dados ficou mal estruturado devido ao forçado crescimento.

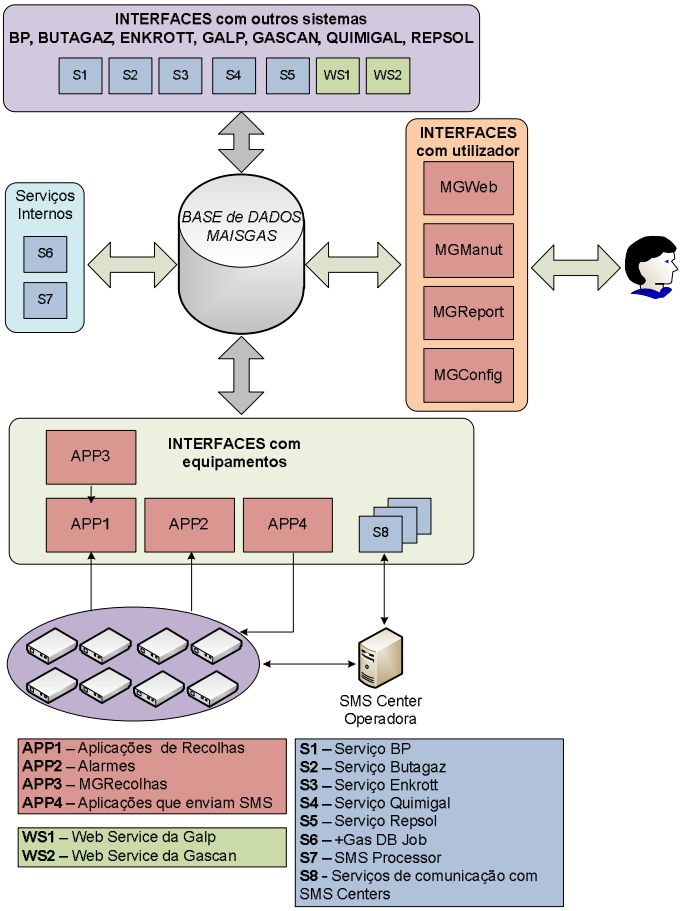


Ilustração 1 - Arquitectura Física do *MaisGas*

Também a forma de comunicar com os equipamentos é mais restrita, relativamente ao sistema usado no *iTelemetry*. Este último utiliza uma plataforma da ISA, o *iCenter*, que está em constante evolução, de forma a acompanhar as novas capacidades dos equipamentos, tendo cada vez mais funcionalidades. Já o *MaisGas*, apesar de poder comunicar com os equipamentos através de um SMS Center, por GPRS ou por Modem, não tem uma plataforma de comunicações independente, dedicada e em constante evolução, como é o caso do *iCenter*.

Um aspecto positivo desta solução é a validação dos dados. Há medida que chegam os dados ao sistema, é executado um algoritmo que, caso os dados cumpram certas condições, estes são validados automaticamente e, caso contrário e se existirem condições para tal, são automaticamente corrigidos. Quando não é possível corrigir automaticamente, é necessária a intervenção manual sobre os respectivos dados. Este módulo é considerado bem sucedido, uma vez que apresenta um baixo número de queixas dos clientes, quando estes comparam os dados fornecidos pelo *software* aos dados medidos nos equipamentos, e, também, porque deixa um número reduzido de dados para serem validados manualmente.

O *iTelemetry* esforça-se por melhorar os pontos menos fortes do *MaisGas*, sem que isso prejudique os aspectos mais fortes, que servem de modelo ao *iTelemetry*.

2.2 IMR Server 4 Enterprise Suite

Esta solução é um pacote de *software* para aquisição de dados e gestão de dispositivos de comunicação. Como citado na referência [4], o pacote consiste num servidor de aquisição de dados, num *data warehouse*, na apresentação de leituras para diferentes tipos de utilizadores, na gestão de dispositivos, nas ferramentas para instalação nos locais e nas interfaces para outros sistemas de gestão. A arquitectura física desta solução é apresentada na Ilustração 2, também retirada da referência [4].

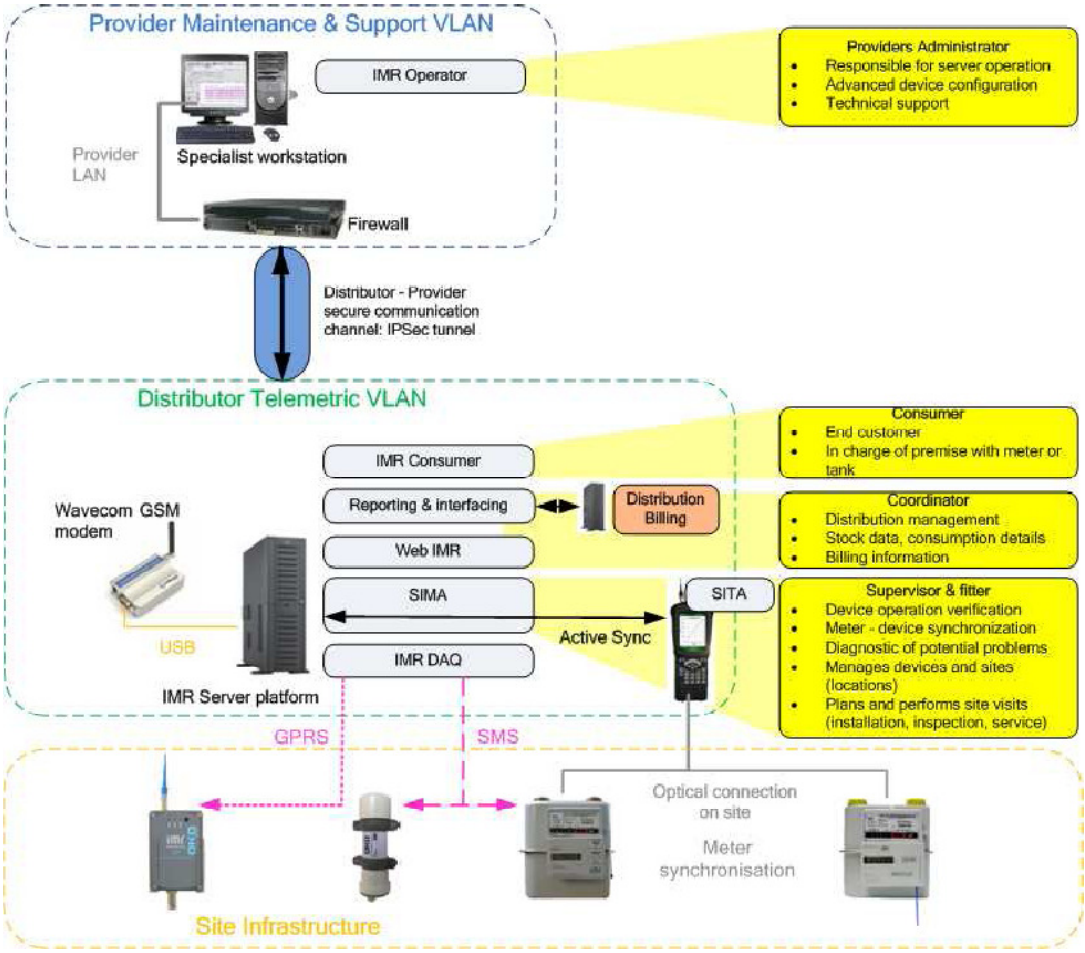


Ilustração 2 - Arquitectura física do IMR Server 4 Enterprise Suite

O servidor de aquisição de dados está preparado para receber e guardar os dados de milhares de dispositivos, sendo que estes dados podem ser recebidos por SMS, GPRS ou UDP/IP. Os dispositivos podem gerar eventos de alarme, que podem resultar em avisos por SMS e/ou *e-mail*. Todos os dados deste servidor, quer sejam de medições, diagnósticos ou parâmetros de configuração, estão armazenados na base de dados, e podem ser utilizados por *3rd party software*'s (*softwares* de outros fabricantes).

O pacote contém, também, a aplicação SIMA, que é uma ferramenta dedicada e intuitiva para gestão de dispositivos e sistemas. Esta aplicação dispõe de várias funcionalidades, tais como: geo-referenciação dos locais onde existem equipamentos; e envio de configurações para os dispositivos, como, por exemplo, comandos para abertura ou fecho de válvulas nos

contadores. Trata-se de uma aplicação windows que necessita de instalação nos computadores dos utilizadores.

A aplicação SITA, também contida no pacote, é uma aplicação de telemóvel. Mais limitada do que a aplicação SIMA, as suas funcionalidades restringem-se às consultas de leituras e do estado de instalação de dispositivos.

2.3 myDatanet

O *myDatanet*, pertencente à *Microtronics*, foi o primeiro sistema do mundo, de entre os sistemas de aquisição de dados, a utilizar a tecnologia *MicroPower*. Neste sistema, segundo a referência [5], os dados são recolhidos num sensor *wireless* e, depois de enviados para um servidor central, são apresentados numa aplicação *web*. Esta solução suporta medições de gás, água ou óleos, medições de emissão de gases, tráfego de trânsito e meteorologia.

A aplicação *web* desta solução é denominada por *myDataweb* e, de entre as suas funcionalidades e características, destacam-se as seguintes:

- apresentação dos dados segundos depois de serem adquiridos;
- estatísticas sobre a utilização do sistema;
- instalação de equipamentos;
- gestão de clientes;
- um ponto central de configurações;
- protocolo de comunicações optimizado.

Esta é uma solução que aparenta ser completa, embora haja pouca informação disponibilizada que contenha características e funcionalidades.

2.4 Netbiter

A solução da *Netbiter* possibilita a gestão remota nos negócios de HVAC (aquecimento, ventilação e condicionamento de ar) em edifícios, de água, de tanques, de geradores de energia, de energias renováveis e de antenas (por exemplo, de rede de telemóveis).

Para a aquisição dos dados, a solução engloba os *Netbiter EasyConnect*, que são dispositivos de comunicação desenhados para transmitir os dados adquiridos por sensores dedicados, que comunicam por *wireless*, e que são integráveis com o *Netbiter Argos*, que é a aplicação *web* que compõe a solução.

Uma funcionalidade disponibilizada por esta aplicação, que pode ser acedida por PC, PDA ou telemóvel (que tenha acesso a internet), é a monitorização dos dispositivos através da sua geo-referenciação e do controlo da sua "saúde". Também é possível, através da aplicação, gerir alarmes, definindo os valores que definem a fronteira entre alarme e não alarme, e configurando formas de receber os alarmes. Na visualização de dados, é possível comparar valores e fazer uma análise ao histórico dos dados. A aplicação também contém um módulo de configurações, para gerir utilizadores e equipamentos, e um módulo para apoio aos técnicos no terreno, com a localização geográfica dos equipamentos e optimização de rotas para os técnicos se deslocarem aos locais.

2.5 Homerider

Homerider é uma solução dedicada à telemetria de vários negócios, que estão listados na referência [8]. Destes, destacam-se a medição remota de água, gás, electricidade e calor, a

medição dos níveis das redes de esgotos e a monitorização remota de tanques para LPG (gás de petróleo liquefeito), combustíveis e depósitos.

Conforme a referência [9], a *Fusion* é a aplicação *web* da solução *Homerider* onde os utilizadores podem consultar e comparar os dados adquiridos em forma de gráfico.

Esta é mais uma solução que pode ser interessante, mas a sua limitada informação disponível não permite uma avaliação muito profunda.

2.6 Análise comparativa

De entre as soluções estudadas, a nível de funcionalidades, não existem diferenças muito significativas. Apesar da informação disponibilizada não ser suficiente para análises mais detalhadas, todas as soluções aparentam ser completas, o que resulta na necessidade de focar o trabalho na qualidade de todos os componentes das soluções e, principalmente, do *software*, que é a ferramenta com que os clientes mais interagem. Quanto à qualidade, dado que não há acesso às soluções, não é possível efectuar uma análise comparativa.

A utilização de produtos externos não foi uma opção válida para completar o *iTelemetry* no módulo de gestão de processos e agendamentos. O principal requisito deste é que seja dedicado a uma ferramenta da ISA, o *ISA Scheduler*, que as possíveis soluções externas não conhecem. Para responder a todos os requisitos deste módulo, decidiu-se pela implementação do *Service Manager*.

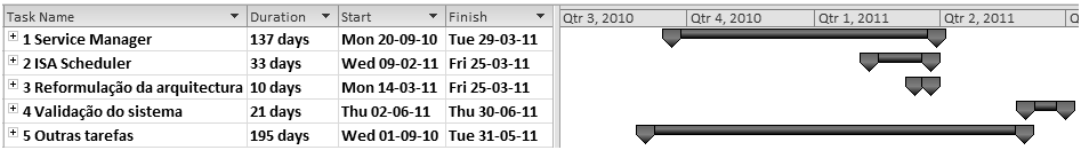
O módulo de validação do sistema do *iTelemetry* terá como base a validação de dados do *MaisGas*, visto que este é um dos pontos fortes desta plataforma. Tendo em conta que esta solução tem qualidade e não resulta em quaisquer custos para a ISA, as possíveis vantagens de outras soluções não são suficientes para se superiorizarem à opção da utilização da validação de dados do *MaisGas*.

Comparando as soluções da ISA às restantes soluções estudadas, pode dizer-se que o *MaisGas*, plataforma de orientação para o *iTelemetry*, até ao momento, é considerado equivalente às outras soluções, mas não está preparado para grandes evoluções. O *iTelemetry* tem um grande potencial de evolução e tem qualidade nas suas funcionalidades implementadas. Por isso, é importante que o *iTelemetry* chegue ao nível das restantes soluções, mantendo a ISA no mais alto patamar da competitividade no negócio de telemetria de *Oil & Gas*. Para alcançar este nível, é necessária, então, a implementação dos módulos planeados.

Capítulo 3 Plano de trabalhos

O presente capítulo serve para apresentar o plano de trabalhos realizados ao longo do estágio. Inicialmente, foi proposto um plano de trabalhos que, ao longo dos desenvolvimentos, foi sendo adaptado, por forma a ir ao encontro das necessidades do projecto. Deste modo, serão apresentadas aqui todas as tarefas que, de facto, foram efectuadas.

A Ilustração 3 apresenta o plano de trabalhos do estágio, num Diagrama de *Gantt*, onde estão incluídas as principais tarefas executadas. Para consultar este diagrama em mais detalhe, veja-se Anexo I - Plano de trabalhos do estágio.



Task Name	Duration	Start	Finish
1 Service Manager	137 days	Mon 20-09-10	Tue 29-03-11
2 ISA Scheduler	33 days	Wed 09-02-11	Fri 25-03-11
3 Reformulação da arquitectura	10 days	Mon 14-03-11	Fri 25-03-11
4 Validação do sistema	21 days	Thu 02-06-11	Thu 30-06-11
5 Outras tarefas	195 days	Wed 01-09-10	Tue 31-05-11

Ilustração 3 - Diagrama de Gantt referente ao plano de trabalhos do estágio

O *Service Manager* diz respeito ao módulo de gestão de processos e agendamentos do *iTelemetry* e é uma aplicação *web* que permite definir processos a executar, organizar quando estes são executados e analisar as execuções. O *ISA Scheduler* é a ferramenta responsável por executar os processos definidos no *Service Manager* e, para o estágio, propôs-se efectuar melhoramentos a nível estrutural e de fiabilidade. A reformulação da arquitectura é a tarefa que tem a finalidade de reformular a estrutura lógica do *iTelemetry*, definindo uma arquitectura que responda melhor às necessidades da plataforma. A validação do sistema tem como objectivo avaliar os dados recebidos antes de disponibilizá-los aos utilizadores.

Note-se que a duração das tarefas no primeiro semestre (até fim de Janeiro) é influenciada pelo regime de *part-time*. Neste regime, as presenças eram de cerca de vinte horas semanais, contrastando com as quarenta horas semanais no segundo semestre. Com menos tempo dedicado por semana, sentiu-se, como previsto, que o *focus* foi menor, e que o esforço necessário para a realização de cada tarefa foi, consequentemente, maior. Para além disso, no primeiro semestre, muito do tempo do estágio foi dedicado às reuniões de planeamento, revisão e retrospectiva de s*print*. É, por isso, razoável que a duração da tarefa *Service Manager* tenha sido tão extensa.

No diagrama de *Gantt* é visível uma sobreposição de duas tarefas. A determinado momento do desenrolar das subtarefas referentes ao *ISA Scheduler*, surgiu a necessidade de se começarem a discutir aspectos relativos à reformulação da arquitectura. Para tal, era necessária a realização de reuniões com alguma regularidade, facto nem sempre possível devido à indisponibilidade de alguns elementos da equipa e das próprias salas de reunião. Como tal, esses "intervalos" foram aproveitados para continuar o desenvolvimento do *ISA Scheduler*, justificando, assim, a sobreposição ocorrida.

De seguida, para cada tarefa do estágio, são apresentadas as suas subtarefas em forma de gráfico temporal, bem como as *sprints* em que estas foram executadas. As subtarefas estão representadas em números e encontram-se legendadas numa tabela que sucede ao gráfico, para uma melhor visualização.

O último subcapítulo refere-se a outras tarefas que, não sendo principais, foram executadas intercaladamente com as ditas principais. A sua explicação encontra-se no subcapítulo 5.5.

3.1 Service Manager

O *Service Manager* é uma aplicação do *iTelemetry* responsável pela gestão de processos e agendamentos. A sua construção foi dividida em várias subtarefas, como apresentado nas Ilustração 4 e Ilustração 5 através de gráficos temporais, acompanhados da Tabela 2 que serve de legenda aos mesmos.

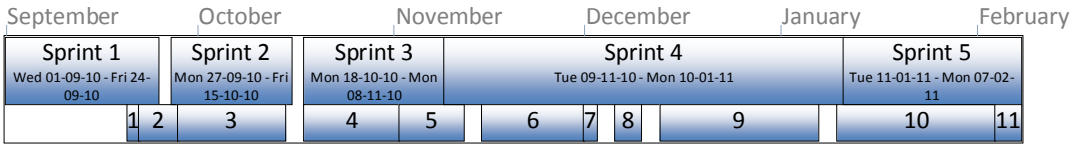


Ilustração 4 - Gráfico temporal das subtarefas do *Service Manager* - Parte 1

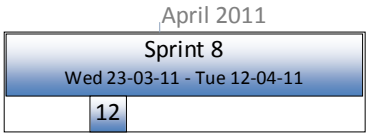


Ilustração 5 - Gráfico temporal das subtarefas do *Service Manager* - Parte 2

	Tarefa	Início	Fim
1	Análise de requisitos	Mon 20-09-10	Tue 21-09-10
2	Protótipo	Wed 22-09-10	Mon 27-09-10
3	Consulta de serviços	Tue 28-09-10	Thu 14-10-10
4	Consulta de processos	Mon 18-10-10	Mon 01-11-10
5	Consulta de agendamentos	Tue 02-11-10	Thu 11-11-10
6	Consulta de resultados de execução	Mon 15-11-10	Tue 30-11-10
7	Gestão de processos e agendamentos- análise de requisitos	Wed 01-12-10	Thu 02-12-10
8	Gestão de processos e agendamentos - protótipo	Mon 06-12-10	Thu 09-12-10
9	Análise da funcionalidade Reflection	Mon 13-12-10	Thu 06-01-11
10	Gestão de processos e agendamentos - implementação	Mon 10-01-11	Thu 03-02-11
11	Gestão de permissões a servidores	Fri 04-02-11	Mon 07-02-11
12	Alteração das configurações de notificações	Mon 28-03-11	Tue 29-03-11

Tabela 2 - Subtarefas do *Service Manager* ilustradas nos gráficos temporais

Na primeira abordagem ao *Service Manager*, foram realizadas as subtarefas de análise de requisitos, prototipagem e implementação, tendo como objectivos as consultas de serviços, processos, agendamentos e resultados de execução. Depois destes desenvolvimentos, foram definidos novos objectivos relativos a configuração de processos, que resultaram em novas fases de análise de requisitos, prototipagem e implementação.

As análises de requisitos e as prototipagens precederam as respectivas implementações. No entanto, ao longo das implementações, tanto os requisitos como o protótipo sofreram pequenas alterações, originadas por novas decisões e funcionalidades não pensadas previamente.

Apesar de não haver subtarefas exclusivamente dedicadas aos testes, estes estão inseridos na implementação, e serviram para garantir que o implementado estava de acordo com o que era esperado.

Devido à sua necessidade na configuração de processos, foi efectuada uma análise à funcionalidade *Reflection*. Pretendia-se conhecer melhor as suas capacidades, bem como a forma da sua utilização.

A tarefa do *Service Manager* foi distribuída entre o final da *sprint* 1 e o final da *sprint* 5, com excepção da subtarefa de alteração das configurações de notificações, que só foi desenvolvida na *sprint* 8. Esta foi executada posteriormente à execução da tarefa *ISA Scheduler*, pois resultou dos novos requisitos da ferramenta *ISA Scheduler*. Esta ferramenta, como está explicado no subcapítulo 5.2, influencia directamente a aplicação *Service Manager*.

3.2 ISA Scheduler

Neste estágio houve, também, o desenvolvimento de uma nova versão do serviço *ISA Scheduler*, que é responsável por executar processos agendados. O gráfico temporal das subtarefas do *ISA Scheduler* está apresentado na Ilustração 6, seguido da Tabela 3 que as explicita.

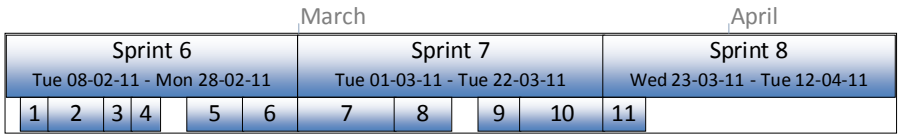


Ilustração 6 - Gráfico temporal das subtarefas do *ISA Scheduler*

	Tarefa	Início	Fim
1	Análise de novos requisitos	Wed 09-02-11	Thu 10-02-11
2	Alteração do esquema da BD	Fri 11-02-11	Mon 14-02-11
3	Forçar execuções - Implementação	Tue 15-02-11	Wed 16-02-11
4	Forçar execuções - Testes	Thu 17-02-11	Fri 18-02-11
5	Notificações por chamadas - modelação	Mon 21-02-11	Thu 24-02-11
6	Notificações por chamadas - testes físicos de chamadas	Fri 25-02-11	Mon 28-02-11
7	Notificações por chamadas - implementação	Tue 01-03-11	Mon 07-03-11
8	Notificações por chamadas - testes	Tue 08-03-11	Fri 11-03-11
9	Reformulação da estrutura do serviço - modelação	Mon 14-03-11	Wed 16-03-11
10	Reformulação da estrutura do serviço - Implementação	Thu 17-03-11	Tue 22-03-11
11	Testes unitários	Wed 23-03-11	Fri 25-03-11

Tabela 3 - Subtarefas do *ISA Scheduler* ilustradas no gráfico temporal

Durante a primeira tarefa, a análise de requisitos, detectou-se a necessidade de três melhorias no *ISA Scheduler*, que foram aplicadas uma de cada vez. O primeiro aspecto tratado foi a inclusão da opção de se poderem forçar as execuções, para os casos da execução agendada ter corrido mal, ou de ser necessária a execução de um processo fora do agendamento definido. Outro aspecto foi a inclusão de notificações por chamadas telefónicas, já que, antes, só existiam notificações por *e-mail*. Para esta melhoria, foi necessário o desenho, por forma a definir o modo como as notificações deviam ser construídas. A última melhoria assentou na reformulação do serviço, para o tornar mais fiável e melhor separado, sem haver tanta dependência entre as funcionalidades. Para este tipo de melhoria, é imperativo que haja dedicação à fase de desenho.

O *ISA Scheduler* é um serviço que corre automaticamente, sem que haja acompanhamento humano permanente, facto que torna mais difícil a detecção de defeitos nas suas execuções. Como tal, foi dada uma grande importância aos testes, para garantir o menor número de defeitos possível.

A tarefa *ISA Scheduler* foi executada durante as *sprints* 6 e 7, tendo-se estendido até ao início da *sprint* 8.

3.3 Reformulação da arquitectura

A reformulação da arquitectura é uma necessidade que surge, por vezes, devido ao surgimento de novos requisitos ou à impossibilidade de se conseguir obter previamente a melhor forma de arquitectar uma plataforma. O *iTelemetry* não foi execpção e, como tal, foi necessário reestruturar a sua arquitectura lógica, sendo que a principal componente desta tarefa consistiu na modelação da nova estrutura. As subtarefas referentes à reformulação da arquitectura em que o estagiário participou estão dispostas de forma temporal na Ilustração 7, à qual se segue a Tabela 4 com a restante informação.

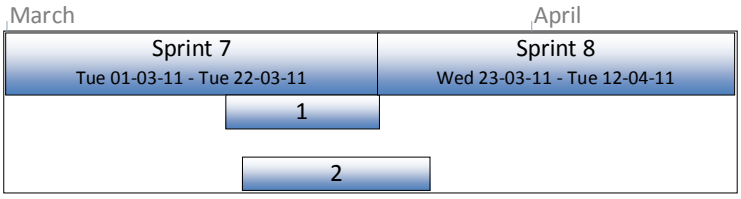


Ilustração 7 - Gráfico temporal das subtarefas da reformulação da arquitectura

	Tarefa	Início	Fim
1	Discussão	Mon 14-03-11	Tue 22-03-11
2	Modelação	Tue 15-03-11	Fri 25-03-11

Tabela 4 - Subtarefas da reformulação da arquitectura ilustradas no gráfico temporal

Como é possível verificar, no gráfico acima, pela sobreposição das tarefas discussão e modelação e desenho, estas não foram independentes uma da outra. De facto, foi necessário começar com a discussão de ideias entre a equipa por forma a poder começar-se o desenho, mas, com a evolução deste, novos aspectos foram surgindo com necessidade de novas discussões. Verificou-se tal sucessão de acontecimentos até à obtenção de uma arquitectura cujas dificuldades, previamente encontradas, já tenham sido ultrapassadas.

As subtarefas em questão foram executadas ao longo das *sprints* 7 e 8.

Da reformulação da arquitectura fizeram parte outras subtarefas, tais como a adequação da estrutura do código e a execução de testes unitários para garantir que as funcionalidades não foram afectadas. Tais subtarefas não estão explicitadas no presente relatório uma vez que não foram executadas pelo estagiário.

3.4 Validação do sistema

O módulo de validação do sistema engloba todas as transformações de dados desde o momento em que são recebidos no *iTelemetry* até ao momento em que são disponibilizados

aos clientes. Os dados dizem respeito às medições efectuadas nos locais sujeitos a telemetria, e devem ser processados de forma a garantir a sua coerência. As subtarefas referentes a este módulo estão dispostas num gráfico temporal na Ilustração 8, que é acompanhado pela Tabela 5 que completa a informação.

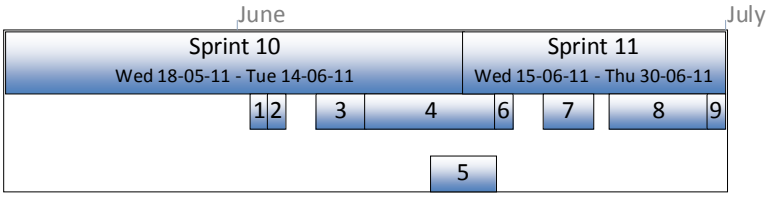


Ilustração 8 - Gráfico temporal das subtarefas da validação do sistema

	Tarefa	Início	Fim
1	Adaptação do esquema da base de dados	Thu 02-06-11	Thu 02-06-11
2	Estudo da linguagem a utilizar	Fri 03-06-11	Fri 03-06-11
3	Contagens de água - análise de casos	Mon 06-06-11	Wed 08-06-11
4	Modelação	Thu 09-06-11	Thu 16-06-11
5	Contagens de água - Implementação	Mon 13-06-11	Thu 16-06-11
6	Testes de carga	Fri 17-06-11	Fri 17-06-11
7	Testes unitários	Mon 20-06-11	Wed 22-06-11
8	Níveis de gás - análise de casos	Fri 24-06-11	Wed 29-06-11
9	Validação manual - protótipo	Thu 30-06-11	Thu 30-06-11

Tabela 5 - Subtarefas da validação do sistema ilustradas no gráfico temporal

A tarefa de validação do sistema foi executada entre a segunda metade da *sprint* 10 e a *sprint* 11. Depois de, inicialmente, se alterar o esquema da base de dados de modo a que esta consiga suportar a calibração e a validação dos dados, estudou-se a linguagem F# por ser mais dedicada a algoritmos e compatível com a linguagem C#, utilizada no *iTelemetry*. No entanto, quando realizados testes, ficou demonstrada uma pior performance aquando do uso de F#, pelo que não se aprofundou este estudo.

Quanto às contagens de água, começaram por analisar-se os diferentes comportamentos que uma sequência de dados pode ter. À medida que se foi modelando todo o serviço para calibração e validação automática de dados, foi-se implementando o algoritmo para as contagens de água, com base nas decisões tomadas durante a modelação. Deste modo, foram surgindo novos aspectos, que implicaram a actualização da modelação. Depois de implementado o algoritmo, foram feitos testes de carga para garantir a boa performance, e testes unitários para garantir o correcto tratamento dos dados.

Depois das contagens de água, ainda houve tempo para a análise dos diferentes casos que uma sequência de dados de nível de gás pode apresentar. No entanto, esta mostrou-se mais complexa do que a análise feita às contagens de água, uma vez que tem mais regras a cumprir. Por fim, construiu-se um protótipo para uma aplicação de validação/calibração manual de dados.

3.5 iTelemetry - Outras tarefas

Além das tarefas ditas principais para o projecto, há, ainda, que referir outras tarefas executadas ao longo do estágio e devidamente enquadradas no *iTelemetry*. Dedica-se este subcapítulo a essas tais tarefas, que são a adaptação à empresa, os estudos efectuados de

comparação de *profilers* e de análise de *Templates* AJAX, as preparações de entregas a clientes, a redefinição da relação entre Unidades e Locais e a integração de unidades de engenharia. Nas Ilustração 9, Ilustração 10 e Ilustração 11 são apresentadas, sob a forma de gráfico temporal e com a Tabela 6 que completa a informação, as subtarefas destas tarefas do *iTelemetry* que fizeram parte do estágio.

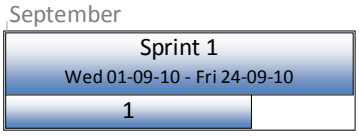


Ilustração 9 - Gráfico temporal das outras tarefas do *iTelemetry* - Parte 1

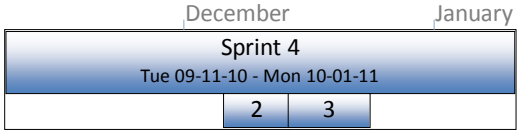


Ilustração 10 - Gráfico temporal das outras tarefas do *iTelemetry* - Parte 2

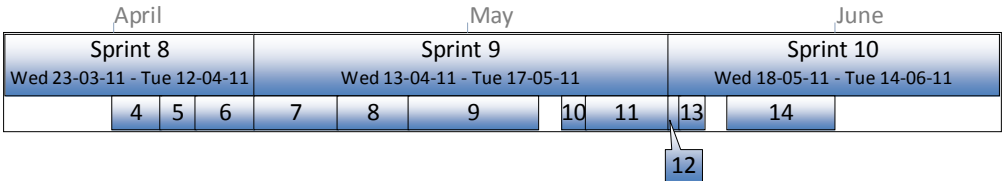


Ilustração 11 - Gráfico temporal das outras tarefas do *iTelemetry* - Parte 3

	Tarefa	Início	Fim
1	Adaptação à empresa	Wed 01-09-10	Fri 17-09-10
2	Comparação de Profilers	Mon 06-12-10	Mon 13-12-10
3	Templates AJAX	Tue 14-12-10	Thu 23-12-10
4	Relação unidades-locais: Análise de requisitos	Fri 01-04-11	Mon 04-04-11
5	Relação unidades-locais: Protótipo	Tue 05-04-11	Thu 07-04-11
6	Criação de templates para fichas de revisão e de verificação	Fri 08-04-11	Tue 12-04-11
7	Preenchimento de fichas de revisão e de verificação	Wed 13-04-11	Tue 19-04-11
8	Testes às aplicações	Wed 20-04-11	Mon 25-04-11
9	Relação unidades-locais: Implementação	Tue 26-04-11	Fri 06-05-11
10	Preenchimento de fichas de revisão e de verificação	Mon 09-05-11	Tue 10-05-11
11	Testes às aplicações	Wed 11-05-11	Tue 17-05-11
12	Unidades de engenharia: Análise de requisitos	Wed 18-05-11	Wed 18-05-11
13	Unidades de engenharia: Protótipo	Thu 19-05-11	Fri 20-05-11
14	Unidades de engenharia: Implementação	Mon 23-05-11	Tue 31-05-11

Tabela 6 - Subtarefas das outras tarefas do *iTelemetry* ilustradas nos gráficos temporais

A adaptação à empresa foi feita durante a *sprint* 1, que antecedeu a tarefa do *Service Manager*. Já as subtarefas referentes aos estudos de comparação de *profilers* e de análise de *Templates* AJAX foram executadas na *sprint* 4, numa interrupção dos desenvolvimentos da tarefa do *Service Manager*. Tanto o preenchimento de fichas de revisão e de verificações como

os testes às aplicações fizeram parte da preparação de publicações do *iTelemetry*, e estiveram incluídos na *sprint* 9. Repartidas entre as *sprints* 8 e 9 estiveram as subtarefas associadas à redefinição da relação Unidades-Locais, que englobaram análise de requisitos, prototipagem e implementação. As subtarefas relacionadas com as unidades de engenharia foram executadas no início da *sprint* 10, existindo, mais uma vez, fases de análise de requisitos, de realização do protótipo e de implementação.

Estas tarefas, consideradas como secundárias, estão organizadas e explicadas no subcapítulo 5.5.

Capítulo 4 iTelemetry

O *iTelemetry* é uma plataforma multi-negócio que prima pela consistência e escalabilidade. Além de, quer a arquitectura, quer o código seguirem a mesma estrutura na maioria dos módulos, o *iTelemetry* está preparado para crescer.

Actualmente, esta plataforma abrange a telemetria de gás e de água, entre outras, sendo que, no futuro, será, também, capaz de suportar outro tipo de negócio que a ISA poderá vir a tratar.

Esta plataforma tem como orientação uma outra plataforma da ISA já existente, denominada por *MaisGas,* que, por sua vez, tem dado provas de sucesso ao longo dos anos. No entanto, com a evolução das tecnologias e com os novos requisitos e exigências do mercado internacional de telemetria, o acompanhamento destas por parte do *MaisGas* tornou-se uma tarefa complexa, dando, assim, oportunidade ao desenvolvimento do *iTelemetry*. Tais desenvolvimentos têm vindo a torná-lo numa plataforma forte neste negócio.

A longo prazo, pretende-se que o *iTelemetry* seja capaz de fazer frente a uma concorrência que se vai tornando cada vez mais feroz, contribuindo, assim, para cimentar a posição da ISA como empresa de referência neste sector do mercado.

Neste capítulo encontram-se os diagramas das arquitecturas lógica e física do sistema, bem como uma análise dos mesmos. Seguem-se a definição e a enumeração das unidades utilizadas no *iTelemetry*, e a apresentação da plataforma *iCenter*, responsável por estabelecer as comunicações entre a plataforma *iTelemetry* e as suas unidades. Por fim, são descritos os serviços e as aplicações que definem o *iTelemetry*.

4.1 Arquitectura lógica do sistema

A arquitectura lógica é a forma como os componentes lógicos de uma solução são organizados e integrados. Na Ilustração 12, através de um diagrama de componentes, é apresentada a interacção lógica do *iTelemetry*.

O *iTelemetry* está dividido em três camadas aplicacionais: a camada de apresentação e de serviços (PSL), a camada de negócio ou camada lógica (BL) e a camada de acesso a dados (DAL).

A camada de apresentação e de serviços engloba as aplicações e os serviços do *iTelemetry*. Nesta camada está presente a gestão das vistas das aplicações *web*, onde se verifica a interacção com o utilizador e a gestão dos eventos disparados por acção deste, e a gestão de serviços. Cada aplicação tem, nesta camada, a sua *View* (veja-se subcapítulo 0), e os seus *Presenter* e *Presenter.Web*, que se distinguem no facto do *Presenter* tratar do que é independente do tipo de aplicação e o *Presenter.Web*, pelo contrário, tratar do que depende do tipo de aplicação. Note-se que estes dois "*Presenters*" formam o conceito de *Presenter* da arquitectura *Model-View-Presenter*, definido no subcapítulo 0. Apesar de existir acesso directo entre o *Presenter* e o *Presenter.Web* da mesma aplicação, o mesmo não acontece com o *Presenter* e a *View*, que só interagem através das interfaces que implementam. Existe, também, nesta camada, uma aplicação virtual, denominada *Common*, da qual são de referir os seguintes factos: a sua *View* contém objectos comuns às *Views* de várias aplicações, tal como o controlo de alteração da *password*; e o seu *Presenter* contém acções comuns aos *Presenter.Web*'s de várias aplicações, como é o caso da acção disparada quando um utilizador altera a *password*. O acesso desta camada à BL só é feito de duas maneiras: através dos serviços que, não tendo qualquer interacção na PSL, acedem às suas componentes da BL; e através dos *Presenter.Web*'s das aplicações, que comunicam com os *Models*, também presentes na BL.

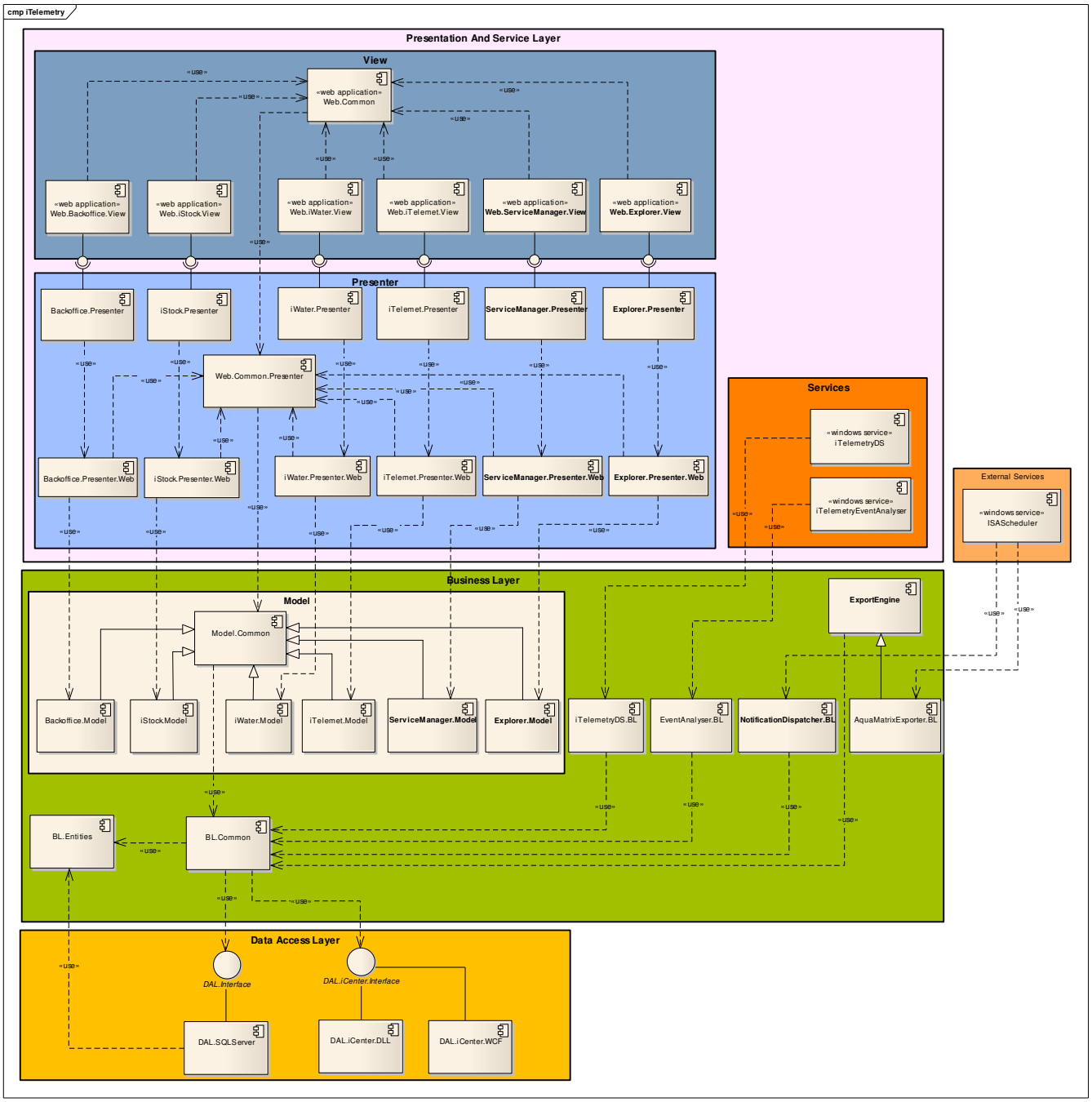


Ilustração 12 - Diagrama de componentes com a Arquitectura Lógica do iTelemetry

A camada lógica tem como finalidade receber os dados provenientes da camada de acesso a dados, tratá-los e entregá-los à camada de apresentação e de serviços. Neste nível estão presentes a lógica e as regras de negócio do *iTelemetry*, tanto a nível geral, na *BL.Common*, como a nível específico para cada aplicação, nos respectivos *Models*. Também é nesta camada que os serviços têm as suas lógicas dedicadas, as quais se encontram em módulos identificados com os nomes dos respectivos serviços. A *BL.Common* só é acedida a partir do interior da camada lógica, e é somente este componente que acede à DAL. Na BL existe, também, o componente *BL.Entities* que contém as entidades definidas no sistema, e que, apesar de não

estar representado na Ilustração 12 devido à confusão visual que geraria, pode ser acedido a partir de qualquer componente do *iTelemetry*.

A camada de acesso a dados é responsável por aceder às bases de dados para ler os registos lá guardados, remover dados, editar os valores guardados, ou inserir novos registos. Os componentes da DAL só são acedidos através da interface que implementam, e não interagem entre si.

Note-se que, devido a requisitos e funcionalidades específicas, existem aplicações do *iTelemetry* que podem não seguir na sua totalidade esta arquitectura. Um exemplo dessa fuga é o *Service Manager* que, conforme decidido entre a equipa, apenas segue esta arquitectura para tratar a autenticação e gestão da sessão. Como é dedicada a uma base de dados externa à solução, não é possível integrá-la nesta arquitectura, na sua totalidade. No entanto, tendo em conta os seus objectivos, faz todo o sentido que esta esteja incluída na plataforma *iTelemetry*. A especificação do *Service Manager* está incluída no subcapítulo 5.2, dedicado ao seu desenvolvimento.

Também há a notar que o serviço externo *ISA Scheduler*, apresentado na Ilustração 12, não faz parte da solução. No entanto, é ele que gere os processos *AquaMatrixExporter* e *NotificationDispatcher*, como se explica no subcapítulo 5.1.

4.2 Arquitectura física do sistema

A arquitectura física define a instalação e distribuição das entidades num sistema. Na Ilustração 13 está representada a interacção do *iTelemetry* com outras entidades importantes nos negócios tratados, através de um digrama de instalação (*deployment diagram*).

No servidor SQL Server encontram-se as bases de dados do sistema, ou seja, a base de dados geral do *iTelemetry*, a base de dados do *ISA Scheduler* e as bases de dados do *iCenter*, tanto a geral como a de *logs*. A base de dados geral do *iTelemetry* é acedida por todas as aplicações e por todos os serviços da plataforma. Todos os acessos às bases de dados são feitos através da rede interna da ISA.

Tal como a base de dados geral do *iTelemetry*, também o Servidor *SMTP* é acedido por vários pontos da plataforma para envio de *e-mails*.

O servidor *Internet Information Services* (IIS) é um servidor *web* que contém as aplicações *web* da plataforma, que são o *iWater*, o *iTelemet*, o *iStock*, o *Service Manager* e o *Backoffice*. É através deste servidor que todas as aplicações utilizam o *Active Directory*, que é a funcionalidade que permite a autenticação numa aplicação *web* com a conta que está em sessão no computador. A aplicação *web Service Manager* acede à base de dados *ISA Scheduler*, e a aplicação *web Backoffice* acede ao *iCenter* para se sincronizarem no que diz respeito às configurações. Este acesso pode ser por *Reflection* de uma *assembly* ou por *WCF*, o qual é decidido nas configurações da aplicação.

O servidor *windows* contém os serviços internos (*Event Analyzer Service* e *Data Service*) e externo (*ISA Scheduler*) da plataforma, os seus processos (*Notification Dispatcher Service* e *AquaMatrix Exporter*) e uma *Message Queue* (*iTelemetryQueue*). O *ISA* Scheduler acede à sua base de dados para consultar e gerir os dados, utiliza um *Modem* para fazer chamadas telefónicas e um *Servidor SMTP* para o envio de *e-mails*. Para além disso, é este serviço que executa os processos *Notification Dispatcher* e *AquaMatrix Exporter*. O serviço *Data Service* trata da aquisição dos dados do *iCenter*, podendo aceder a estes de duas formas distintas: uma opção passa pela colocação destes dados na *iTelemetryQueue*, que notifica o *Data Service* quando são recebidos novos dados; e a outra opção, semelhante à forma como o *Backoffice* o faz, consiste no carregamento de uma *assembly* ou por WCF.

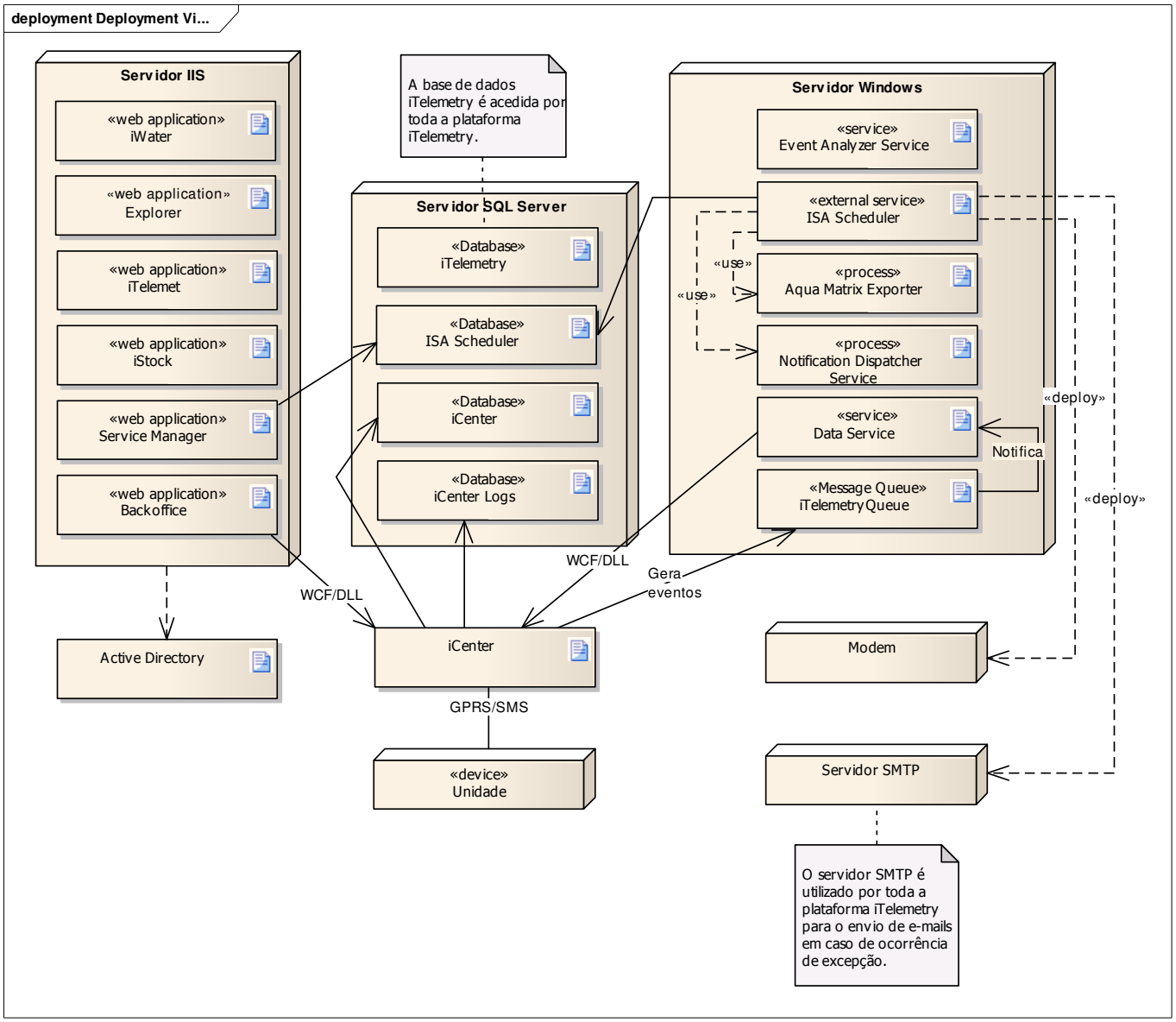


Ilustração 13 - Diagrama de instalação da Arquitectura Física do iTelemetry

O *iCenter*, para além de aceder às suas bases de dados, comunica com as Unidades, podendo esta comunicação ser estabelecida por GPRS ou por SMS.

4.3 Unidades suportadas

Uma unidade é um dispositivo físico constituído por vários canais, aos quais estão ligados sensores dedicados. As suas funções são as de armazenar e transmitir os valores medidos pelos sensores, que podem representar níveis, contagens, pressões, temperaturas ou protecção do nível catódico.

Contundo, nem todas as unidades estão preparadas para enviar todos os tipos de dados, sendo que algumas unidades são especializadas em certas funções. Exemplos disso são os *iWater*'s, os *iBottleRack*'s e os *iGasCylinder's*. Como exemplo de uma unidade completa, ou seja, que não é especializada, tem-se o *iLogger*.

Além dos tipos de valores já referidos, as unidades estão também preparadas para poder enviar o nível de bateria do próprio equipamento, e incorporam, ainda, tratamento de eventos de alarme, que podem ser de nível de conteúdo ou de intrusão.

Estes dispositivos podem comunicar por GSM, SMS e GPRS.

4.4 iCenter

O *iCenter* é uma plataforma independente do *iTelemetry*. Tem, como principal objectivo, suportar as comunicações com as unidades, que podem ser da ISA ou não, sendo que tais comunicações podem ser estabelecidas por SMS ou por GPRS.

Esta plataforma contém bases de dados onde guarda os dados recebidos pelas unidades. A partir destes registos, consegue verificar falhas nas transmissões de dados, assim como ignorar dados repetidos.

Estes dados são disponibilizados a várias plataformas, como é o caso do *iTelemetry*.

4.5 Serviços

Um serviço diz respeito a um processamento automático, ou seja, que pode ser executado sem a intervenção de um utilizador. No *iTelemetry* existem os seguintes serviços: o *Data Service*, responsável pela chegada de dados ao sistema; o *Event Rules Analyzer*, que se dedica ao accionamento de alarmes; o *Notification Dispatcher*, que envia notificações em forma de *e-mail*; e o *AquaMatrix Exporter*, o qual exporta dados da base de dados para ficheiros, com o formato pretendido. Segue-se a apresentação de todos estes serviços.

O *Data Service* é o serviço responsável pela sincronização de dados do *iCenter* com o *iTelemetry*. Através de uma *Message Queue*, ou acedendo ao *iCenter* por WCF ou por carregamento de uma *assembly*, o *Data Service* vai buscar novos dados e actualiza a base de dados do *iTelemetry*. Este serviço tem de garantir que não são perdidos dados a partir da comunicação do *iCenter* até estes estarem arquivados na base de dados.

Quanto ao *Event Rules Analyser,* é um serviço que analisa os dados recebidos, para verificar se se trata de uma situação de alarme. Para alarmar as pessoas responsáveis, é criada uma notificação, que é posteriormente enviada pelo *Notification Dispatcher*. Este serviço é executado sempre que chegam ao *iTelemetry* novos dados. É de referenciar que este é um serviço bastante importante para os clientes, uma vez que é a partir dele que surgem os avisos de situações críticas, tais como rupturas em depósitos ou fugas de gás.

O *Notification Dispatcher* é um processo gerido pelo serviço externo *ISA Scheduler*, responsável por enviar aos utilizadores do *iTelemetry* as notificações correspondentes. Não construindo as notificações, este processo limita-se a aceder às mensagens armazenadas na base de dados, controlá-las e enviá-las aos devidos destinatários, na forma de SMS ou *e-mail*. Além disto, é, ainda, responsável pelo controlo da recepção, por parte dos respectivos destinatários, das notificações enviadas.

Por último, o *AquaMatrix Exporter* descende de um engenho de exportação, responsável por exportar dados para ficheiros que podem ser de formato Excel, CSV ou XML. Para utilizar este engenho é necessário definir processos que organizem os dados que devem ser exportados, e que definam o formato de exportação. O *AquaMatrix Exporter* é um desses processos, e está definido para exportar um ficheiro XML periodicamente, tal como foi pedido por clientes, para se integrarem os dados do *iTelemetry* com outro *software* denominado *AquaMatrix*. Este processo é, também, gerido pelo serviço externo *ISA Scheduler*.

4.6 Aplicações

As aplicações do *iTelemetry* são *web*, ou seja, são aplicações cujo acesso do utilizador é feito na internet através de um navegador. Podendo os utilizadores serem clientes ou administradores da ISA, nem todas as aplicações estão disponíveis para ambos. O *Backoffice*, exemplo de aplicação exclusiva a administradores da ISA, serve, essencialmente, para configurações, o *Service Manager* trata a gestão de processos agendados, enquanto que as restantes tratam, de um modo geral, da apresentação de dados.

O *Backoffice* é uma aplicação que, sendo dirigida apenas aos administradores da plataforma *iTelemetry*, é responsável por permitir a criação, remoção ou edição de:

a) utilizadores e respectivas configurações de permissões;
b) locais de clientes e, ainda, a definição dos locais a que cada utilizador terá acesso;
c) perfis de clientes para cada aplicação;
d) unidades e elementos; e, ainda, a associação entre ambas as entidades.

Também nesta aplicação é disponibilizada a visualização do estado de comunicação das unidades de todo o *iTelemetry*, assim como a apresentação dos *logs* registados em toda a plataforma. Esta é uma aplicação onde é se encontram todas as configurações existentes no *iTelemetry*, o que a torna restrita quanto aos utilizadores que a ela podem ter acesso.

O *Service Manager* é uma aplicação que gere todos os processos que o *ISA Scheduler* executa, podendo aqui agendar várias execuções, assim como verificar os resultados de cada execução. É ainda possível instalar novos processos no *ISA Scheduler* e parametrizar determinadas configurações nestes.

Quanto ao *iStock*, a sua função passa por monitorizar os *racks* de botijas de gás, permitindo aos clientes (como são exemplo os distribuidores destas botijas) o acesso a relatórios de estado de cada um dos seus pontos de venda ao público. A aplicação contém um mapa com a distribuição de *racks* de botijas de gás, que permite aos utilizadores uma melhor análise dos reabastecimentos a efectuar.

O *iWater* é uma aplicação de visualização de dados recolhidos através das unidades do tipo *iWater*. Estas unidades são dedicadas à monitorização de medições ligadas ao negócio das águas. Com esta aplicação, os clientes têm acesso às medições efectuadas nas suas condutas, que podem ser vizualizadas em gráficos, por forma a oferecer aos utilizadores uma mais fácil leitura dos dados.

Já o *Explorer*, que é mais uma aplicação de apresentação de dados, é semelhante ao *iWater*, mas não é restrito quanto ao tipo de unidades, podendo apresentar dados vindos de qualquer unidade e que correspondam a qualquer tipo de medições. As funcionalidades do *iWater* dedicadas, exclusivamente, ao negócio de telemetria de água não existem no *Explorer*, uma vez que, aqui, deixam de fazer sentido.

Por fim, resta o *iTelemet*, que se encontra, já há algum tempo, fora de uso. Foi uma aplicação desenvolvida para integração com unidades do tipo *Telemet*, que actualmente já não são usadas.

Capítulo 5 Trabalho desenvolvido

Este capítulo é dedicado à exposição detalhada de todas as tarefas efectuadas ao longo do estágio. Apesar de não ter sido a primeira a ser executada, a tarefa *ISA Scheduler* é apresentada em primeiro lugar, devido à necessidade da sua compreensão para a explicação do *Service Manager*, tarefa que se segue. A reformulação da arquitectura lógica surge seguidamente, à qual sucede a validação do sistema. Além destas tarefas, ditas principais, foram, ainda, realizados outros trabalhos de menor relevância, detalhados no último subcapítulo.

5.1 ISA Scheduler

O *ISA Scheduler* é uma ferramenta da ISA que tem como objectivo executar determinados processos, com uma periodicidade configurada. Esta regularidade é definida através de um agendamento que pode ser, por exemplo, diariamente às 04:00.

O *ISA Scheduler* foi, ao início, pouco modelado, por ser uma ferramenta auxiliar do *MaisGas* e servir, apenas, para uso interno. A sua primeira versão continha os agendamentos definidos em XML, mas a complexidade na definição destes agendamentos levou à necessidade de centralizar tudo numa base de dados.

Com o passar do tempo, e com cada vez mais utilidade, esta ferramenta tornou-se muito importante, tanto para o *iTelemetry*, como para o *MaisGas*. Esta importância fez com que o *ISA Scheduler* necessitasse de uma reformulação, para responder a novos requisitos, e para se tornar mais fiável. Acompanhando esta reformulação, houve também modelação e criação de documentação, que suportam tanto as implementações já existentes, como as novas implementações.

Os processos estão armazenados como *assemblies* que têm de respeitar determinadas regras, para que o *ISA Scheduler* possa carregá-los e corrê-los.

De seguida são apresentadas as suas funcionalidades, a modelação efectuada, as decisões tomadas ao longo da sua implementação, e, ainda, a contribuição do estagiário para esta tarefa.

5.1.1 Funcionalidades

Geral

A estrutura base do *ISA Scheduler* está definida na Ilustração 14.

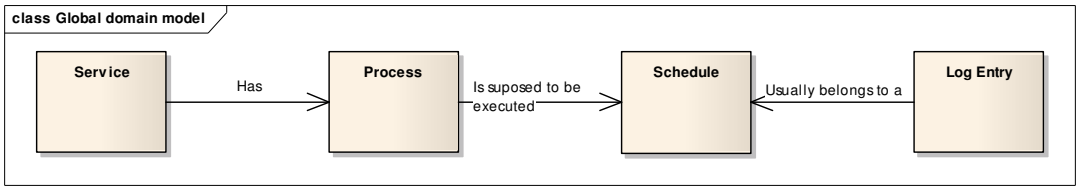


Ilustração 14 - Relação dos Serviços, Processos, Agendamentos e *Logs* do *ISA Scheduler*

Cada instância do *ISA Scheduler* denomina-se Serviço, o qual vai gerir os processos a si associados. Este serviço tem que ter indicações do servidor onde está a correr e da directoria que gere.

Um processo representa uma *assembly,* que pode ser executada pelo *ISA Scheduler*. Para isso acontecer, a *assembly* tem que conter uma classe, da qual o *ISA Scheduler* tem que conhecer a estrutura. Assim sendo, foi definida uma classe abstracta, de nome *AScheduledProcess*, que o *ISA Scheduler* conhece e da qual a classe do processo tem de descender para poder ser reconhecida pelo *ISA Scheduler*. Esta classe do processo, através de *Reflection*, pode ser reconstruída e executada pelo *ISA Scheduler*, sendo que, para isso, é necessário que a *assembly* esteja acompanhada das suas dependências (outras *assemblies*), caso existam. Todas estas *assemblies* devem estar organizadas numa pasta, que tem de estar depositada na directoria que o respectivo serviço gere.

Coerente com a classe *AScheduledProcess*, a classe do processo tem que disponibilizar: um método responsável por preparar a execução, de nome *OnBeforeExecution*; um método responsável pela própria execução, chamado *ExecuteProcess*; e uma propriedade que recebe configurações em formato XML, de nome *Parameters*. Estas configurações são definidas num atributo do processo, com o nome *GlobalParameters*.

Um processo tem, também, a possibilidade de conter configurações de notificação, as quais são compostas por: tipo de notificação, que pode ser por *e-mail* ou por chamada telefónica; subscritor, que identifica a pessoa que vai receber a notificação; modem, de onde serão feitas as chamadas caso o tipo de notificação seja chamada telefónica; e acontecimento (na execução do processo) que gera a notificação, podendo ser um erro, um aviso ou um registo regular.

Um agendamento está associado a um processo, e define quando é que o processo deve ser executado, através de uma periodicidade que pode ser do tipo mensal, semanal, diária ou apenas entre um determinado intervalo de segundos. Apesar de parecer um número limitado, estes tipos de periodicidade conseguem contemplar vários esquemas. Por exemplo, caso se queira um processo que ocorra duas vezes por dia, às 8h e às 14h, podem, facilmente, definir-se dois agendamentos diferentes, ambos do tipo diário, mas cada um com a sua hora definida.

Cada agendamento possui vários atributos. Além da data da última execução efectuada (através deste agendamento), tem, também, a data da próxima execução do processo, calculada no final de cada execução. Repare-se que, quando uma dada execução fica presa, a data da próxima não será calculada, facto que teve de ser ultrapassado. Para isso, criou-se o atributo que calcula o tempo máximo esperado em cada execução e cuja existência serve para ter um *timeout* que, depois de ultrapassado, obriga ao cálculo da data/hora da próxima execução. Deste modo, é possível que as execuções posteriores ao erro sejam efectuadas. Tal como um processo, um agendamento tem, ainda, um atributo de Parâmetros. Serve para, dependendo do agendamento, definir configurações específicas para a execução do processo. Considere-se, por exemplo, um processo que gera relatórios. Este pode estar definido para, diariamente, às 00h00, gerar um relatório, apenas com os dados do cliente A e, também diariamente, às 02h00, gerar um relatório, apenas com os dados do cliente B. Assim sendo, definem-se dois agendamentos com parâmetros diferentes que especificam o cliente para o qual se quer gerar um relatório.

Um *log* é um registo, que pode referenciar vários tipos de eventos:

- carregamento de execução de processo;
- início de execução de processo;
- acções na execução de processo;
- fim de execução de processo;
- tentativa de envio de notificação;
- acção de um utilizador;
- acontecimento regular.

Independentemente do tipo de evento, cada *log* tem que ter a data/hora do registo, a mensagem a registar, uma referência ao serviço ao qual está associado e o tipo de *log*, que pode ser regular, de aviso ou de erro. Em caso do *log* ser um registo relativo a uma execução, tem que existir, ainda, associação ao agendamento que a gerou. O semelhante ocorre nos *logs* que registam uma tentativa de envio de notificação, os quais têm de estar associados a uma configuração de notificação.

É importante, ainda, que o utilizador possa assinalar um *log* como verificado, por forma a não perder tempo na leitura de *logs* já analisados. Assim sendo, quando um *log* é considerado verificado, tem que ter a data de verificação, assim como a referência do utilizador que fez a verificação.

Execução de processos

Os processos, conforme a estrutura da classe *AScheduledProcess*, têm um método de nome *OnBeforeExecution* e outro de nome *ExecuteProcess*. Respeitando esta estrutura, o *ISA Scheduler* está preparado para, antes de executar o processo (*ExecuteProcess*), preparar esta execução (*OnBeforeExecution*).

Capaz de executar vários processos ao mesmo tempo, o *ISA* Scheduler, entre certos intervalos de tempo, analisa os que devem ser executados, tendo em conta as periodicidades presentes nos respectivos agendamentos. No entanto, também é possível forçar execuções, através dos agendamentos e ignorando a periodicidade estabelecida anteriormente.

É importante que qualquer acontecimento de uma execução, natural ou imprevisto, seja registado como *log*, para que, caso haja necessidade de analisar a execução, existam provas do que sucedeu.

No final da execução, a data/hora da próxima execução de um agendamento deve ser calculada. Caso a execução tenha ficado presa, o tempo máximo esperado de execução será esgotado. Quando isto acontecer, a data/hora da próxima execução também será calculada.

Reflection de assemblies

Tal como já foi referenciado, um processo tem de ter a si associado uma *assembly*, que deve ser composta por uma classe descendente da classe abstracta *AScheduledProcess*. Através de *Reflection*, é necessário que o *ISA Scheduler* consiga carregar a *assembly*, para que o processo possa ser executado.

Notificações

O *ISA Scheduler* é capaz de enviar notificações por *e-mail* ou por chamada telefónica. Enquanto que as notificações enviadas por chamada telefónica são efectuadas sem voz, alertando, apenas, o subscritor, as notificações por *e-mail* tornam-se mais completas, por possuírem uma mensagem. No entanto, comparando os dois tipos de notificação, a por chamada telefónica contempla uma maior capacidade de alerta, visto que a leitura de *e-mails* pode ser facilmente adiada ou ignorada, ao contrário das chamadas.

O envio de uma notificação é sempre registado, mesmo quando não é conseguido. Este registo inclui quer a mensagem da notificação, quer a informação sobre o sucesso ou insucesso do envio.

Sempre que o envio de uma notificação falha, existe, passado um certo tempo, uma nova tentativa de envio. No entanto, para as tentativas não serem em número infinito, existe uma configuração que define o número limite de tentativas.

Já houve preparação do código para a inserção de notificações por SMS, mas esta funcionalidade ainda não está implementada.

5.1.2 Modelação

Para um melhor entendimento sobre o *ISA Scheduler*, são detalhadas, se seguida, as sequências dos seus pontos fulcrais.

Sequência de execução

A Ilustração 15 mostra a sequência de execução dos processos que, num certo momento, necessitam de ser executados.

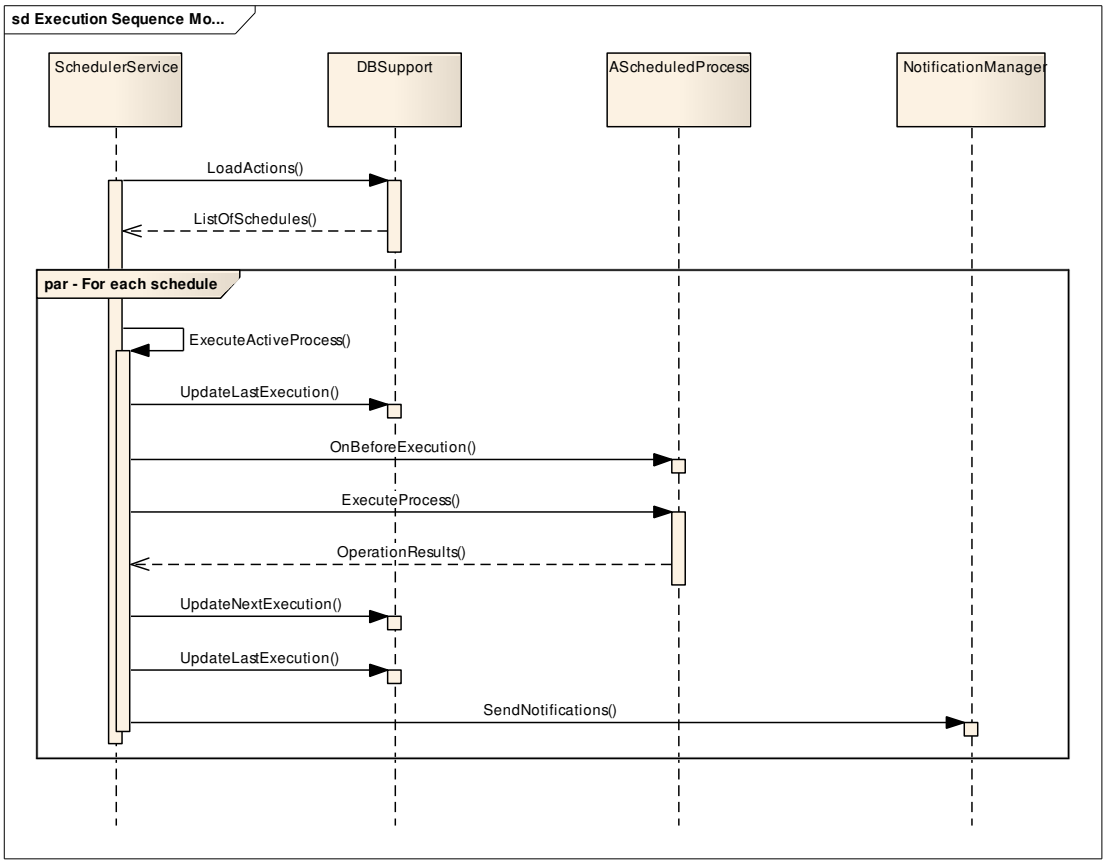


Ilustração 15 - Diagrama de Sequência das Execuções dos Processos

Depois de pedir à base de dados os agendamentos cujos processos devem ser executados, procede-se à execução paralela de todos eles. Como explicado nas funcionalidades, o método *OnBeforeExecution* prepara a execução, enquanto o método *ExecuteProcess* é responsável pela própria execução e pelo retorno do conjunto de resultados originados por esta.

O facto da data/hora da última execução ser actualizado duas vezes tem um objectivo específico. Enquanto que a actualização depois de uma execução é considerada "normal", a actualização antes de preparar a execução serve para assinalar que o processo de agendamento já está a ser executado.

No final, além de actualizar a data/hora da última execução, calcula-se a data/hora da próxima execução e enviam-se as notificações respeitantes à presente.

Sequência de reenvio de notificações

O *ISA Scheduler* está preparado para reenviar notificações falhadas. A sequência desta acção está demonstrada na Ilustração 16.

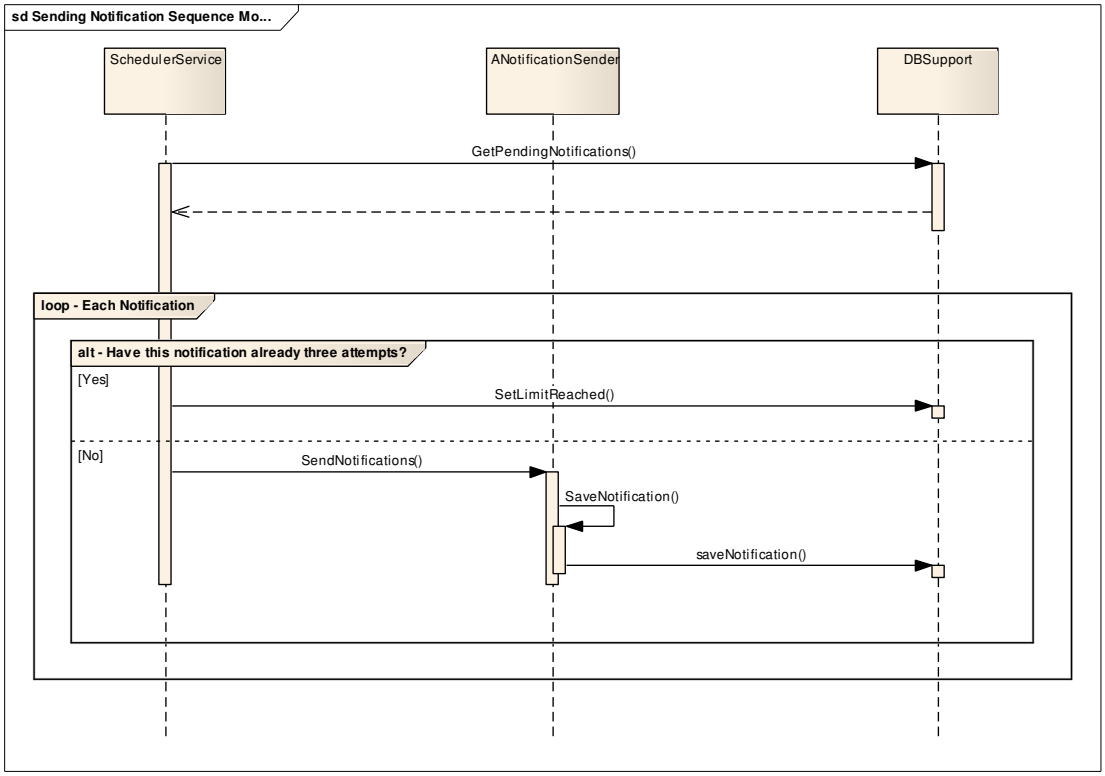


Ilustração 16 - Diagrama de Sequência do Tratamento das Notificações Pendentes

Para não sobrecarregar o modelo da base de dados, optou-se por ir buscar aos *logs* as notificações cujo envio tenha falhado. Obviamente, são excluídas deste conjunto as notificações cujo envio só foi conseguido após várias tentativas.

O conjunto devolvido pela base de dados denomina-se por notificações pendentes. Para cada notificação pendente, há uma nova tentativa de envio seguida de novo registo da notificação, enquanto o limite de tentativas não é atingido. Se esse limite for atingido, é, então, criado um *log* que regista tal acontecimento, impedindo, assim, que a notificação seja considerada novamente como pendente.

Sequência de avaliação de agendamento

A Ilustração 17 define a sequência de avaliação de um agendamento. Esta avaliação depende dos valores atribuídos às datas da última e da próxima execuções, que podem ou não ser nulos.

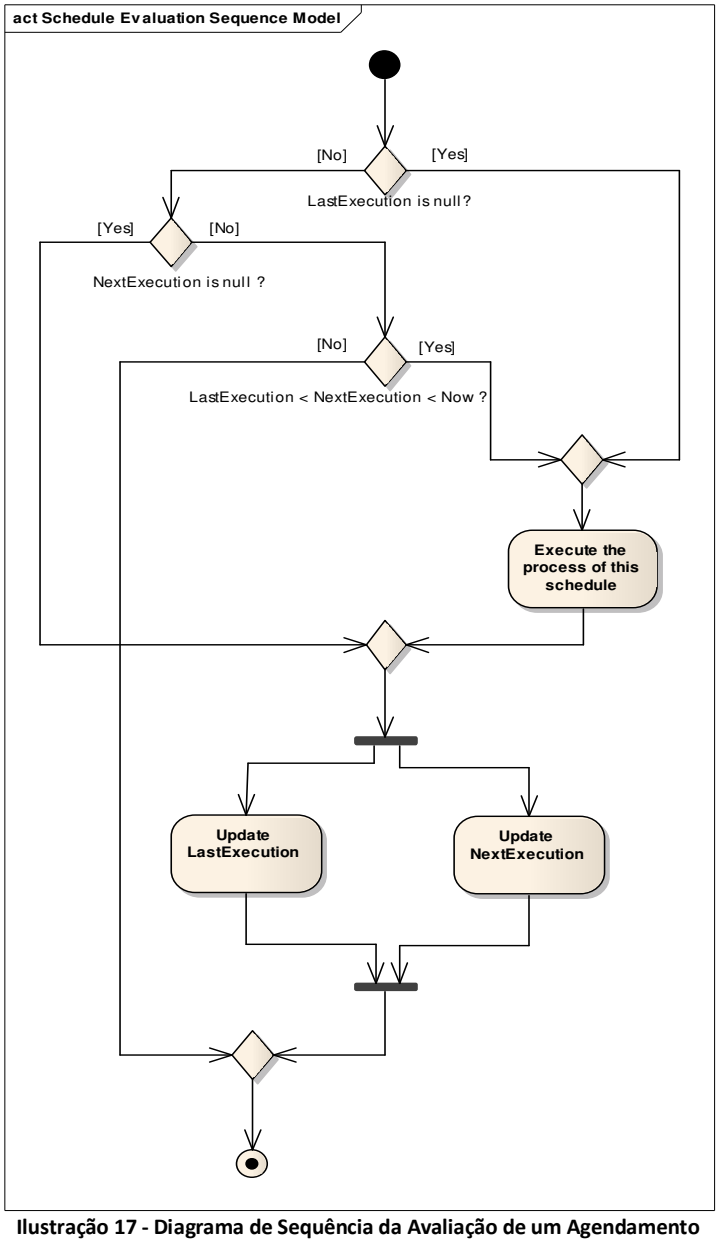


Ilustração 17 - Diagrama de Sequência da Avaliação de um Agendamento

Quando um agendamento está com o atributo da última execução a nulo, significa que a sua execução foi forçada e, portanto, tem de ser executado. Caso o atributo da última execução não seja nulo, mas o da próxima execução seja, significa que precisa de ser calculada a data da próxima execução. Este caso acontece quando o agendamento é criado, ou quando o serviço inicia. Em ambos os casos, o primeiro objectivo passa por calcular quando será a próxima execução.

Quando nem o atributo da última execução, nem o atributo da próxima execução são nulos, podem acontecer três situações:

- a data/hora da próxima execução ainda não foi ultrapassada: ainda não chegou o momento da execução e, portanto, nada acontece ao agendamento;
- a data/hora da próxima execução é inferior à data/hora da última execução: significa que o processo do agendamento ainda está em execução e, mais uma vez, nada acontece ao agendamento;
- a data/hora da próxima execução já foi ultrapassada, e é posterior à data/hora da última execução: o processo do agendamento deve ser executado.

Depois de cada execução, os atributos das última e próximas execuções são actualizados.

5.1.3 Implementação

Factory Method

No módulo das notificações, foi implementada a *Design Pattern Factory Method*, cujo processo teve como base a referência [7].

O objectivo que leva ao uso desta *Design Pattern* é a dividisão das notificações por emissores específicos, tendo em conta o tipo de notificação (*e-mail*, chamada telefónica ou SMS). Os emissores são criados pelo *Factory Method.*

Nesta implementação, foi usado um *ConcreteCreator*, ao qual se chamou *NotificationSenderFactory*. Por este ser único e para não criar redundância, não foi criado um *Creator* abstracto, e, portanto, este *ConcreteCreator* é instanciado directamente.

Outro facto a notar na implementação deste *Factory Method* é a criação de vários *Products* num só *ConcreteCreator*, de acordo com o segundo ponto da secção "Implementação" da referência [7].

Na Ilustração 19 está definido o diagrama de classes desta implementação, acompanhada, na Ilustração 18, pela estrutura geral de um *Factory Method*.

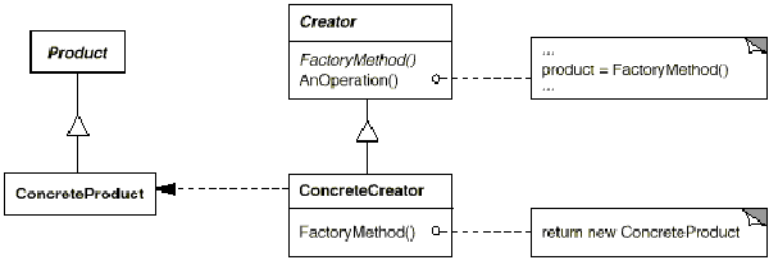


Ilustração 18 - Estrutura Geral da *Design Pattern Factory Method*

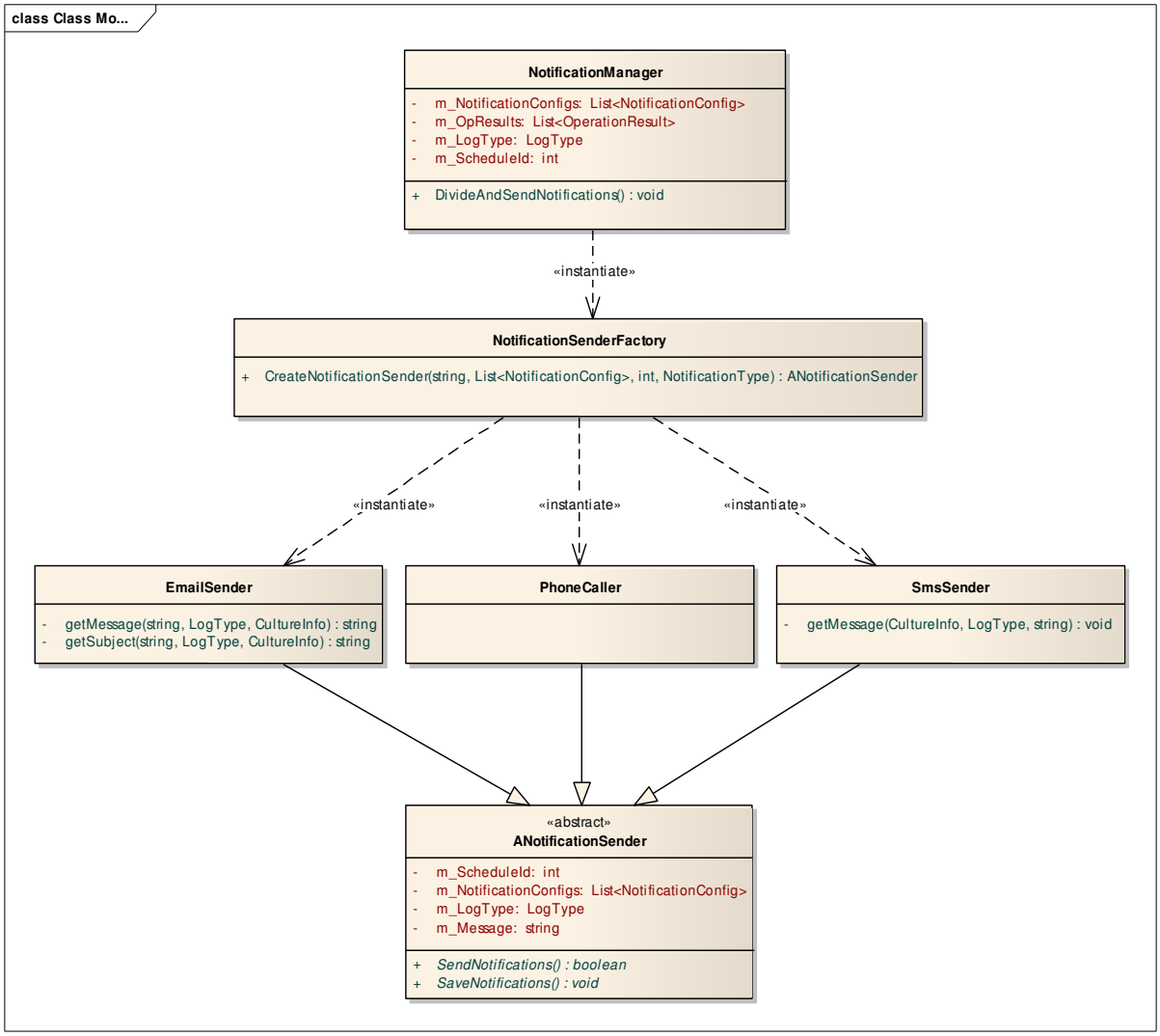


Ilustração 19 - Diagrama de Classes da Criação de Notificações

Fiabilidade

Para combater os bloqueios nos processos, foram adicionados tratamentos de excepção ("*try-catch*"), que impedem que as execuções fiquem paradas. Para além disso, a ferramenta foi preparada para fazer várias tentativas de acesso à base de dados, sempre que este falhar. Isto faz com que, caso a rede tenha problemas durante alguns segundos, os serviços esperem até que estes problemas sejam ultrapassados. Assim, há menos risco de um processamento correr mal. No entanto, o tempo em que a rede não está operacional pode ultrapassar o tempo de espera e, por isso, a execução do serviço deve ser parada. Como esta paragem não pode dar origem a bloqueios, houve, mais uma vez, o cuidado de tratar as excepções. Todos estes acontecimentos são anotados, para que, mais tarde, se saiba o que aconteceu.

5.1.4 Participação do estagiário

Dado que a elaboração de documentos relativos a esta ferramenta estava atrasada, o estagiário ocupou-se de todo o processo de modelação e estruturação de requisitos.

Para além disto, teve que ser feita uma remodelação das classes, de forma a tornar o *ISA Scheduler* mais escalável e com uma mais fácil percepção. Esta remodelação foi debatida entre a equipa, sendo que o importante papel do estagiário verificou-se, quer nesta discussão, quer na implementação.

O módulo de notificações por chamada telefónica foi, todo ele, desenvolvido pelo estagiário.

Foram, ainda, feitas uma revisão e uma re-implementação de alguns dos passos mais importantes, por forma a tornar o processo do *ISA Scheduler* menos susceptível a ruptura.

5.2 Service Manager

O *Service Manager* é uma aplicação *web* que corre sobre a plataforma *iTelemetry* e organiza os dados tratados pela ferramenta *ISA Scheduler*.

Esta aplicação permite, não só a verificação do estado dos processos e agendamentos do *ISA Scheduler*, como também a monitorização destes, tornando-o mais manipulável e mais orientado ao utilizador. Deste modo, através do *Service Manager*, é mais fácil para o utilizador perceber e gerir o que está a acontecer no *ISA Scheduler*.

Com o aumento da importância do *ISA Scheduler* para várias plataformas da ISA, a importância do *Service Manager* aumentou de igual forma. Neste subcapítulo começa-se por explicar as funcionalidades desta aplicação, sendo, de seguida, apresentadas as decisões tomadas no protótipo e ao longo da implementação.

5.2.1 Funcionalidades

Apresentação de serviços

O *Service Manager* disponibiliza uma tabela de serviços do *ISA Scheduler*. Esta é sujeita a uma filtragem e pode ser ordenada tanto pelo nome do serviço, como pelo nome do servidor onde este se encontra a correr.

Os requisitos referentes ao módulo dos serviços estão numerados com os números de 2 a 6 do Anexo II - Service Manager - Requisitos.

Apresentação de processos

Noutra secção da aplicação, encontra-se disponível ao utilizador uma tabela de processos. Como foi referido no subcapítulo 5.1.1 referente às funcionalidades do ISA Scheduler, um processo tem de pertencer a um serviço. Para que o utilizador reconheça cada processo, existe uma descrição detalhada associada a cada um deles. Além disso, cada processo é acompanhado, na tabela, por uma imagem ilustrativa do estado das execuções do mesmo, que pode ser identificativo de:

- não terem existido ainda execuções;
- todas as execuções terem sucedido com sucesso;
- existir, pelo menos, um aviso nas execuções, e não existirem erros nestas;
- existir, pelo menos, um erro nas execuções.

A tabela de processos pode ser filtrada por uma dada cadeia de selecções, que começa na selecção de um servidor. De seguida, são disponibilizados, noutra lista, os serviços do servidor escolhido. Enquanto não é feita a selecção de um dado serviço, aparecem, na tabela, todos os processos de todos os serviços do servidor escolhido, sendo que, quando um serviço é escolhido, são listados apenas os processos desse serviço. A hierarquia está demonstrada na

Ilustração 20.

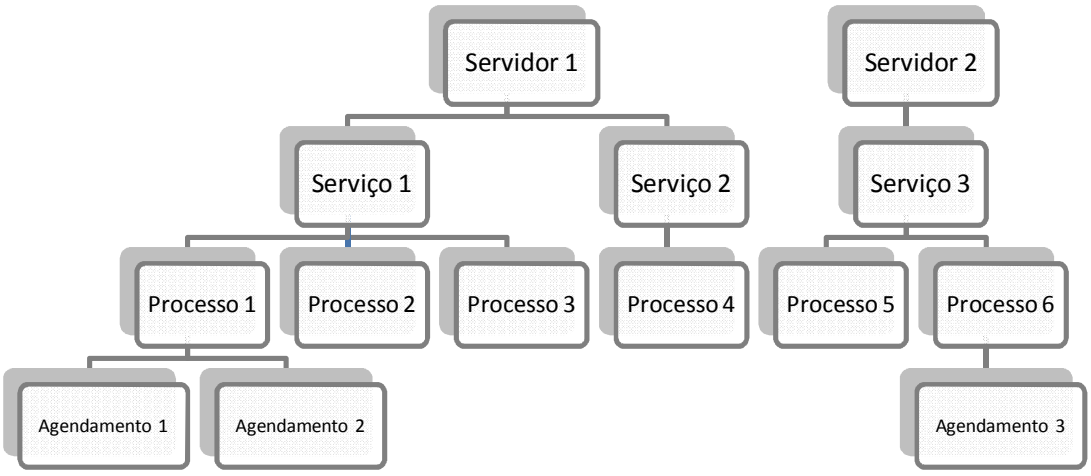


Ilustração 20 - Representação da Hierarquia entre Servidores, Serviços, Processos e Agendamentos

A restante informação de cada processo é disponibilizada numa nova página que pode ser acedida por um botão que acompanha o processo na tabela. Esta informação contém, entre outros, a localização da *assembly* que é executada pelo *ISA Scheduler*, e as configurações das notificações relativas ao processo.

Os requisitos sobre a secção dos processos estão referenciados no Anexo II - Service Manager - Requisitos, indicados entre os números 7 e 12.

Gestão de processos

É possível, no *Service Manager*, a instalação/inserção, a edição e a remoção de processos.

Para a instalação, é necessário ter um ficheiro zipado com as *assemblies* do processo. Uma dessas *assemblies* deve seguir as regras definidas no *ISA Scheduler*, para ser possível a execução do processo. Depois do carregamento do ficheiro zipado, é necessária a indicação dos seguintes parâmetros:

- nome;
- descrição;
- serviço ao qual o processo pertence;
- parâmetros globais utilizados (dependendo da implementação do processo) durante a execução;
- caminho da pasta destino onde ficará o processo;
- configurações das notificações.

Ao confirmar a inserção do processo, o conteúdo do ficheiro zipado é colocado no servidor onde está alojado o serviço do processo.

A edição de processos não permite a alteração das *assemblies*. Assim, quando um utilizador selecciona o botão de edição de um processo, só pode alterar os campos descritos acima. A alteração do caminho da pasta destino onde ficará o processo altera as *assemblies* de local, mantendo-os no mesmo servidor. Da mesma forma, é possível mudar o serviço ao qual um

dado processo pertence, transferindo as *assemblies* desse processo para o servidor do novo serviço.

Além destas acções, os processos podem, ainda, ser removidos, o que implica remoção das *assemblies* do servidor e, respectivamente, dos agendamentos associados ao processo. Os registos de execução associados a estes agendamentos não são removidos, mas deixam de estar associados aos mesmos.

Os requisitos que dizem respeito aos acessos rápidos estão definidos no Anexo II - Service Manager - Requisitos, marcados com os números de 13 a 17 e de 19 a 22.

Apresentação de agendamentos

Os agendamentos representam outra secção do *Service Manager*, e encontram-se, também, listados numa tabela. Um agendamento tem a si associado a data da última execução, a data calculada da próxima execução e a sua periodicidade. Para além disso, um agendamento tem de estar associado a um processo. Logo, pela hierarquia demonstrada na Ilustração 20, está associado a um serviço e a um servidor.

É possível filtrar a tabela de agendamentos segundo vários aspectos:

- servidor;
- serviço, cuja lista é actualizada de acordo com o servidor seleccionado;
- processo, cuja lista é actualizada de acordo com o serviço seleccionado;
- tipo de periodicidade, que pode ser:
  - nunca,
  - de x em x segundos,
  - diário,
  - semanal,
  - mensal;
- data da última execução;
- data da próxima execução.

Tal como acontece em cada processo, cada registo de agendamento é acompanhado, na tabela, por uma imagem ilustrativa do estado das execuções.

Um agendamento tem, também, outros atributos, que estão armazenados noutra página, que pode ser acedida através de um botão que acompanha cada agendamento. A informação armazenada nesta página diz respeito a:

- parâmetros utilizados (dependendo da implementação do processo) durante a execução do processo, que permitem dar diferentes configurações para diferentes agendamentos do mesmo processo;
- uma ilustração que permite uma melhor visualização da periodicidade do agendamento;
- o tempo máximo que se espera que o agendamento demore a executar.

É, também, a partir desta tabela que se acede aos registos das execuções referentes a um dado agendamento. Para isso, existe um botão em cada linha da tabela que direcciona o utilizador para a secção de registo de execuções.

Os requisitos referentes ao módulo dos agendamentos são os numerados com os números de 23 a 29 do Anexo II - Service Manager - Requisitos.

Gestão de agendamentos

O *Service Manager* permite a inserção, a edição e a remoção de agendamentos.

Na inserção de um agendamento, é necessário definir o processo ao qual o agendamento está associado, a periodicidade do agendamento, os parâmetros e o tempo máximo esperado para a execução.

Qualquer agendamento pode ser editado. Esta edição pode incluir a alteração de qualquer um dos campos definidos no parágrafo anterior.

A remoção de um agendamento implica a desassociação dos registos de execução ao mesmo agendamento.

Os requisitos sobre gestão de agendamentos estão indicados com os números 18, 30 e 31 no Anexo VI - Requisitos Service Manager.

Apresentação dos registos de execução

A última secção trata os registos de execução. Aqui, encontra-se uma tabela com os registos de execução, que pode ser preenchida de duas formas:

- utilizando o botão presente em cada linha da tabela de agendamentos, que dá acesso aos registos de execução apenas do referido agendamento;
- utilizando o botão presente na secção de registos de execução, que dá acesso a todos os registos de execução.

Cada registo de execução é definido pela sua data/hora e por uma mensagem que explica o registo. Existe, também, um tipo de evento, que indica se o registo é de início da execução, de fim da execução, de processamento da execução, de regularidade ou de notificação.

O registo tem, ainda, uma indicação do tipo de registo, ou seja, se é de normalidade, aviso ou erro, e, na tabela de registos de execução, este campo é acompanhado de uma ilustração. Essa mesma tabela permite filtração por tipo de evento, por tipo de registo e por data/hora do registo de execução.

Para que um utilizador não perca tempo, por exemplo, com erros já verificados, é possível assinalar a verificação de registos, os quais vão conter os dados referentes ao responsável pela verificação e a data da mesma. Para além disso, a verificação de um registo de execução resulta na verificação de todos os registos mais antigos que ainda não tinham sido sinalizados como verificados.

Quando estão listados os registos de execução de um só agendamento, a identificação desse mesmo agendamento encontra-se como subtítulo. Caso estejam listados todos os registos, cada linha da tabela tem, também, o nome do processo executado. Neste campo, quando um registo não pertence a nenhum agendamento, aparece informação de que não há nenhum processo associado.

Com os números de 32 a 37 do Anexo II - Service Manager - Requisitos, estão numerados os requisitos referentes a registos de execuções.

Acessos rápidos entre secções

Existem, ainda, três atalhos entre secções:

- dos serviços para os agendamentos;
- dos serviços para os processos;
- dos processos para os agendamentos.

Tratar-se-á à frente, apenas, o primeiro exemplo, visto que os restantes são semelhantes.

Para aceder aos agendamentos a partir dos serviços, existe um botão em cada linha da tabela de serviços, associado ao serviço apresentado na respectiva linha. O utilizador é redireccionado, directamente, para a secção dos agendamentos, onde se apresentam na tabela apenas os agendamentos pertencentes ao serviço seleccionado. Deste modo, já não é necessária a definição dos filtros para visualização de uma tabela mais reduzida e directa.

Os requisitos que dizem respeito aos acessos rápidos estão definidos no Anexo II - Service Manager - Requisitos, com os números 2, 3 e 7.

5.2.2 Protótipo

Uma decisão que se estendeu desde a primeira até à última versão do protótipo foi a de manter quatro secções na aplicação, acessíveis por separadores. Tais secções são:

- serviços;
- processos;
- agendamentos;
- registos de execução.

Serviços

A secção dos serviços é a mais simples, uma vez que apresenta uma só vista. Durante a implementação, surgiu a hipótese de permitir, remotamente, iniciar e parar um serviço. Chegou a ser feito o protótipo para esta ideia, criando uma segunda vista desta secção, mas acabou por ser posta de parte, devido à não necessidade e à complexidade do controlo remoto.

Processos

No início, a secção dos processos tinha duas vistas possíveis. Uma delas era a apresentação dos dados em tabela, e a outra a vista de detalhes. Logo na segunda versão do protótipo, houve uma alteração na vista de detalhes, disponibilizando os parâmetros de *e-mail* (que na altura existiam no lugar das notificações de configuração) numa tabela. Na primeira versão, estes apareciam em forma de texto no seu formato original, ou seja, XML. Esta alteração foi possível nos parâmetros de *e-mail* porque seguem sempre um determinado *schema*. Quanto aos parâmetros globais, apesar de também terem o formato XML, não é possível a apresentação em tabela, já que não seguem um dado *schema*, ou seja, têm uma estrutura livre.

Entretanto, com o surgirmento de novos requisitos, deu-se, aos utilizadores, a possibilidade de instalarem, editarem e removerem processos. Houve, com isto, necessidade de criar uma nova vista, onde é possível ao utilizador a inserção e a edição dos dados relativos aos processos. Também a vista de apresentação teve de sofrer alguns ajustes, por forma a incluir botões de inserção, edição e remoção dos processos.

Agendamentos

Os agendamentos estão inseridos numa secção que tem, desde o primeiro protótipo, uma vista de apresentação dos dados numa tabela, e uma vista que apresenta os detalhes de cada agendamento. Quando se decidiu colocar as funcionalidades de inserção, edição e remoção de processos e agendamentos, decidiu-se, também, adicionar uma terceira vista, dedicada à inserção ou edição de dados de um agendamento. Com isto, surgiu a necessidade de, mais uma

vez, se colocarem novos botões na vista de tabela relativos à inserção, edição e remoção de agendamentos.

Mais tarde, surgiu, ainda, a ideia de incluir um botão de actualização da tabela.

Resultados de execução

Inicialmente, a secção de resultados de execução tinha apenas a funcão de apresentar os registos das execuções. Mais tarde, surgiu uma proposta para a possibilidade de verificação dos registos no *Service Manager*. Com estes novos requisitos, foi necessário inserir um novo botão em cada linha da tabela de resultados de execução, que permite, ao utilizador, assinalar como verificado o resultado de execução da respectiva linha.

Um botão de actualização foi, também, proposto, para que o utilizador possa actualizar a tabela com os novos registos.

5.2.3 Implementação

Carregamento de resultados de execução

Durante a implementação detectou-se um problema: o número de registos carregados para a tabela de resultados de execução era enorme, levando a que o tempo gasto em cada actualização da tabela fosse elevado.

Como solução, passou a utilizar-se um *ObjectDataSource*, controlo da tecnologia ASP.NET que permite a paginação do lado do servidor. Definiu-se, para isso, um método que retorna os dados apenas de uma dada página, e outro que retorna o número total de dados, por forma a que a tabela saiba quantas páginas deve disponibilizar. O uso destes métodos obriga a mais cuidado, principalmente, na selecção dos dados correctos referentes a cada página. Um outro cuidado a ter reside na ordenação dos dados, que a tabela também suporta, e que pode ser uma fácil fonte de erros quando conjugado com o novo controlo *ObjectDataSource*.

Apesar de, mesmo assim, a *query* à base de dados ser extensa e complexa, notou-se que a actualização da tabela ficou mais rápida, depois de se começar a usar este controlo.

Estados das execuções dos agendamentos

Em cada agendamento há uma imagem representativa do estado das respectivas execuções, que pode representar diferentes situações:

- existência de erros, quando há, pelo menos, um erro nos novos registos de execução;
- existência de avisos, quando há, pelo menos, um aviso nos novos registos de execução, e não há erros;
- correu tudo bem, quando os novos registos não têm erros nem avisos;
- não existem novos registos, quando não há registos por verificar.

Para cada agendamento, pensou-se na utilização do *Design Pattern State*, onde se incluiriam os estados já descritos das execuções. Isto permitiria que os agendamentos fossem actualizados para um novo estado, sempre que surgissem novos registos ou ocorresse verificação de registos. No entanto, verificou-se que este *Design Pattern* não era adaptado, e que faria mais sentido existir uma actualização da tabela de agendamentos com nova *query* à base de dados, aquando da existência de alterações na tabela de registos de execução.

5.3 Reformulação da arquitectura lógica

Nem sempre é possível prever, à partida, a melhor forma de arquitectar uma plataforma. Como tal, caso se verifiquem possíveis melhorias, e elas se justifiquem, deve ocorrer uma reformulação da arquitectura. No *iTelemetry*, sentiu-se necessidade de efectuar uma reformulação da arquitectura lógica.

Ao longo dos próximos subcapítulos vão ser explicados os pontos mais importantes da nova arquitectura lógica, que é apresentada na Ilustração 22. Esta explicação faz-se acompanhar, também, da comparação com a anterior arquitectura lógica, representada na Ilustração 21, sendo que, o que não se alterou, já foi explicado no subcapítulo 4.1.

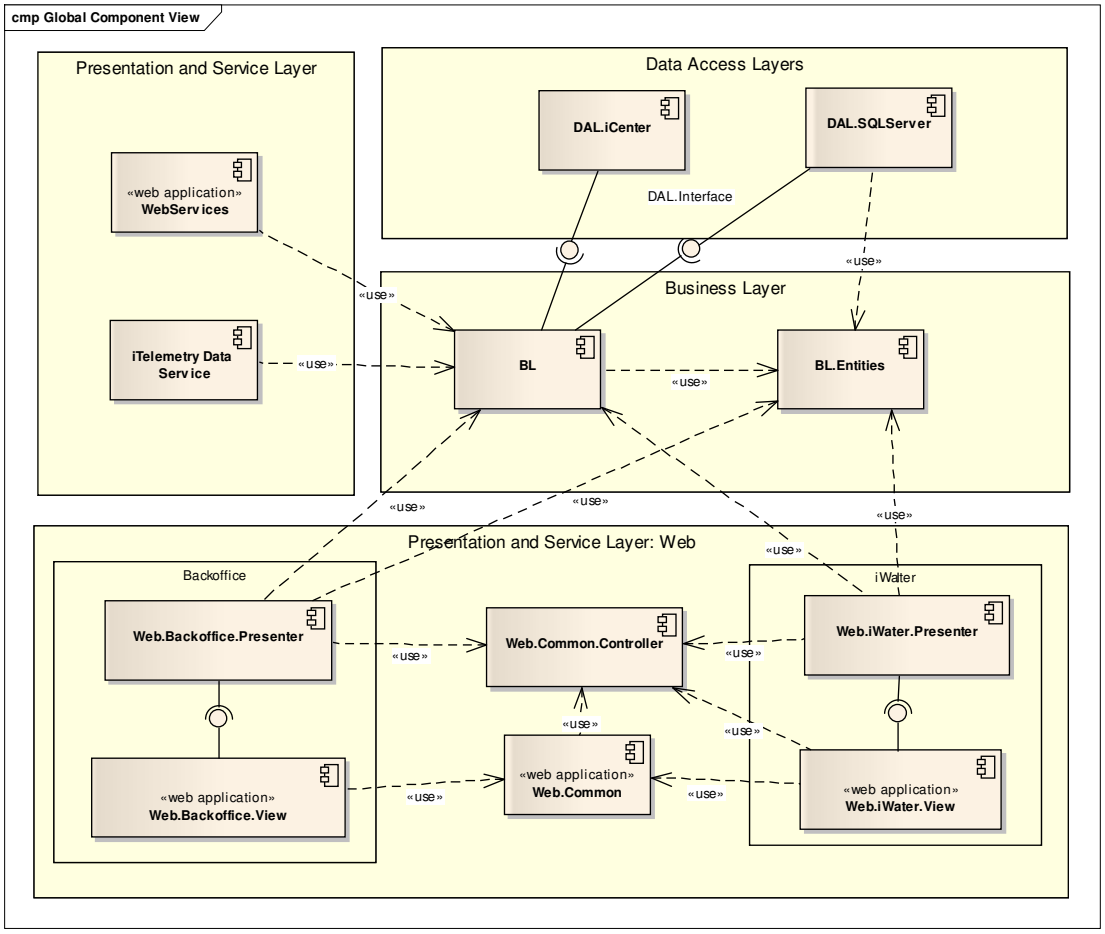


Ilustração 21 - Arquitectura Lógica Antiga do *iTelemetry*

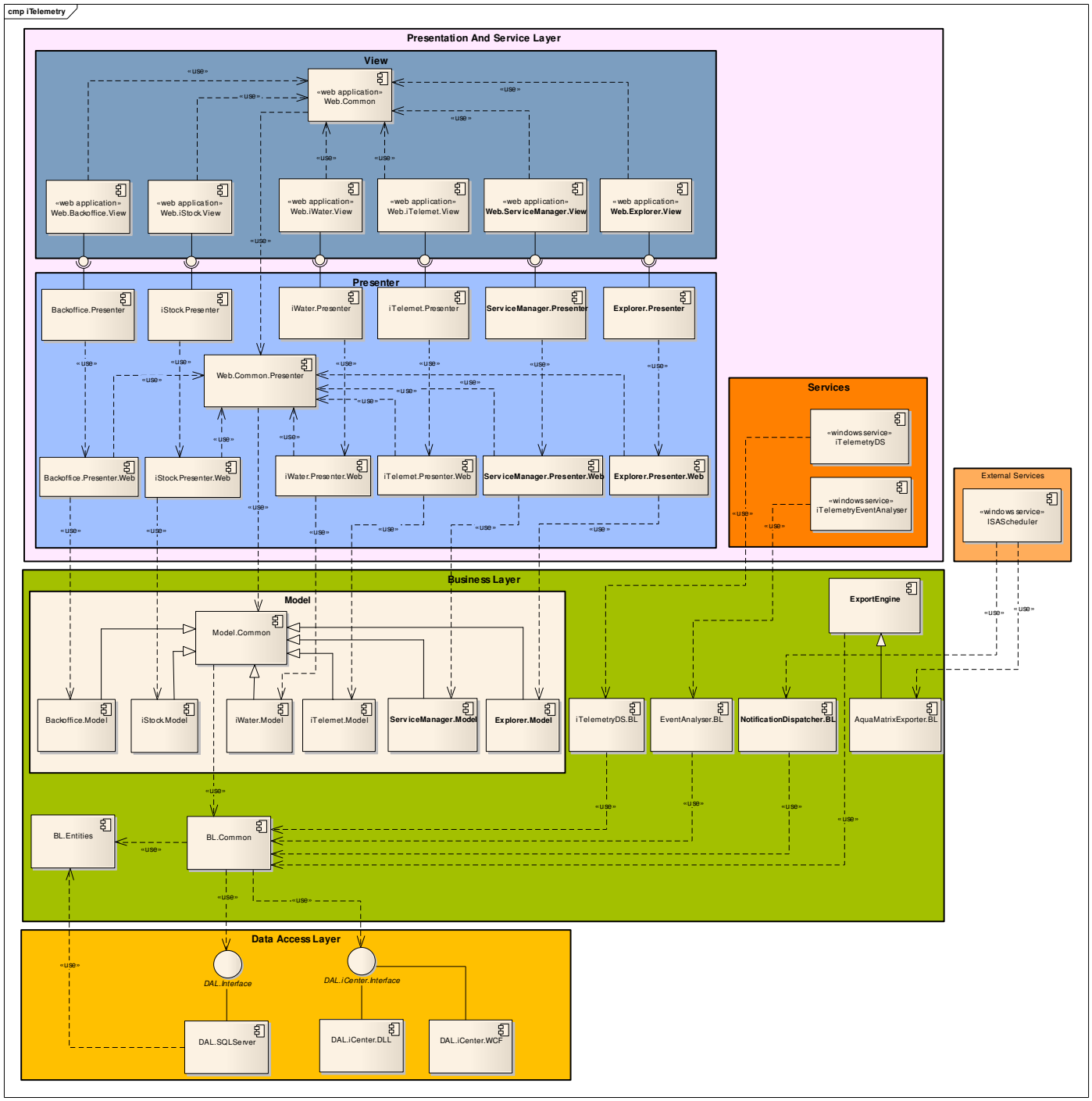


Ilustração 22 - Arquitectura Lógica (Actual) do *iTelemetry*

5.3.1 Presenters

A função de um *Presenter* passa por ir buscar dados à camada lógica e formatá-los, de modo a fornecê-los à *View* (conceito definido no subcapítulo 0). Adicionalmente, um *Presenter* também tem de gerir os dados gerais da aplicação.

Uma das alterações na arquitectura lógica é o facto de passarem a existir, por aplicação, dois *Presenters*, e foi motivada por se pretender separar o que é independente do tipo de aplicação,

daquilo que é dependente. Na anterior versão da arquitectura lógica, apenas existia um *Presenter*, que tratava de tudo.

Na nova arquitectura, o *Presenter.Web* de uma aplicação gere as propriedades globais de aplicações *web*, como são exemplo os dados de sessão que, caso a aplicação não fosse *web*, não existiriam. Já o *Presenter* (não *Web*) de uma aplicação gere os dados que vêm da camada lógica, e que devem ser mostrados na *View*.

Para não existir redundância, apenas o *Presenter.Web* tem acesso à camada lógica. Assim sendo, caso o *Presenter* queira aceder à camada lógica, tem que o fazer via *Presenter.Web*.

5.3.2 Model

Anteriormente, cada aplicação apenas continha uma *View* e um *Presenter*. Era a partir deste último que se acedia à camada lógica, que era partilhada por todas as aplicações e serviços da plataforma.

Também se verificou que, na formulação antiga, a lógica específica para cada aplicação estava disponível para todas as aplicações, mesmo não fazendo qualquer sentido para estas. Como exemplo deste facto na formulação antiga tem-se a obtenção de unidades. Para a aplicação *iWater*, eram obtidas apenas as unidades do negócio da água. Já na aplicação *iStock*, as unidades obtidas eram as do negócio de *racks* de botijas de gás. Assim, existiam dois métodos diferentes na mesma camada lógica, que seleccionavam tipos de unidades diferentes.

Esta lógica não era adequada. Tendo em conta as necessidades, decidiu-se completar a arquitectura com um *Model* dedicado a cada aplicação. É este que trata, agora, da lógica dedicada, o que permite tratar dados especificamente para a aplicação e para o negócio. O processamento da lógica partilhada fica, assim, generalista, ou seja, faz apenas o tratamento de dados que é global a toda a plataforma, a qualquer tipo de negócio.

Na nova arquitectura, pegando novamente no exemplo das unidades para o *iWater* e para o *iStock*, a lógica partilhada tem apenas um método que retorna todos os tipos de unidades. Este método é acedido por cada um dos *Models*, que filtram os dados para os respectivos negócios.

De notar que, mesmo quando se quer aceder a lógica partilhada a partir de um *Presenter*, deve-se utilizar o *Model*.

5.3.3 Serviços

Acompanhando as alterações anteriores feitas na camada lógica, os serviços também tiveram de sofrer alterações, por forma a conterem lógica dedicada a si. Não usando *Models*, foram criados componentes semelhantes a estes, que se encontram na camada lógica, têm um nome correspondente ao serviço a que estão dedicados, e têm um sufixo ".BL" que permite identificar a presença na camada lógica.

Outro facto a notar é a existência, na nova arquitectura, do serviço externo *ISA Scheduler*. Embora não pertença à solução do *iTelemetry*, é uma ferramenta da ISA, responsável por executar processos tais como o *AquaMatrix Exporter* e o *Notification Dispatcher*. O *AquaMatrix Exporter*, por ser um processo descendente do *Data Exportation Service*, tem também uma componente lógica descendente de um *Export Engine*, que contém a lógica comum dos serviços de exportação.

5.3.4 Data Access Layer – iCenter

O *iCenter* passou a ter, na nova arquitectura, dois tipos de acesso a dados: por WCF ou por *assembly*. No entanto, estas diferenças têm de ser invisíveis para a restante plataforma

iTelemetry, ou seja, a forma como são feitos os pedidos à DAL *iCenter* não pode ser influenciada pelo tipo de acesso. A maneira de conseguir isto é através da partilha de interface por parte dos dois tipos de acesso aos dados do *iCenter*. Assim, basta configurar qual o tipo de acesso a dados, por forma a que a camada lógica instancie a DAL correspondente.

5.4 Validação do sistema

O módulo de validação do sistema engloba as acções sobre os dados, desde o momento da sua chegada ao *iTelemetry* até ao momento em que podem ser apresentados aos clientes.

Os dados nem sempre são recolhidos, pelos equipamentos, com os valores correctos, podendo estes ter uma ordem de grandeza diferente da real. Por exemplo, num depósito de combustível cilíndrico deitado, como representado na Ilustração 23 é medido o nível de ocupação (altura), o que não reflecte o verdadeiro volume do combustível nele presente. Apesar de o depósito apresentar 25% do nível, o volume do mesmo não está a um quarto da sua capacidade, o que remete para a calibração dos dados medidos neste depósito. De facto, mesmo que os dados cheguem à plataforma na ordem de grandeza correcta, é necessário que passem pelo processo de calibração para que sejam aprovados.

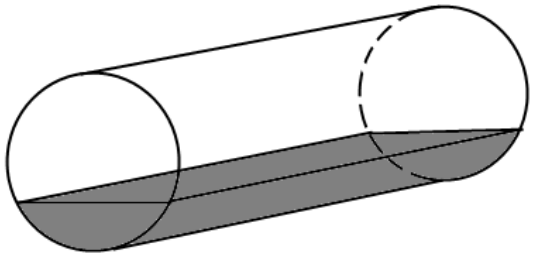


Ilustração 23 - Representação de um depósito de combustível cilíndrico deitado

Por outro lado, os dados que chegam, necessitem ou não de calibração, podem estar inválidos, devido a erros de medição ou erros de transmissão dos equipamentos. Nalguns tipos de negócio, é possível detectar estes dados inválidos e, por vezes, até corrigi-los automaticamente. Caso tenham sido detectados, mas não tenha sido possível corrigi-los automaticamente, os dados devem ser corrigidos por responsáveis pelo acompanhamento do sistema. Um exemplo de um erro que pode acontecer, e que ajuda a perceber a importância da validação do sistema, é o caso em que, num depósito, uma bóia de nível fica presa, dando uma informação errada, podendo o depósito estar vazio quando e bóia indica que o mesmo está cheio.

Assim sendo, o *iTelemetry* precisa de um serviço que suporte os processamentos automáticos de calibração e de validação/correcção automática, e de uma aplicação na qual os responsáveis pelo acompanhamento do sistema possam validar ou corrigir os dados, manualmente. Dentro do serviço, todas as correcções e validações automáticas devem ser da responsabilidade de diferentes algoritmos, dependendo do tipo de negócio a que os dados dizem respeito.

5.4.1 Algoritmos de análise e validação de dados

O processo de análise de dados tem o objectivo de garantir que os dados, que chegam das unidades (equipamentos no terreno), cumpram determinadas regras. É necessário que estes estejam de acordo com o negócio que tratam, com as características do respectivo elemento e com as acções que são exercidas neste (como, por exemplo, abastecimentos).

Todos os dados que respeitem as regras são validados automaticamente. Quando é detectada uma situação que infringe uma regra, os respectivos dados são corrigidos automaticamente se possível. Isto acontece para esconder os efeitos de erros nas transmissões de dados ou na própria medição. Caso a correcção automática dos dados não seja possível, a validação automática pára, sendo necessária a correcção ou validação manual por parte de um responsável pelo acompanhamento do sistema.

De seguida, são apresentados os aspectos mais importantes dos algoritmos aplicados em dois negócios diferentes: contagem de água e nível de gás. Existem outros negócios a tratar, embora não tenham sido enquadrados no estágio. Depois dos algoritmos, é apresentada a forma de correcção dos dados e o modo geral de como está organizada a estrutura dos algoritmos.

Note-se que, neste subcapítulo, os gráficos que exemplificam sequências temporais de dados são meramente ilustrativos, não representando valores reais, mas sim exemplos do que pode acontecer nas sequências de dados. É de referir, também, que todos os gráficos apresentam a relação entre os valores medidos e a data/hora da sua aquisição.

Contagem de água

Na Tabela 7 estão definições de expressões usadas no que se vai seguir. É importante ter em conta que os dados estão organizados crescentemente pela data/hora da sua aquisição.

Actual	Valor do dado actual
Anterior	Valor do dado imediatamente anterior ao 'Actual'
Próximo	Valor do dado imediatamente posterior ao 'Actual'
AntesDoAnterior	Valor do dado imediatamente anterior ao 'Anterior'
Consumo	Diferença entre leituras

Tabela 7 - Definições para explicação dos casos de contagens de água

Para os dados de contagens de água, há, apenas, uma regra a cumprir:

Regra 1: Anterior ≤ Actual

Significa que o valor de um dado tem que ser maior ou igual ao valor do dado anterior, implicando assim uma sequência não decrescente dos dados, em relação à hora da aquisição dos dados. De seguida, são listados os acontecimentos possíveis relativos aos valores dos dados, primeiro em forma de sumário, na Tabela 8, e depois de uma forma mais descritiva.

	Acontecimentos possíveis	Fonte do erro	Acção
1	Anterior ≤ Actual	(Não é um erro)	Validação automática do Actual
2	Actual < Anterior ≤ Próximo	Actual tem valor inválido (inferior ao esperado/real)	Correcção automática do Actual
3	{Actual, Próximo} < AntesDoAnterior ≤ Anterior	Actual e Próximo têm valores inválidos (inferiores ao esperado/real)	Correcção/validação manual a partir do Actual
4	AntesDoAnterior ≤ Actual ≤ Próximo < Anterior	Anterior tem valor inválido (superior ao esperado/real)	Correcção automática do Anterior

Tabela 8 - Acontecimentos para contagens de água

Acontecimento 1: Anterior ≤ Actual

O acontecimento 1 diz respeito ao cumprimento da **Regra 1**, e está representado na Ilustração 24. No gráfico verifica-se, como esperado, que a sequência nunca é decrescente.

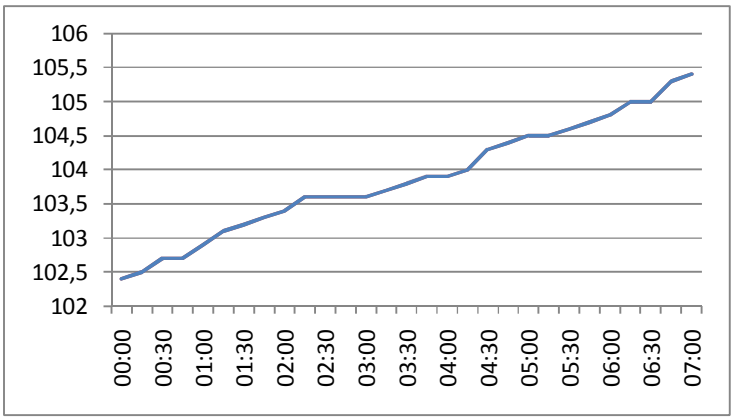


Ilustração 24 - Exemplo do acontecimento 1 para contagem de água

Acontecimento 2: Actual < Anterior ≤ Próximo

A Ilustração 25 representa o acontecimento 2, tendo uma sequência que contém um dado inválido, o qual se observa pela existência de um decréscimo. Trata-se de um caso pontual pois a sua não existência levaria a que a **Regra 1** fosse cumprida. Neste caso, procede-se à correcção automática deste dado (o cálculo do dado de substituição está explicado mais à frente, em "Dados de subsitituição").

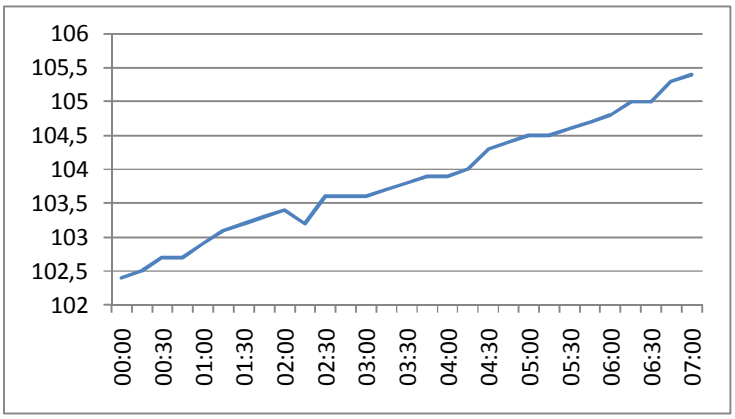


Ilustração 25 - Exemplo do acontecimento 2 para contagem de água

Acontecimento 3: {Actual, Próximo} < AntesDoAnterior ≤ Anterior

Outro caso possível é o da Ilustração 26, que representa o acontecimento 3, e que contém um grande intervalo de dados inválidos (inferiores ao esperado/real). Aqui, a opção adequada é a correcção manual desses dados.

É exemplo deste acontecimento a troca de um determinado contador por outro novo, realizada por um técnico. Neste caso, durante um espaço de tempo os dados não são transmitidos correctamente, e é necessária a validação manual dos dados.

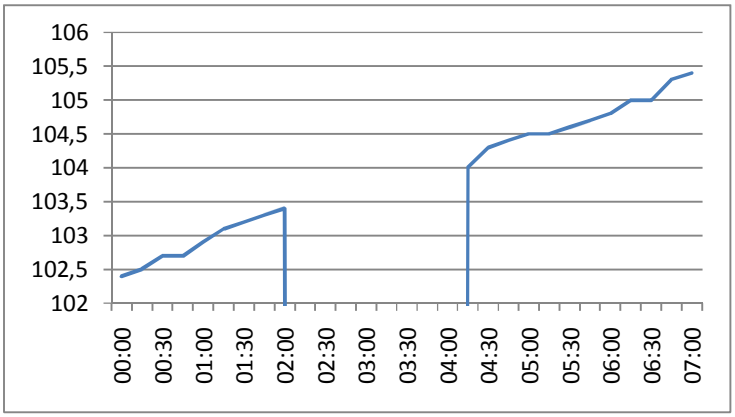


Ilustração 26 - Exemplo do acontecimento 3 para contagem de água

Acontecimento 4: AntesDoAnterior ≤ Actual ≤ Próximo < Anterior

Os casos irregulares são detectados pela existência de descidas. No entanto, nem sempre são os valores inválidos que provocam a descida. Esta pode resultar do dado anterior à descida ser, erradamente, elevado. Este caso representa o acontecimento 4 que é visível na sequência da Ilustração 27. Repare-se que, uma vez que o erro não se encontra no Actual mas sim no Anterior, é este último que tem de ser corrigido automaticamente.

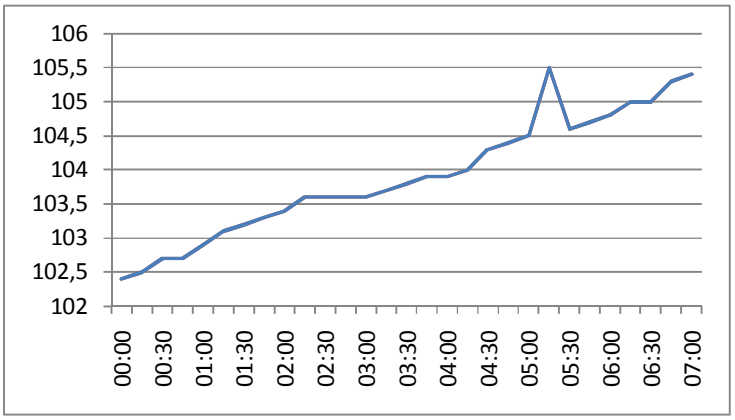


Ilustração 27 - Exemplo do acontecimento 4 para contagem de água

Tendo em conta todos os acontecimentos possíveis, criou-se um algoritmo que consegue obedecer às consequências de cada um, e que está representado na Ilustração 28. Estando os dados organizados pela data/hora da sua aquisição, este algoritmo analisa, apenas, um dado de cada vez. Para cada dado, começa por verificar-se o caso mais comum, em que o dado é superior ou igual ao anterior. Caso isto aconteça, valida-se o dado e passa-se ao seguinte (caso exista; senão, interrompe-se a análise). Caso contrário passa-se à próxima condição, que verifica se o dado é inferior ao anterior e se o dado seguinte "recupera" esta descida. Se esta condição for satisfeita, tenta corrigir-se o dado; senão, verifica-se se o dado anterior é erradamente elevado e se o actual "recuperou" esta subida. No caso desta terceira condição ser satisfeita, tenta corrigir-se o dado anterior ao que estava a ser analisado; caso não se verifique, a análise pára, sem que se valide o dado automaticamente, fazendo com que seja necessária a validação/correcção manual deste dado. Sempre que se consegue corrigir um dado, passa-se

para a análise do próximo, se este existir. Se a correcção não é conseguida, interrompe-se a análise, e os restantes dados passam para validação/correcção manual.

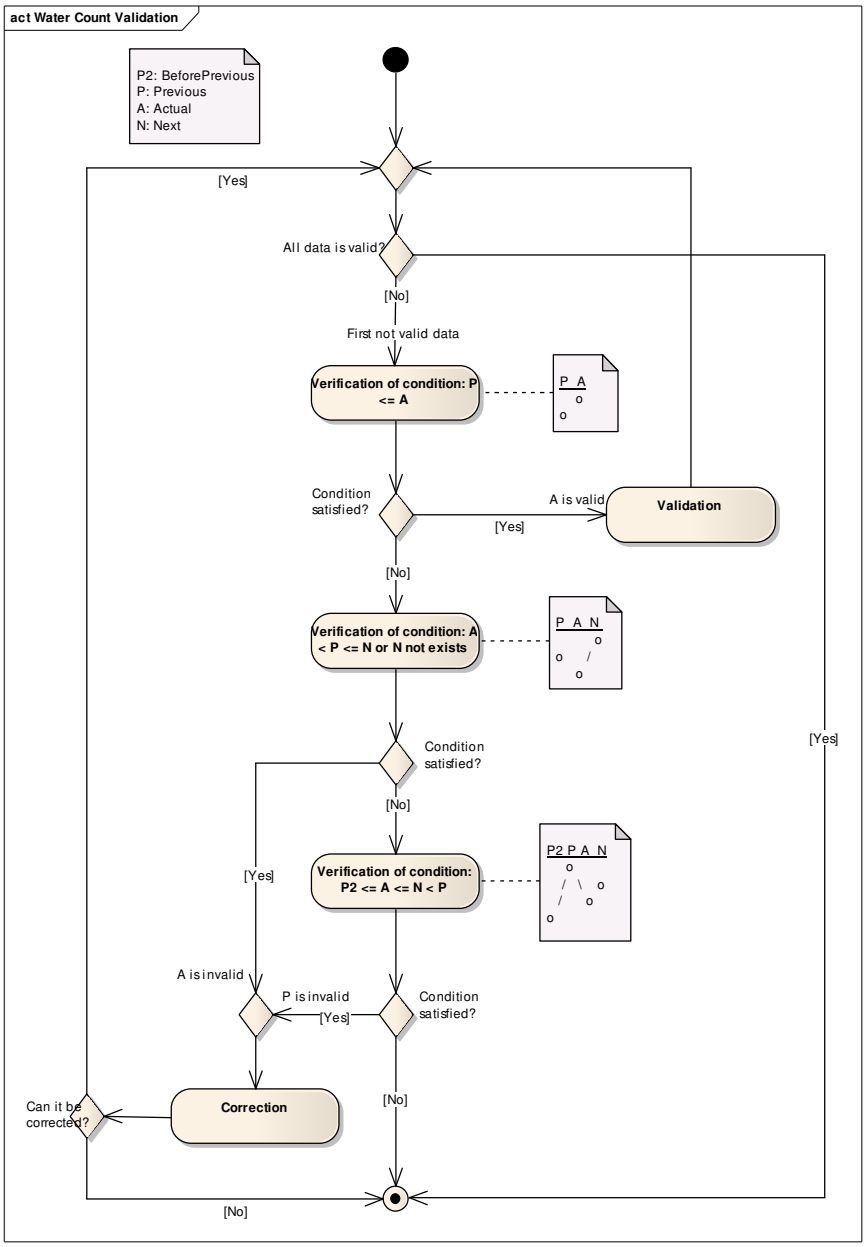


Ilustração 28 - Diagrama de sequência que reflecte a análise de dados do algoritmo para contagens de água

Nível de gás

A Tabela 9 define expressões usadas ao longo da explicação do algoritmo para níveis de gás. Mais uma vez, considere-se que os dados estão organizados crescentemente pela data/hora da sua aquisição.

Actual	Valor do dado actual
Anterior	Valor do dado imediatamente anterior ao 'Actual'

Próximo	Valor do primeiro dado posterior ao 'Actual' e diferente deste (excepto se a diferença das datas/horas deste e do 'Actual' for demasiado elevada; neste caso, considera-se que não existe próximo)
AntesDoAnterior	Valor do dado imediatamente anterior ao 'Anterior'
Consumo	Descida do nível
Abastecimento	Grande subida do nível
ConsumoDiário	Soma dos consumos do dia
ConsumoDiárioEsperado	Média dos consumos dos três dias anteriores e dos mesmos dias da semana das três semanas anteriores (exemplo: o consumo esperado para dia 28 de Junho, terça-feira, é a média entre os consumos dos três dias anteriores, 27, 26 e 25 de Junho, e ainda das três terças-feiras anteriores, 21, 14 e 7 de Junho)
Máximo	Valor máximo aceitável
Mínimo	Valor mínimo aceitável

Tabela 9 - Definições para explicação dos casos de níveis de gás

Os níveis de gás são medidos em depósitos e representados em percentagem, relativamente à capacidade do depósito. Tendo em conta que existem aumentos e diminuições do nível de gás, o que há a avaliar é, sobretudo, a amplitude, em percentagem, desses aumentos e diminuições. Tanto para as regras, como para os acontecimentos, foram fixados valores que distinguem as amplitudes baixas das elevadas. Estes valores tiveram origem na já elevada experiência que a ISA tem nos níveis de gás através da plataforma *MaisGas*.

A sequência dos dados medidos é, normalmente, decrescente, sem que os decréscimos sejam elevados, pois representam os consumos (domésticos ou industriais) do gás que se encontra num depósito. Tem-se, deste modo, a seguinte regra:

Regra 2: Consumo ➔ Actual ≤ Anterior
Anterior − Actual ≤ 4%

No entanto, estes níveis não são sempre decrescentes, pois necessitam de abastecimento, para que o gás de um depósito não se esgote, como sugere a **Regra 3**.

Regra 3: Abastecimento ➔ Actual > Anterior
Actual − Anterior ≥ 4%

Estes abastecimentos são representados por aumentos rápidos do nível de gás num depósito. A Ilustração 29 demonstra a variação típica do nível de gás de um depósito, em percentagem, ao longo do tempo.

Apesar de, neste contexto, serem aceitáveis tanto subidas como descidas, nem todos os valores são válidos. Na Tabela 10, são apresentados os acontecimentos possíveis relativos à variação dos valores dos dados para níveis de gás, à qual se segue uma descrição dos mesmos.

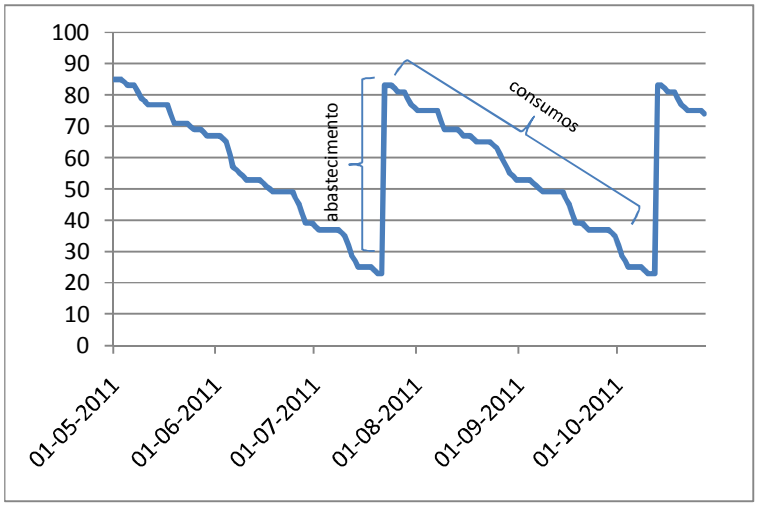


Ilustração 29 - Variação normal do nível de gás em depósitos ao longo do tempo

	Acontecimentos possíveis	Fonte do erro/significado	Acção
1	Actual > Máximo ou Actual < Mínimo	Actual tem valor inválido	Correcção/validação manual do Actual
2	Actual > Anterior e Actual − Anterior > 4	Abastecimento	Validação automática do Actual
3	Próximo > Actual > Anterior e Próximo − Anterior ≥ 5 e Actual − Anterior ≤ 4	Abastecimento do Anterior para o Próximo	Validação automática do Actual
4	Actual > Anterior > AntesDoAnterior e Actual − AntesDoAnterior ≥ 5 e Actual − Anterior ≤ 4	Abastecimento do AntesDoAnterior para o Actual	Validação automática do Actual
5	Anterior > AntesDoAnterior ≥ Actual e Anterior − AntesDoAnterior ≥ 4 e AntesDoAnterior − Actual ≤ 4	Anterior tem valor inválido	Correcção automática do Anterior
6	Próximo > Anterior > Actual e Próximo − Anterior ≥ 5 e Anterior − Actual ≥ 4	Abastecimento do Actual para o Próximo	Validação automática do Actual
7	Actual > Próximo > Anterior e Próximo − Anterior ≥ 5 e Actual − Próximo ≥ 4	Abastecimento do Anterior para o Actual	Validação automática do Actual
8	Anterior ≥ Actual e Anterior − Actual ≤ 4	Consumo normal	Validação automática do Actual
9	Anterior > Actual > Próximo e Anterior − Actual ≥ 4	Consumo elevado, mas não inválido	Validação automática do Actual
10	Anterior ≥ Próximo > Actual e Anterior − Próximo ≤ 4 e Anterior − Actual > 4	Actual tem valor inválido	Correcção automática do Actual
11	Actual > Anterior ≥ Próximo e Actual − Anterior ≤ 4 e Anterior − Próximo ≤ 4	Actual tem valor inválido	Correcção automática do Actual
12	Próximo > Actual > Anterior e Próximo − Anterior ≤ 2	Pequena perturbação	Validação automática do Actual
13	Próximo > Actual > Anterior e Próximo − Actual ≤ 4 e Actual − Anterior ≤ 4 e Próximo − Actual > 2	Actual e Próximo têm valores inválidos	Correcção/validação manual a partir do Actual

14	Anterior > Próximo > Actual e Anterior − Próximo ≥ 4 e Próximo − Actual ≤ 4	Actual e Próximo têm valores inválido	Correcção/validação manual a partir do Actual
15	ConsumoDiário > ConsumoDiárioEsperado * 2 ou ConsumoDiário < ConsumoDiárioEsperado / 4	Consumo diário fora do esperado	Correcção/validação manual a partir do primeiro dado do dia

Tabela 10 - Acontecimentos para níveis de gás

Acontecimento 1: Actual > Máximo ou
Actual < Mínimo

Os depósitos de gás devem ter uma percentagem mínima de gás, para não existirem falhas no fornecimento aos consumidores. Além disso, não podem exceder uma percentagem máxima, devido às próprias propriedades do gás. Assim sendo, se existir algum valor fora destes limites, os quais são configuráveis, deve proceder-se à sua análise, efectuada por um responsável pelo acompanhamento do sistema. Tendo em conta a gravidade desta situação, os responsáveis pelo depósito devem ser informados de imediato. No entanto, como no *iTelemetry* já existe uma funcionalidade que gere as situações de alarme, não é da responsabilidade do algoritmo proceder a estas notificações.

Acontecimento 2: Actual > Anterior e
Actual − Anterior > 4

O acontecimento 2 refere-se a um abastecimento, tal como indica a **Regra 3**, e cuja Ilustração 30 exemplifica.

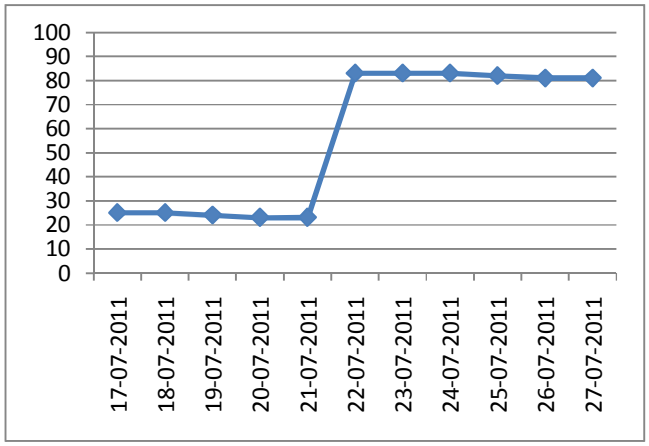


Ilustração 30 - Exemplo do acontecimento 2 para níveis de gás

Note-se que, neste exemplo, o abastecimento ocorre entre os dias 21-07-2011 e 22-07-2011.

Tendo em conta o funcionamento habitual destes depósitos, a única forma de o nível de gás subir num depósito é nos abastecimentos, representados por grandes subidas. Assim sendo, uma subida pequena do nível de gás deve ser, à partida, considerada como erro.

Acontecimento 3: Próximo > Actual > Anterior e
Próximo − Anterior ≥ 5 e
Actual − Anterior ≤ 4

Acontecimento 4: Actual > Anterior > AntesDoAnterior e

Actual − AntesDoAnterior ≥ 5 e
Actual − Anterior ≤ 4

Nem todos os abastecimentos ocorrem como no acontecimento 2, podendo uma medição ser feita quando se está a iniciar ou a acabar um abastecimento, já que estes não são instantâneos. Resultantes destas situações, tem-se o acontecimento 3, que representa uma medição no início de um abastecimento, e o acontecimento 4, que representa uma medição no final de um abastecimento. Nestes casos, para além de uma grande subida provocada pelo abastecimento, existe, ainda, uma outra subida mais pequena, considerada aceitável por ter resultado de uma medição efectuada durante o abastecimento. As Ilustração 31 e Ilustração 32 apresentam exemplos dos acontecimentos 3 e 4, respectivamente.

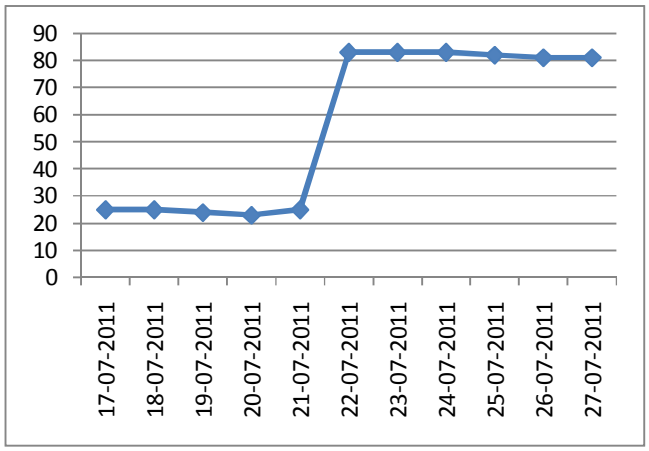


Ilustração 31 - Exemplo do acontecimento 3 para níveis de gás

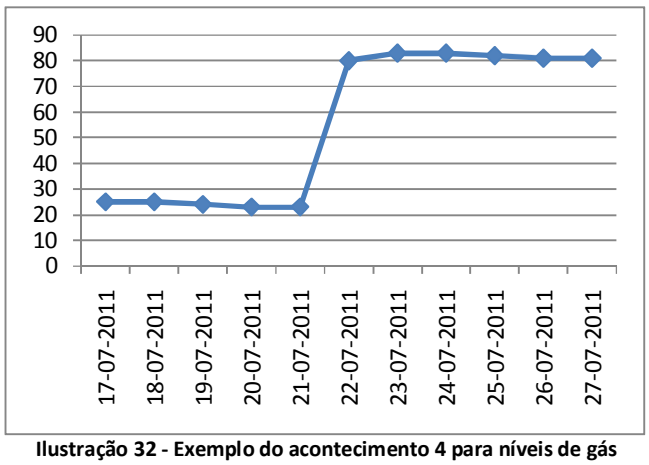


Ilustração 32 - Exemplo do acontecimento 4 para níveis de gás

Acontecimento 5: Anterior > AntesDoAnterior ≥ Actual e
Anterior − AntesDoAnterior ≥ 4 e
AntesDoAnterior − Actual ≤ 4

O acontecimento 5 representa um falso abastecimento, ou seja, um caso em que, depois de uma subida que parecia um abastecimento, há uma descida para um valor aproximado ao que

antecedeu a subida. Este caso está exemplificado na Ilustração 33, o qual resulta na correcção do valor que provocou a subida (esta correcção será explicada mais à frente).

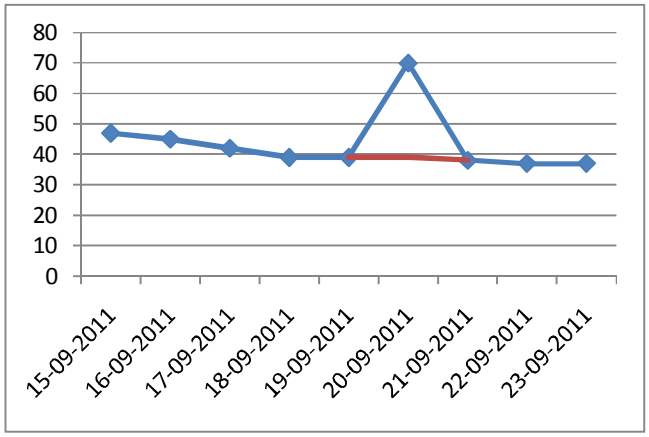


Ilustração 33 - Exemplo do acontecimento 5 para níveis de gás

Acontecimento 6: Próximo > Anterior > Actual e
Próximo − Anterior ≥ 5 e
Anterior − Actual ≥ 4

Acontecimento 7: Actual > Próximo > Anterior e
Próximo − Anterior ≥ 5 e
Actual − Próximo ≥ 4

Os acontecimentos 6 e 7 representam grandes descidas que ocorrem imediatamente antes e depois de um abastecimento, respectivamente, sendo que, assim, não é possível perceber-se se são ou não válidas. A opção tomada para estas situações foi a de validar automaticamente estas sequências, já que, caso a descida tenha sido mesmo elevada, pelo acontecimento 15 (verificação do consumo diário), a mesma sequência irá passar para validação manual. Estes acontecimentos são exemplificados nas Ilustração 34 e Ilustração 35.

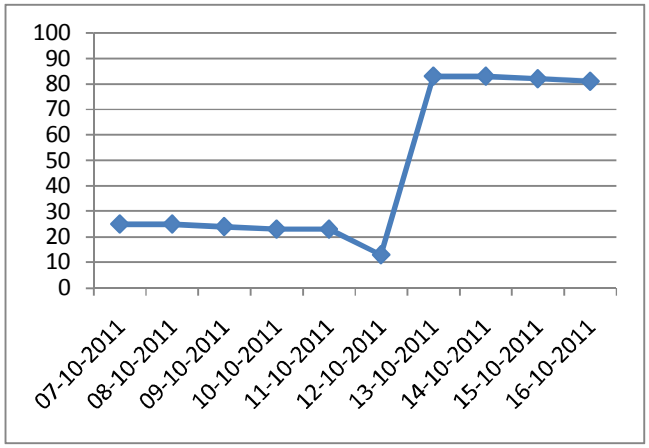


Ilustração 34 - Exemplo do acontecimento 6 para níveis de gás

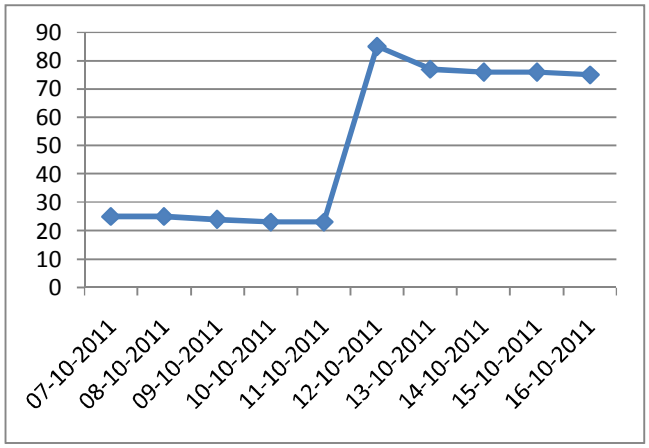


Ilustração 35 - Exemplo do acontecimento 7 para níveis de gás

Acontecimento 8: Anterior ≥ Actual e
Anterior − Actual ≤ 4

O acontecimento 8 retrata um consumo normal e está de acordo com a **Regra 2**. Portanto, o dado em questão será validado. Este acontecimento é o mais comum na sequência de dados de níveis de gás, e está exemplificado na Ilustração 36.

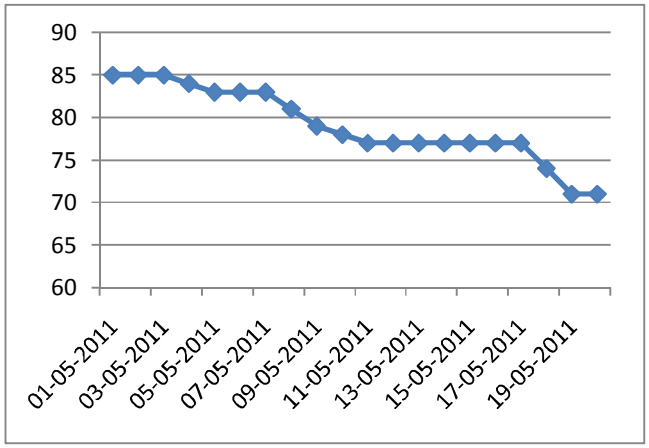


Ilustração 36 - Exemplo do acontecimento 8 para níveis de gás

Acontecimento 9: Anterior > Actual > Próximo e
Anterior − Actual ≥ 4

Representando o acontecimento 9, tem-se o caso onde ocorre um consumo elevado, seguido de um consumo (que pode ser também elevado ou não), como mostra o exemplo da Ilustração 37.

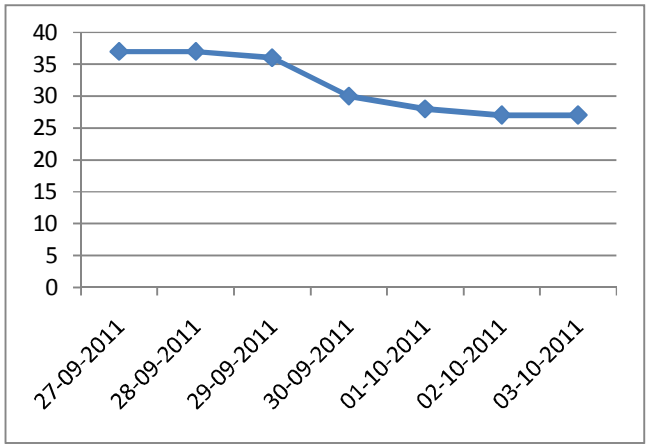


Ilustração 37 - Exemplo do acontecimento 9 para níveis de gás

Tendo em conta que o consumo elevado foi continuado com novo consumo, o dado é considerado válido. Caso este consumo tenho sido realmente excessivo, o consumo diário deverá ser considerado inválido, como retrata o acontecimento 15.

Acontecimento 10: Anterior ≥ Próximo > Actual e
Anterior − Próximo ≤ 4 e
Anterior − Actual > 4

O acontecimento 10 representa os casos em que acontece uma descida elevada, seguida de uma recuperação, ou seja, se o dado que provocou a descida não existisse, o consumo estaria de acordo com a **Regra 2**. Este acontecimento é apresentado na Ilustração 38, e será efectuada a correcção automática do dado que provocou a descida, considerado inválido.

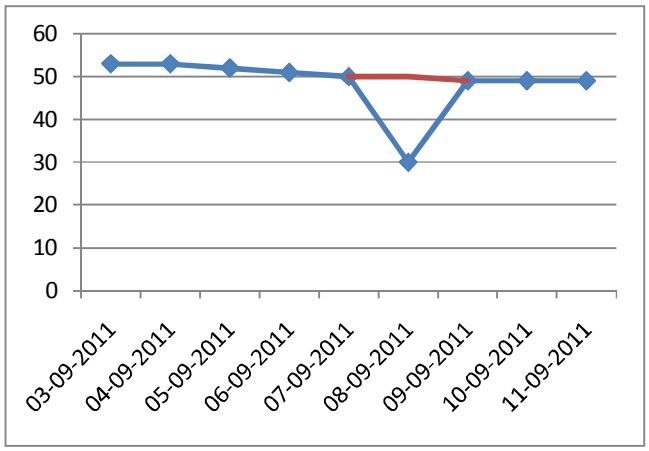


Ilustração 38 - Exemplo do acontecimento 10 para níveis de gás

Acontecimento 11: Actual > Anterior ≥ Próximo e
Actual − Anterior ≤ 4 e
Anterior − Próximo ≤ 4

O acontecimento 11 retrata uma pequena subida seguida de uma recuperação para o valor antecedente à subida. Neste caso, o dado que provocou a subida deve ser corrigido

automaticamente, para que a sequência fique de acordo com a **Regra 2**. A Ilustração 39 é um exemplo deste acontecimento.

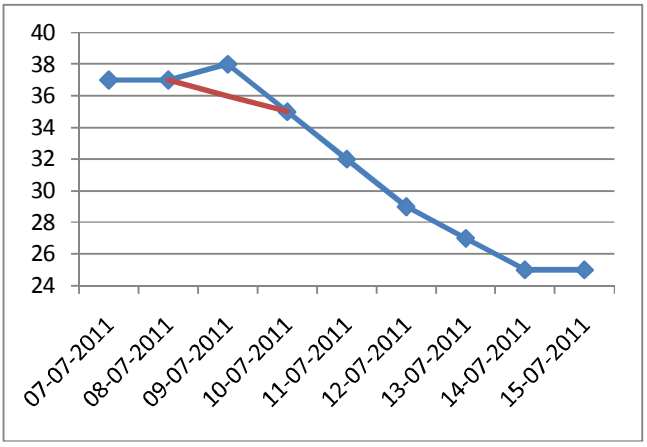


Ilustração 39 - Exemplo do acontecimento 11 para níveis de gás

Acontecimento 12: Próximo > Actual > Anterior e
Próximo − Anterior ≤ 2

O acontecimento 12 trata-se de uma excepção, em que uma pequena subida é considerada válida, pois representa uma pequena variação, que pode dever-se a pequenas perturbações nos sensores. Um exemplo deste acontecimento está representado na Ilustração 40.

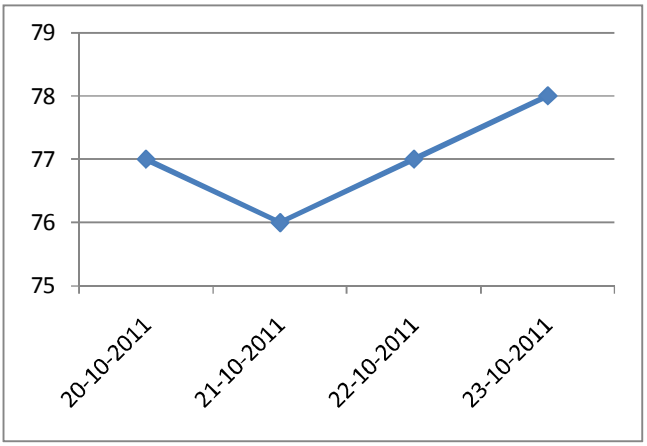


Ilustração 40 - Exemplo do acontecimento 12 para níveis de gás

Acontecimento 13: Próximo > Actual > Anterior e
Próximo − Actual ≤ 4 e
Actual − Anterior ≤ 4 e
Próximo − Actual > 2

Quando ocorre uma subida pequena, mas suficientemente elevada para não ser considerada uma simples perturbação, como representa o acontecimento 13, os dados devem ser avaliados por um responsável pelo acompanhamento do sistema. A Ilustração 41 exemplifica este acontecimento.

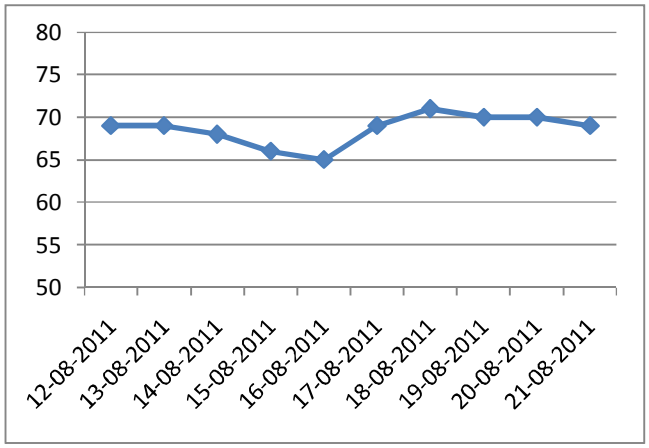


Ilustração 41 - Exemplo do acontecimento 13 para níveis de gás

Acontecimento 14: Anterior > Próximo > Actual e
Anterior − Próximo ≥ 4 e
Próximo − Actual ≤ 4

O acontecimento 14 é mais um caso em que não é possível ajuizar o que está inválido pois, depois de uma grande descida, ocorre uma pequena subida. Mais uma vez, o responsável pelo acompanhamento do sistema tem que validar ou corrigir a sequência de dados. Este acontecimento está representado, com um exemplo, na Ilustração 42.

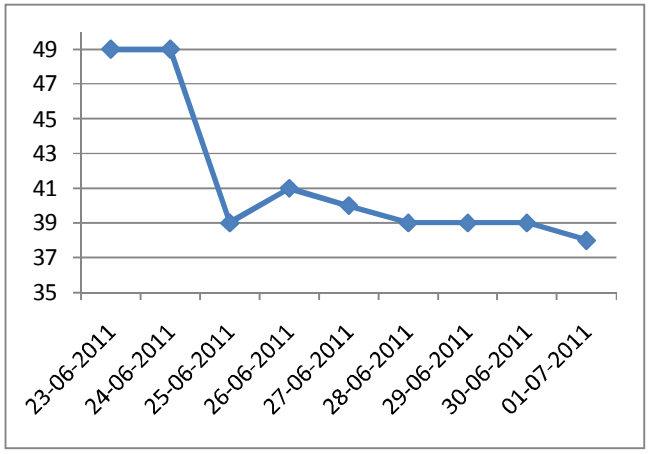


Ilustração 42 - Exemplo do acontecimento 14 para níveis de gás

Acontecimento 15: ConsumoDiário > ConsumoDiárioEsperado × 2 ou
ConsumoDiário < ConsumoDiárioEsperado ÷ 4

O acontecimento 15 deve ser analisado ao final do dia, uma vez que diz respeito ao consumo verificado durante todo o dia. Quando este consumo diário é muito superior ou muito inferior ao esperado, tal dia deve ser considerado inválido (ou seja, todos os valores recolhidos nesse dia devem ser considerados inválidos), dando origem a uma análise cuidada por parte de um responsável pelo acompanhamento do sistema. Pensou-se na hipótese de, em vez de analisar cada dia, analisar o consumo entre cada dois dados consecutivos. No entanto, chegou-se à conclusão que esta seria uma má opção, uma vez que grande parte do consumo diário pode, na

realidade, ser feito num curto espaço de tempo, ou seja, o consumo instantâneo pode variar bastante, ao contrário do consumo diário que, por norma, não sofre grandes variações.

Apesar de terem sido analisados os acontecimentos das sequências de dados para níveis de gás, não foi construído, ainda, um algoritmo que os trate. Esta é mais uma tarefa proposta para o futuro.

Dados de substituição

Tanto para a contagem de água, como para o nível de gás, cada dado só pode ser corrigido de uma maneira. Tendo em conta que, neste caso, nunca são corrigidos dados consecutivos ao mesmo tempo, considera-se sempre que os dados anterior e posterior ao dado inválido (seja, neste caso, o Actual) estão correctos. Assim sendo, pode assumir-se que é seguido o consumo médio. Exemplifique-se com um caso de contagem de água: suponha-se que às 13h o Anterior tem o valor de 140L, às 14h houve um dado inválido, que é o Actual, e às 16h o Próximo está em 146L. Neste caso, às 14h pode assumir-se que o valor correcto é 142L, sendo assim coerente com os 2L/h observados entre as 13h e as 16h.

Para esta substituição, pensou-se, também, em efectuar comparações entre os dados mais antigos, para perceber quais são os consumos normais para o mesmo momento do dia. No entanto, esta foi uma opção posta de parte, porque verificou-se que os consumos não seguem padrões.

Estrutura de um algoritmo

Os algoritmos são diferentes uns dos outros na análise que fazem aos dados, mas devem ser reconhecidos como iguais na sua estrutura, já que, quem os executa, deve fazê-lo da mesma forma, independentemente do algoritmo. Deste modo, foi definido que cada algoritmo concreto deve ser conhecido através de uma interface comum a todos. No entanto, há propriedades que todos os algoritmos devem implementar de igual forma, sendo que, para isso, foi criada uma classe abstracta comum.

Para não repetir código na análise aos dados pelos vários algoritmos, decidiu-se pela criação de novas classes que representam condições e acções. Das primeiras, pretende-se saber se estas são ou não satisfeitas para um certo dado, enquanto das acções pretende-se tomar uma acção sobre um dado. Cada condição ou acção pode ser usada em diferentes algoritmos, sendo que os vários algoritmos estruturam estas condições e acções de diferentes maneiras. Por exemplo, o algoritmo de contagens de água, para os dados que satisfaçam a condição de um dado ser maior do que o anterior, pode tomar a acção de validar, enquanto o algoritmo de níveis de gás, para os dados que satisfaçam esta mesma condição, pode tomar a acção de corrigir.

As condições têm, também, propriedades comuns entre elas, que são definidas numa classe abstrata da qual todas as condições descendem. Para além disso, os algoritmos não conhecem a estrutura de cada condição, mas sim a estrutura comum, definida por uma interface. Quanto às acções, estas seguem a mesma estrutura que as condições, descendendo, também, de uma classe abstrata que implementa uma interface.

Toda esta estrutura está definida no diagrama de classes da Ilustração 43.

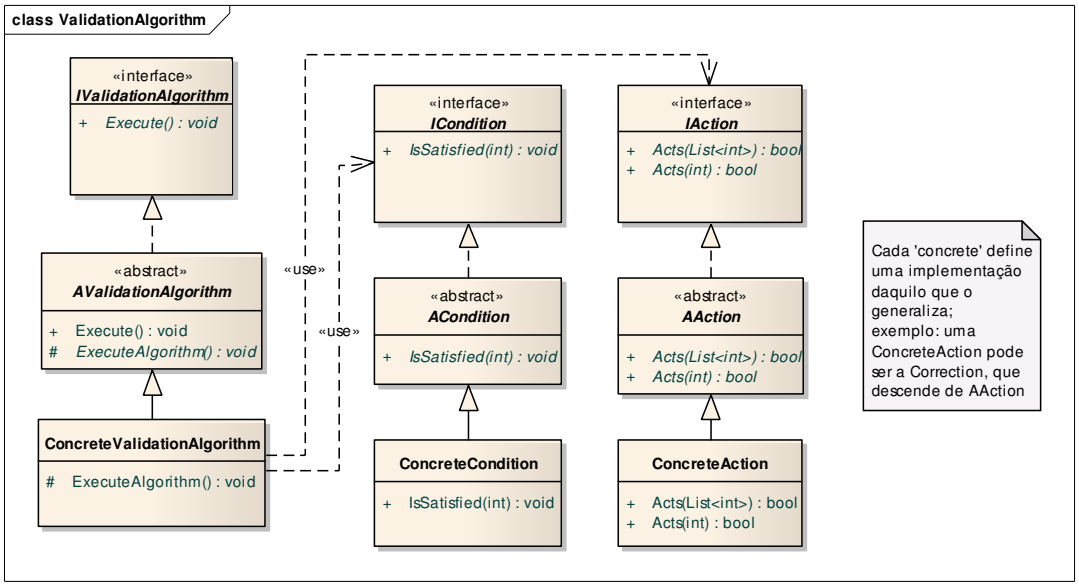


Ilustração 43 - Diagrama de classes referente da estrutura dos algoritmos de validação

5.4.2 Data Pre-Processor

O *Data Pre-Processor*, serviço de pré-processamento de dados, tem a finalidade de validar os dados que chegam ao *iTelemetry*, para que, posteriormente, sejam correctamente disponibilizados aos clientes. Este serviço, para além de tratar da calibração e da validação/correcção automática, também sinaliza os dados que necessitam de validação ou correcção manual, por parte dos responsáveis pelo acompanhamento do sistema.

Este serviço interage com uma lista, denominada *Data Pre-Processor List*, onde se encontram os pedidos para calibração, e para validação/correcção tanto automática como manual.

Quando chegam dados à plataforma, são colocados na base de dados assinalados como não calibrados. É, de imediato, inserido um pedido na *Data Pre-Processor List*, que dá origem, por parte do serviço de pré-processamento de dados, à calibração dos respectivos dados. Depois de calibrados, procede-se à sua actualização na base de dados, e é inserido um pedido na *Data Pre-Processor List* para validação/correcção automática dos mesmos dados.

O mesmo serviço, quando na lista há pedidos para validação/correcção de dados, faz com que os dados passem pelo algoritmo correcto, que depende do tipo de produto (água, gás, temperatura, etc.) e do tipo de medição (nível num depósito, contagem que passa numa conduta, etc.) a que os dados pertencem. Cada algoritmo tem que ser definido de acordo com as suas regras, sendo que, depois de verificadas condições nos dados, são-lhes aplicadas as respectivas acções. Enquanto um exemplo de condição é "o dado é superior ao imediatamente anterior", um exemplo de acção é a validação do dado.

A Ilustração 44 representa a sequência do serviço, incluindo a calibração e a validação, como está descrito acima.

É esperado que este serviço tenha muitos processamentos, e que estes sejam longos pois, normalmente, chegam ao *iTelemetry* muitos dados ao mesmo tempo. Para que os tempos de espera não sejam tão longos, o serviço foi criado com a possibilidade de seleccionar, mediante o(s) cliente(s), os dados que calibra e valida, podendo ter, assim, várias instâncias do serviço a analisar dados diferentes.

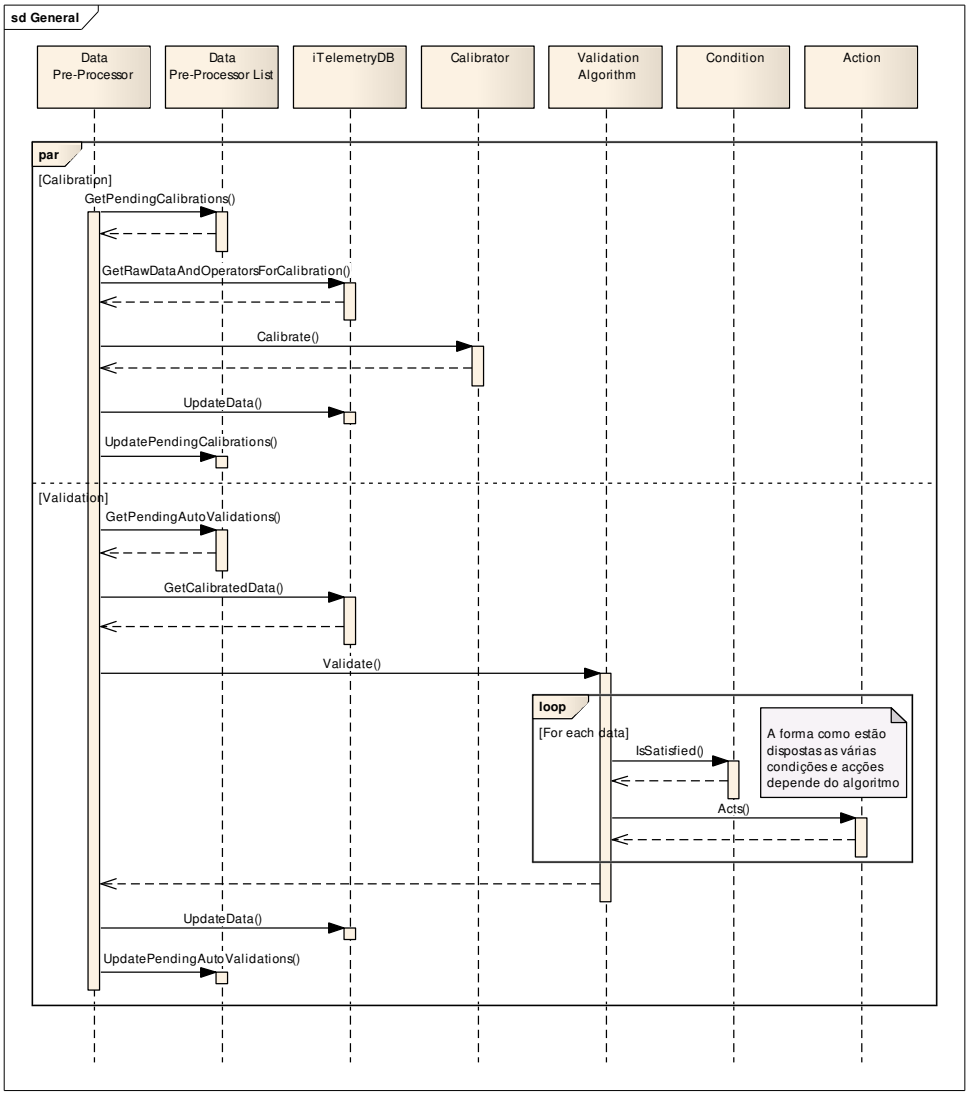


Ilustração 44 - Diagrama de sequência referente ao funcionamento do *Data Pre-Processor Service*

Por exemplo, se dois clientes querem os dados todos validados às 9h de cada dia, é normal que as suas unidades estejam configuradas para enviar todos os seus dados guardados pouco antes das 9h. Isto significa que o *Data Pre-Processor* teria muitos dados, referentes aos dois clientes, para validar num curto espaço de tempo. Neste sentido, é importante existirem duas instâncias que tratem, cada uma, dos dados de cada cliente, para que um não tenha de esperar pelo outro.

5.4.3 Aplicação para validação de dados manual

Para a validação/correcção manual por parte dos responsáveis pelo acompanhamento do sistema, é necessária a existência de uma aplicação na qual estes responsáveis possam interagir com os dados. Para se perceber quais os clientes que necessitam de validação/correcção manual de dados, é importante organizar um sumário com essa informação, mostrando, também, quais os tipos de dados que necessitam dessa validação/correcção. Outra funcionalidade que a aplicação deve ter é a de permitir, aos responsáveis pelo acompanhamento do sistema, a visualização dos dados como se fosse cliente, por forma a perceber melhor a diferença entre os dados considerados válidos e os considerados inválidos.

Quanto à validação/correcção, é possível executar várias acções sobre os dados:

- inserir um dado novo, no caso de existir uma medição manual que se queira inserir na sequência;
- apagar dados, que, por exemplo, tenham sido erradamente medidos;
- validar dados;
- ligar dois dados, corrigindo os dados que se encontram entre eles (estes dados corrigidos passam a tomar valores correspondentes ao diferença médio entre os dados ligados).

Tendo em conta que os dados são disponibilizados ao utilizador graficamente, decidiu-se permitir estas acções na interacção com tal gráfico. Deste modo, sempre que o utilizador clica sobre um dado ou cria uma linha imaginária entre dois pontos (consecutivos ou não) com o ponteiro do rato, são lhe mostradas as respectivas possibilidades de acções. Repare-se que a criação da linha imaginária permite criar conjuntos de pontos que podem, por exemplo, ser validados ou apagados de uma só vez.

Quando o utilizador valida ou corrige um dado que não foi possível validar ou corrigir automaticamente (o algoritmo de validação parou neste dado), é chamado o algoritmo para se tentar validar/corrigir automaticamente os dados seguintes a este.

Esta aplicação ainda não se encontra desenvolvida, mas já está prototipada, como se pode ver no Anexo IV - Aplicação para validação de dados manual - Prototipo. Neste, pode ser observada a aplicação das funcionalidades descritas acima.

5.4.4 Participação do estagiário

Em primeiro lugar, é de notar que o módulo de validação de sistema ainda não está completo, uma vez que: o *Data Pre-Processor* não foi, ainda, posto em produção; a aplicação para validação de dados manual não foi, ainda, implementada; o algoritmo de validação/correcção para níveis de gás não foi, ainda, implementado; e existem outros algoritmos que não foram começados.

Quanto às partes já completas, que foram descritas neste subcapítulo, apenas uma não foi executada maioritariamente pelo estagiário: a construção do *Data Pre-Processor*. No entanto, a modelação do serviço foi feita pelo mesmo, tendo sofrido alterações durante a discussão com os restantes elementos da equipa. Todas as outras partes do módulo de validação do sistema já construídas, apesar de discutidas em equipa, foram executadas pelo estagiário.

5.5 iTelemetry - Outras tarefas

Ao longo do estágio, foi necessária a participação do estagiário na execução de outras tarefas do *iTelemetry*. Inicialmente, houve necessidade deste se adaptar à empresa, o que incluiu o conhecimento do projecto. Paralelamente ao *iTelemetry*, foram efectuados estudos de comparação de *profilers* e de análise de *templates* AJAX, que serviram para estudar possíveis melhoramentos, não só no *iTelemetry*, como noutros projectos da ISA. Além disso, houve trabalhos nas principais aplicações, como é o caso da redefinição da relação entre Unidades e Locais, e da definição das unidades de engenharia relativas aos dados adquiridos no terreno. Para as entregas de produto, efectuaram-se preenchimentos de fichas de revisão e verificação, assim como testes à plataforma.

5.5.1 Adaptação à empresa

As primeiras semanas de estágio foram dedicadas à adaptação e integração na empresa, equipa, projecto e tecnologias.

A falta de experiência profissional por parte do estagiário motivou o rigoroso conhecimento dos procedimentos, tanto da empresa, como do modo de trabalhar no projecto. Também o baixo nível de conhecimento das tecnologias levou a que fosse necessário resolver alguns exercícios, por forma a conhecer melhor as características e capacidades dos meios utilizados durante o estágio.

Procedimentos da ISA

Para a introdução do estagiário na ISA, foram lidos documentos respeitantes à estrutura e procedimentos da empresa. Estes documentos listam os direitos e deveres de um colaborador no que diz respeito ao uso dos bens que a empresa dispõe. Para complementar esta leitura, houve uma reunião com a Gestora de Qualidade da ISA, onde as regras e métodos foram explicados de forma mais clara.

Depois da estrutura da ISA, foi necessário perceber o funcionamento do departamento de *software*, local onde o estágio foi realizado. Para isso, também foram lidos os respectivos documentos.

Documentação do iTelemetry

Para a integração no projecto, foi feita uma leitura de documentos relativos à especificação geral e à especificação dos módulos mais importantes. Nestes documentos, contextualiza-se o projecto e os seus módulos, e ainda são apresentadas as funções de cada módulo, e os requisitos e a arquitectura física do *iTelemetry*.

Também o documento relativo ao desenho detalhado foi consultado. Este documento contém as opções técnicas tomadas no desenho da arquitectura, bem como a arquitectura lógica, diagramas de sequência e diagramas de estados.

À medida que foram necessários, foram consultados outros documentos como fichas de revisão, fichas de verificação e planos de teste.

Exercícios de adaptação

Depois de percebido o projecto, foram realizados exercícios de iniciação à linguagem. Apesar de não terem sido utilizados no projecto, estes exercícios integraram a base de dados do *iTelemetry*, e os seus objectivos foram baseados nalguns objectivos do projecto. Especificamente, pretendia-se a criação de uma aplicação *web* com apresentação de dados numa tabela, e que incluísse filtros e a possibilidade de adição e edição de dados. Estes eram simples combinações chave-valor, devido à sua utilização em vários pontos do *iTelemetry*.

5.5.2 Comparação de profilers

Cada vez mais se sente a necessidade de monitorização dos projectos. O mesmo se passou, tanto com o *iTelemetry*, como com todos os projectos do departamento de *software* da ISA. Como tal, foi definida a necessidade do uso de um *Profiler*, cujo objectivo passa por analisar utilização de CPU, de memória e de tempos de execução. Decidiu-se, então, elaborar uma

análise comparativa de *Profilers*, que foi feita à base de pesquisas para saber quais os mais populares, e instalando as versões de experiência (*trials*) de cada *Profiler*.

Metodologia

Analisaram-se os *Profilers* VSTS (*Visual Studio Team System*), ANTS (*Red Gate*), *AQTime*, *dotTrace*, EQATEC, *Intel VTune* e *GlowCode*, e os principais pontos tidos em conta nesta comparação foram o tempo de execução, as percentagens de utilização de CPU e memória, os preços e a facilidade de trabalhar com o *software*.

Primeiro, houve uma fase de aprendizagem de cada *Profiler*. Depois disso, monitorizou-se uma mesma aplicação com cada um dos *Profilers* e, retirando os dados necessários, foi feita uma análise comparativa entre as diferentes opções. Esta análise e as respectivas conclusões ajudam na escolha entre as ferramentas estudadas.

Acompanhando este processo, foi feito um relatório, que foi entregue aos superiores, e o qual se encontra em anexo, com o nome Anexo V - Estudo de Profilers.

Conclusões

Depois da análise comparativa, concluiu-se que o ANTS (*Red Gate*) é o mais completo, tendo também uma boa apresentação. Devido a questões financeiras, são, ainda, de destacar o VSTS, por ser *freeware*, e o EQATEC, por ser o mais barato.

5.5.3 Templates AJAX

A ideia deste estudo passa por comparar as normais abordagens de preenchimento de tabelas em ASP.NET com uma nova abordagem, que inclui *Templates* AJAX. É importante perceber se a utilização deste novo método, tanto no *iTelemetry*, como noutros projectos da ISA, é compensatória no que diz respeito à performance e à fiabilidade.

Normalmente, em ASP.NET, utilizam-se *GridViews* com *Update Panels* nas actualizações das tabelas, e *ObjectDataSources* quando se tratam grandes quantidades de dados. Os *Update Panels* são controlos que actualizam um conjunto de outros controlos, evitando assim os *Postbacks*.

Na abordagem com *Templates* AJAX, as tabelas são preenchidas através de *JavaScript*, evitando também os *Postbacks*, e injectando os registos em cada linha da tabela sem interferência na apresentação do resto da página.

Metodologia

Para comparação das duas abordagens, foi criada uma pequena aplicação *web*, contendo uma tabela, para ser preenchida com milhares de dados. Para forçar a paginação, limitou-se a tabela a dez linhas, ou seja, a dez registos. Esta aplicação foi duplicada para, em cada uma das versões, se poder aplicar uma das duas abordagens que se queriam comparar.

Na abordagem com *Templates* AJAX, foi necessário criar a tabela a partir do controlo *html* "Table" e botões para paginação. Já na abordagem com *GridViews*, apenas é necessário definir os parâmetros que formatam a tabela.

Para a comparação, avaliaram-se os tempos de carregamento e os bytes recebidos através da ferramenta *Fiddler2*, e, manualmente, foram verificados os tamanhos dos códigos-fonte da página das aplicações *web*.

Começaram por ser carregados cerca de 3000 dados. Na abordagem com *GridViews*, não se recorreu aos *ObjectDataSources*. Já na abordagem dos *Templates* AJAX, os dados são carregados em bruto e seleccionados de forma a serem mostrados dez de cada vez, representando-se, assim, a paginação.

De seguida, ao aumentar o carregamento para cerca de 50000 dados, verificou-se que nenhuma das abordagens apresentou uma performance razoável na mostragem dos dados, pois ambas demoraram alguns minutos a carregar a tabela. Por este motivo, na *GridView* decidiu inserir-se um *ObjectDataSource*, para se carregarem apenas os dados da referida página. Já na abordagem dos *Templates* AJAX, optou por se utilizar uma estrutura que, sendo semelhante aos *ObjectDataSources*, contivesse um método que retornasse dez registos para serem inseridos na tabela, e um método que retornasse o número de dados, para se calcular o número de páginas.

O relatório deste estudo pode ser consultado no Anexo VI - Análise de templates.

Conclusões

Com os 3000 dados injectados, os tempos de execução resgistados beneficiam a abordagem de *GridViews*, embora os *Templates* AJAX tenham códigos-fonte mais pequenos e transfiram menos dados.

Quando se aumentou a quantidade de dados para 50000 e foram implementadas as decisões referidas em cima, ambas as abordagens ficaram rápidas, não se notando muitas diferenças, tanto em tempo, como no tamanho dos códigos-fonte.

Concluiu-se, com isto, que os *Templates* AJAX são interessantes, mas não compensam quando comparadas com o uso de *ObjectDataSources,* uma vez que não têm melhor performance nem uma implementação mais fácil. Portanto, decidiu-se que não seria uma boa opção utilizar *Templates* AJAX no *iTelemetry*.

5.5.4 Nova definição na relação "Unidades - Locais"

No sistema, têm-se unidades, que representam os nossos dispositivos que comunicam os dados, e elementos, que representam as entidades que são alvo de telemetria (como, por exemplo, um depósito). Diz-se que uma unidade monitoriza um dado elemento quando esta está responsável por enviar, para os servidores, os dados medidos neste mesmo elemento.

Tanto as unidades como os elementos estão representados em locais, que eram definidos a nível geral, representando zonas. Neste sentido, um elemento só poderia ser monitorizado por unidades que se encontrassem no mesmo local.

Entretanto, houve necessidade de especificar melhor os locais. Tomando um exemplo, inicialmente considerava-se que uma praça com várias casas representava um só local. Pretendeu-se, depois, especificar cada uma das casas como locais distintos. Assim sendo, a restrição existente de uma unidade monitorizar somente elementos do mesmo local perdeu o sentido. Tomando o mesmo exemplo, seria necessário ter uma unidade para cada casa.

Para ultrapassar este facto, surgiu a ideia de poder associar uma unidade a vários locais, os quais a unidade era capaz de monitorizar. No entanto, acabou por ser colocada de lado, uma vez que se perdia a referência à localização de cada unidade (sabia-se apenas onde estavam os elementos).

Deste modo, a importância de distinguir os locais que têm elementos dos que têm unidades levou a que fosse tomada a decisão de associar a cada local uma determinada função: seria de

medição, se o local só contivesse elementos; de comunicação, se o local só contivesse unidades; e de medição e de comunicação, se o local contivesse tanto elementos como unidades. No caso do local não ter nem elementos nem unidades, não se associa qualquer função.

Participação do estagiário

Foi necessário adaptar o *Backoffice*, por forma a poderem associar-se elementos a unidades doutro local. Para isto, criou-se um protótipo, que foi discutido antes de se passar para a implementação. Nesta, houve necessidade de se criar uma pesquisa por unidades do cliente, uma vez que, não havendo restrições quanto às unidades monitorizarem apenas elementos do mesmo local, a única restrição existente era o cliente ao qual pertencem as unidades.

Os requisitos foram actualizados, assim como o plano de testes da aplicação.

5.5.5 Unidades de engenharia

Os sensores, dos quais o *iTelemetry* recebe dados, medem pressões, contagens, níveis, temperaturas, entre outros tipos de medição. As unidades de engenharia nas quais os dados chegam sempre foram, para cada tipo de medição, interpretadas numa só unidade de engenharia. Por exemplo, os dados das contagens sempre foram lidos no *iTelemetry* como metros cúbicos.

No entanto, esta regra deixou de se verificar no *iTelemetry*, podendo os sensores, para o mesmo tipo de medição, emitir dados noutras unidades de engenharia, como são exemplos o litro e o quilograma para as contagens.

Configurações

Como os sensores não transmitem a unidade de engenharia em que estão a ler os dados, é necessário que, no *iTelemetry*, se possa configurar qual é a unidade de engenharia de cada sensor.

Perante esta necessidade, a aplicação do *iTelemetry* responsável pelas configurações teve de ser adaptada, ou seja, o *Backoffice*. Cada *tag*, que representa um canal da unidade (equipamento físico) que recebe dados de um sensor, tem de ter a si associada uma unidade de engenharia, e esta pode ser configurada quer no momento da criação de cada *tag*, quer num momento em que se pretenda editá-la. Outro facto a notar é que as unidades de engenharia têm de ser limitadas por tipo de medição, para que, por exemplo, não se leiam litros num sensor de temperatura.

Apresentação de dados

As aplicações de apresentação de dados do *iTelemetry* indicavam a unidade de engenharia definida para o tipo de medição dos dados. Por exemplo, a aplicação dedicada ao negócio das águas mostrava todas as contagens em metros cúbicos. Esta definição deixou de existir, sendo que, agora, a aplicação pesquisa qual a unidade de engenharia que está associada aos dados.

Em algumas páginas desta aplicação, é possível apresentar dados de vários elementos em simultâneo, por exemplo, num gráfico. Tal facto pode ser complicado de se conseguir quando estão em questão elementos medidos em unidades de engenharia diferentes. Neste caso, por

exemplo, para pôr no mesmo gráfico dados em metros cúbicos e dados em litros, é necessário aplicar uma conversão para a unidade de engenharia por defeito do tipo de medição.

Conversões

As conversões no *iTelemetry* são efectuadas linearmente, podendo aplicar-se a equação da recta:

$$y = m * x + b$$

onde y representa o resultado da conversão, x representa o dado a converter, e m e b os conhecidos parâmetros da equação da recta que permitem a conversão.

No entanto, há conversões que não são sempre efectuadas com os mesmos parâmetros m e b, podendo depender, por exemplo, da densidade e da temperatura, no caso da conversão de metros cúbicos para quilogramas. Sendo assim, para além de se ter uma conversão generalizada, podem ser definidos diferentes parâmetros m e b para diferentes elementos.

Trabalho planeado

Com esta nova estrutura das unidades de engenharia, ficam abertas novas possibilidades de melhorias. Está previsto, como tarefa futura no *iTelemetry*, disponibilizar nas suas aplicações de apresentação de dados a possibilidade de escolher a unidade de engenharia em que os dados são apresentados, independentemente da unidade de engenharia em que estes são obtidos. Obviamente, só serão disponibilizadas as unidades de engenharia cuja conversão seja possível a partir da unidade de engenharia original dos dados.

5.5.6 Revisões, verificações e testes

A metodologia *SCRUM* utilizada inclui entregas regulares do produto. Estas entregas incluem o preenchimento de fichas de revisão e de verificação de requisitos, e execução de testes.

A equipa do *iTelemetry*, tal como sugere a metodologia *SCRUM*, é multi-facetada. Assim, como cada tarefa pode ser feita por qualquer elemento da equipa, houve participação do estagiário tanto nas fichas de revisão e verificação de requisitos, como em testes das aplicações do *iTelemetry*.

Capítulo 6 Considerações finais

O presente documento apresenta o trabalho efectuado durante o estágio curricular referente ao Mestrado em Engenharia Informática. As principais tarefas efectuadas foram: *ISA Scheduler*, *Service Manager*, reformulação da arquitectura e validação do sistema.

O *ISA Scheduler*, com as alterações efectuadas, tornou-se mais fiável, facto demonstrado pelo *feedback* positivo dado por quem usa a ferramenta com maior regularidade. O principal motivo de satisfação deve-se à existência de um muito menor número de falhas durante as execuções dos processos.

O *Service Manager* possibilita a vários membros da ISA uma mais fácil gestão dos processos do *ISA Scheduler*, permitindo um ganho de tempo diário por parte dos mesmos.

Com a nova arquitectura lógica, o *iTelemetry* obteve uma melhor divisão. Agora, a plataforma está mais organizada e está mais apta para evoluções futuras.

A validação do sistema, apesar de ainda não estar concluída, torna os dados mais fiáveis, corrigindo dados inválidos. Para isso, é importante que se consigam corrigir o maior número possível de dados inválidos, retirando trabalho aos responsáveis pelo acompanhamento do sistema.

O *iTelemetry* evoluiu em termos de ambição e longevidade, e está cada vez mais perto de substituir a antiga plataforma de telemetria da ISA. Com este passo, é garantida a entrada nos mais altos patamares do mercado internacional, como pretendido.

Outro aspecto importante do estágio foi a utilização da metodologia *SCRUM*, que permitiu um planeamento adequado e uma boa integração do estágio no projecto. Outros pontos positivos que o *SCRUM* mostrou, não só para o estágio, como para a engenharia de *software* em geral, foram:

- uma equipa com elementos multi-facetados evita as dependências de pessoas especializadas em certas funções, e evita que tarefas do mesmo tipo tenham de ser feitas todas por um só elemento;
- as reuniões diárias permitem que os elementos se actualizem do estado das tarefas, evitando-se que várias pessoas peguem na mesma tarefa; também permite que tarefas dependentes possam ser feitas em simultâneo, com os seus responsáveis a partilharem as evoluções entre si;
- as *sprints* de poucas semanas fazem com que os elementos da equipa saibam sempre o conjunto de tarefas que há a fazer, e fazem com que estes tenham presente a hierarquia de prioridades das tarefas.

Em termos gerais, este estágio contribuiu para a evolução da plataforma *iTelemetry*, ajudando no desenvolvimento de novas funcionalidades, tanto a nível aplicacional, como a nível de processamentos.

Quanto ao futuro, é de notar que o *iTelemetry* ainda não está completo. Um módulo de manutenção de sistemas para técnicos no terreno ainda é um objectivo futuro, tal como a continuação do módulo de validação do sistema. Ainda em desenvolvimento no final deste estágio encontra-se a reformulação do módulo de recepção de dados, que, anteriormente, era composto pelo serviço *Data Service*, e que, no final destes desenvolvimentos, para além de deixar de conter este serviço, englobará um outro serviço de recepção das comunicações e outro de sincronização dos dados.

Capítulo 7 Referências bibliográficas

[1] http://books.google.pt/books?id=3LK44UI0YCEC&printsec=frontcover&dq=telemetry&hl=pt-PT&ei=VwjoTZrBD9C7hAeJ5sm8Cg&sa=X&oi=book_result&ct=result&resnum=1&ved=0CC4Q6AEwAA#v=onepage&q&f=false - Telemetry systems engineering - Google Livros;

[2] http://scrumalliance.org/pages/what_is_scrum - Scrum Alliance - What Is Scrum?;

[3] http://msdn.microsoft.com/en-us/magazine/cc188690.aspx - Design Patterns: Model View Presenter;

[4] http://www.imrlpg.com/download.php?id=73 - IMR Server 4 - Enterprise Suite;

[5] http://www.mydatanet.at/en/technologie/mydataweb.html - myDatanet - Technology - myDataweb;

[6] http://www.netbiter.com/products/argos.shtml - Netbiter Argos Data Center;

[7] GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John - "Design Patterns: Elements of Reusable Object-oriented Software", Massachusetts: Addison-Wesley, 2009. p.107-116

[8] http://www.homeridersystems.com/en/ - Homerider Systems, energies under control;

[9] http://www.homeridersystems.com/en/applications/fusion-application-homerider - Fusion - HOMERIDER application;

[10] http://msdn.microsoft.com/en-us/netframework/aa663324 - Windows Communication Fundation.

Todas as restantes referências consultadas fazem parte dos documentos internos da ISA.