Mestrado em Engenharia Informática
Estágio
Relatório Final

Aplicação Para Dispositivos Móveis de Uma Solução de Monitorização de IT

Nuno Filipe Rosa Dias Simões
nfds@student.dei.uc.pt

Orientadores:
Luis Macedo
Tiago Jorge
Data: 04 de Setembro de 2013

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

Resumo

Com a crescente necessidade de ter serviços de alta disponibilidade, com as aplicações cada vez mais *web-based* e com a constante informatização das empresas, torna-se imperativa a possibilidade de monitorizar remotamente e de forma centralizada, por forma a conseguir tornar residuais os períodos de falha das infra-estruturas de IT.

O objectivo deste estágio era então o de desenvolver uma aplicação robusta e visualmente apelativa, que permita a consulta das métricas e alarmes de uma solução de monitorização de IT, em dispositivos móveis.

O projecto tem moldes inovadores, ao capacitar os gestores de Infra-estruturas Tecnológicas de gerir os seus sistemas de forma centralizada, nos seus telemóveis e tablets. Outro dos pontos fortes desta solução, é a possibilidade de gerir o sistema à distância, de qualquer ponto do mundo, bastando para tal ter uma ligação à internet.

Palavras-Chave

Alarmes; Android; "Aplicações Móveis"; Estágio; IT; Métricas; Monitorização; "Sell as a Service"; "Sistemas Distribuídos".

Índice

Lista de Figuras

Glossário

ACRA	*Application Crash Reports for Android*. Mecanismo para reportar erros em aplicações Android.
Android	Sistema operativo para dispositivos móveis desenvolvido pela Google.
Alarmes /Alarms	Evento despoltado quando um determinado valor definido é atingido por uma métrica.
Curva de aprendizagem	Tempo tomado a aprender uma determinada tecnologia.
DEI	Departamento de Engenharia Informática.
Escalabilidade	Capacidade de adaptação ao crescimento de necessidades de recursos.
FCTUC	Faculdade de Ciências e Tecnologia da Universidade de Coimbra.
Framework	Conjunto de bibliotecas de facilitação ao desenvolvimento de aplicações.
GCM	Refere-se ao Google Cloud Messaging, serviço de notificações para o sistema operativo Android.
iOS	Sistema operativo para dispositivos móveis desenvolvido pela Apple.
IT	Tecnologias de informação.
Métricas / Metrics	Valor que se pode recolher de um determinado elemento físico ou de *software* (e.g. utilização de CPU).
Multi-plataforma	Aplicação que funciona em diversos sistemas operativos distintos com o mesmo código fonte.
NoSQL	Bases de Dados não relacionais.
ROI	*Return of Investment*: Valor que se consegue obter através de um investimento
SaaS	Diminutivo de "*Software as a Service*". Refere-se a software centralizado geralmente acedido por utilização de um *web browser*.
Targets	Nós de rede ou *softwares* em nós de rede sobre os quais se pode operar monitorização.
WEB	Diminutivo de Internet.

Capítulo 1 - Introdução

O presente documento relata o trabalho desenvolvido no projecto *"Aplicação Para Dispositivos Móveis de Uma Solução de Monitorização de IT"*, realizado no âmbito da disciplina de estágio, do Departamento de Engenharia Informática (DEI) da Faculdade de Ciências e Tecnologia da Universidade de Coimbra (FCTUC), integrando a solução Tellai da empresa guberni. A supervisão deste projecto de estágio ficou a cargo do Professor Luis Macedo (DEI) e do Eng. Tiago Jorge (guberni).

A aplicação desenvolvida, tem por objectivo permitir visualizar os alarmes e métricas de clientes já existentes na solução Tellai. Então, e por forma a realizar um verdadeiro projecto de engenharia, o elemento estagiário usou-se do conhecimento adquirido na sua formação académica, complementando-o com o conhecimento prático proveniente da convivência na equipa de desenvolvimento da guberni, para planear e executar todas as fases inerentes ao processo de engenharia, com diligência, conforme relatado no presente documento.

Assim, e navegando pela estrutura do documento temos os seguintes capítulos:

- **Estado da Arte:** Onde será dado conhecimento sobre o estado da solução Tellai aquando do início do estágio, para melhor entender como se encaixou a aplicação móvel desenvolvida. Neste capítulo identifica-se ainda os elementos da equipa.

- **Objectivos:** Apresenta-se aqui todos os objectivos relativos ao projecto, quer a nível de *software*, quer a nível cognitivo do estagiário.

- **Tecnologias:** Levantamento das tecnologias utilizadas no projecto com as devidas fundamentações.

- **Plano de Trabalho:** Explicação de como se procedeu à gestão das iterações da fase de desenvolvimento, bem como a identificação da sua dispersão temporal.

- **Trabalho Realizado:** Identificação dos cenários de utilização, dos requisitos de *software*, da infra-estrutura final da solução, da arquitectura de *software*, da organização do código fontes nas diferentes soluções e dos testes de usabilidade.

- **Conclusão:** Considerações finais sobre o trabalho efectuado.

Capítulo 2 - Estado da Arte

2.1 - Enquadramento

A guberni actua na área de Gestão de Infra-estruturas e Serviços de Tecnologias de Informação. Presta serviços de consultoria nos principais produtos de gestão para esta área, nomeadamente produtos da Hewlett-Packard (HP) e da Computer Associates (CA), e desenvolve soluções de *software* que colmatam necessidades não fornecidas por estes produtos, ora desenvolvendo soluções que se integram neles, permitindo optimizar o seu funcionamento e obter um maior retorno do investimento (ROI), realizado neste tipo de produtos, ora desenvolvendo soluções completamente independentes destes, à medida do cliente, e que permitem gerir de uma forma mais eficaz os recursos tecnológicos das empresas.

O projecto de estágio "Aplicação Para Dispositivos Móveis de Uma Solução de Monitorização de IT" realizou-se na divisão de desenvolvimento de *software* da guberni e serve para visualizar, em dispositivos móveis, métricas e alarmes de diversos sistemas previamente recolhidas pela solução previamente existente, em que o projecto de estágio se integrou, denominada de Tellai.

2.2 – O Paradigma

A monitorização de sistemas informáticos, embora não seja algo de novo, é uma área que tem tido fortes desenvolvimentos nos últimos anos. Motivado largamente pela cada vez maior utilização de sistemas distribuídos, de aplicações SaaS e da grande dispersão geográfica das infra-estruturas das empresas, torna-se extremamente necessária a possibilidade de uma gestão centralizada destas estruturas, por sistemas altamente robustos e de elevada disponibilidade. A guberni, com o sólido *know-how* que dispõe neste mercado, desenvolveu o seu produto de monitorização, com o seu próprio paradigma.

Assim, e numa abordagem muito clara, a monitorização pelo sistema da guberni assenta, de uma forma simplificada em, ***Monitor Groups*** que contêm ***Monitors***, *Monitors* esses que são constituídos de ***Metrics***. Associados a esses *Monitor Groups* estão pares ***Target/SmartAgent*** que definem em quem e a partir de onde se realiza a monitorização.

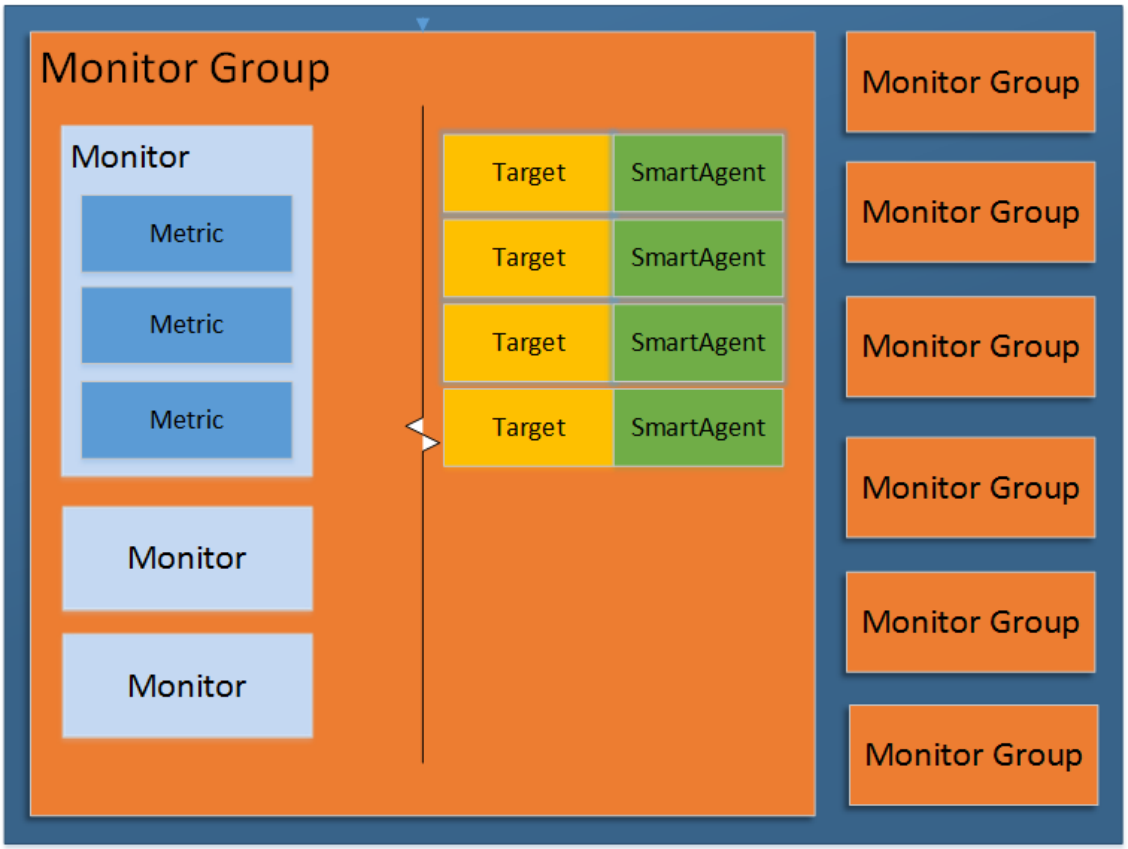

Figura 1 – Interligação dos vários elementos da aplicação Tellai

2.2.1 – Target

Um *Target*, é um elemento passível de ser monitorizado, ou seja, um nó de rede ou um software num nó de rede de onde podem ser recolhidas métricas.

2.2.2 – SmartAgent

Um *SmartAgent* é um software que tem como responsabilidade recolher métricas dos Targets, enviando-as para o software Tellai, para posterior tratamento.

2.2.4 – Metric

Uma métrica ou *Metric*, é um valor, numérico ou não, recolhido de um target (e.g.: percentagem de utilização da CPU).

2.2.5 Monitor

Um *Monitor* é um agrupamento de Métricas, que geralmente são do mesmo tipo ou âmbito. Por exemplo, um Monitor chamado CPU poderá conter um conjunto de métricas relacionadas com CPU.

2.2.6 Monitor Group

Um *Monitor Group*, conceptualmente, é o que define a relação entre o que vai ser monitorizado, onde vai ser e por quem vai ser. Assim, tipicamente, *Monitors* que se aplicam a tipos de *Target* iguais são agrupados num *Monitor Group*, associando-lhe então os *Targets*

aos quais esses *Monitors* se aplicam, e os *SmartAgents* responsáveis por fazer a recolha das *Metrics*.

2.3 - O Projecto

A aplicação desenvolvida enquadra-se numa solução maior, como um módulo integrante.

Passamos então a explicar primeiramente a arquitectura geral da solução na qual o projecto foi integrado, a fim de melhor entender o enquadramento do projecto de estágio.

A solução já existente, na qual o projecto *"Aplicação Para Dispositivos Móveis de Uma Solução de Monitorização de IT"* foi integrado, subdivide-se em várias áreas distintas e independentes, mas ligadas entre si, trabalhando como um todo:

- **SmartAgent:** esta aplicação é responsável pela recolha das métricas nos vários nós da rede e pelo envio destas para o ***Data Reception***, comunicando ainda com o ***Configuration Manager*** para receber novas configurações a si associadas.
- **Data Reception:** o ***Data Reception*** serve somente como porta de entrada dos dados enviados por cada ***SmartAgent***, validando se o formato dos dados enviados é correcto. Caso isto se verifique, os dados são armazenados numa base de dados (***Client DataStore***), podendo esses dados ser métricas, logs de execução do ***SmartAgent*** e infos. As infos podem então ser nós descobertos na rede, actualizações de propriedades de nós já existentes ou *snapshots* de servidores. No caso de os dados enviados serem métricas, estas são ainda colocadas numa fila, ***Rabbit Queue,*** para serem posteriormente processadas por uma aplicação denominada de ***Alarm Manager***.
- **Configuration Manager:** o ***Configuration Manager*** é o responsável por consultar a base de dados de configurações da aplicação (***Configuration DataBase***), e comunicar estas configurações para os ***SmartAgents***.
- **Alarm Manager:** a aplicação ***Alarm Manager*** encarrega-se de analisar cada métrica colocada na pilha ***Rabbit Queue***, verificando se o valor enviado encaixa em algum alarme configurado pelo cliente. Caso se verifique, é gerado um registo de alarme que é arquivado na ***Client DataStore***.
- **Processing:** o ***Processing*** é formado por um conjunto de serviços, invocados por um calendário de execução previamente definido, sendo eles:
    - **Metrics Aggregation:** O serviço ***Metrics Aggregation*** encarrega-se de ler as métricas armazenadas na ***Client DataStore*** e de as agregar temporalmente, por forma a facilitar a consulta destas métricas por período de tempo, por parte do utilizador.
    - **Info Processing:** O serviço ***Info Processing*** processa os registos do tipo Info armazenados na ***Client DataStore*** que contêm dados de novos nós descobertos na rede ou de actualização de propriedades de nós já existentes no sistema.
    - **Client Account:** Este serviço analisa o estado das contas dos clientes, verificando se estas ainda são válidas.
    - **Data Maintenance:** É o serviço que efectua a manutenção da ***Client DataStore***, removendo dados que sejam mais antigos que o tempo de retenção definido para o cliente.
    - **SmartAgent Checker:** É possível configurar a aplicação para gerar um alarme e enviar notificação ao cliente no caso de um ou mais ***SmartAgents*** não comunicarem por um período superior ao estipulado, sendo que este

serviço encarrega-se de fazer essa verificação e de notificar o ***Alarm Manager,*** por via da fila ***Rabbit Queue,*** para ser gerado um alarme.

- **SLA Aggregation:** É o serviço que efectua os cálculos de SLAs das métricas do cliente.

- **UI:** o UI é composto pelo conjunto de páginas web pelas quais o utilizador pode alterar configurações da sua infra-estrutura, visualizar métricas, alarmes e muitas outras funcionalidades da aplicação.

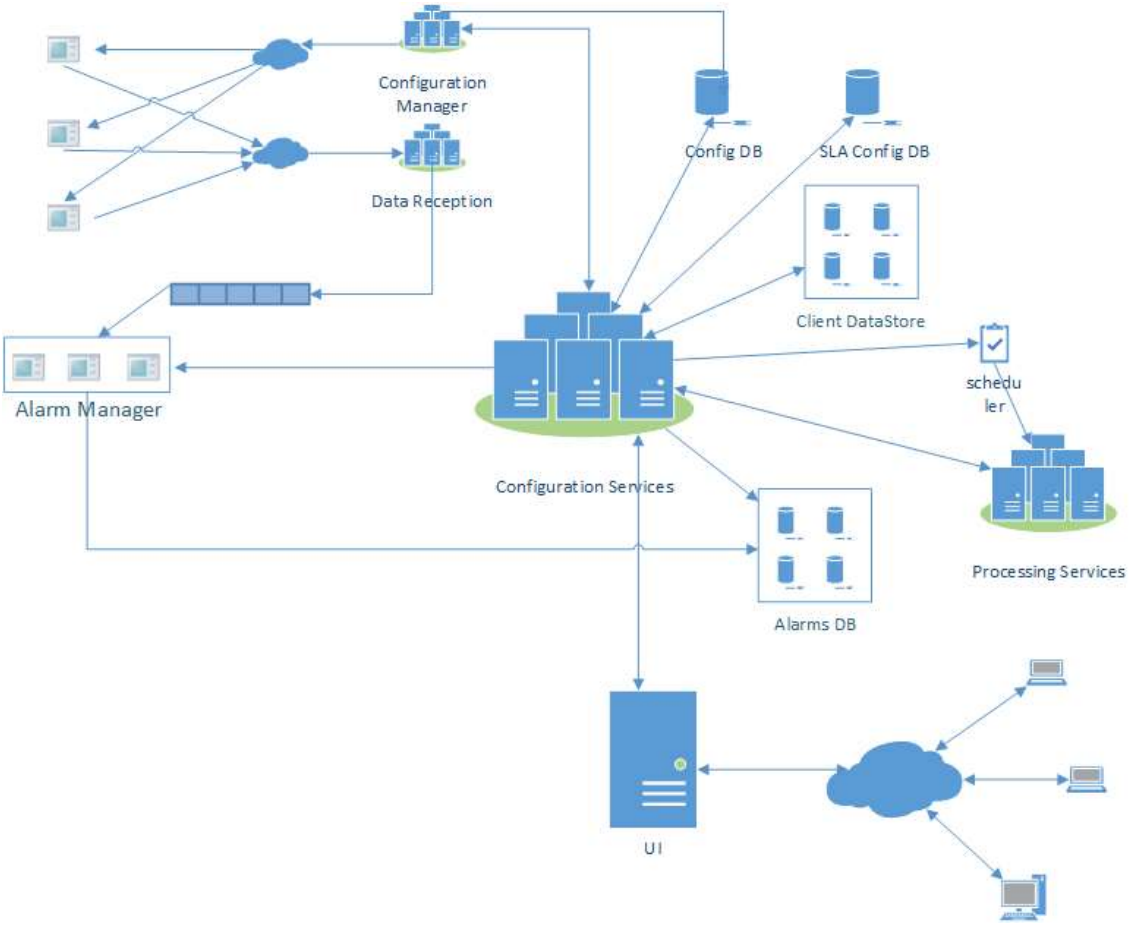

Figura 2 – Visão Geral da Infra-Estrutura actual do software

2.4 - Equipa

Este projecto esteve destinado a uma pessoa (Nuno Simões), sendo apoiado por dois membros da guberni (Eng. Pedro Laranjo e Eng. Tiago Jorge).

Dada a reduzida dimensão da equipa, existiram alguns elementos com mais do que uma função, sendo que ainda assim, isso não conferiu um risco ao sucesso do projecto. Temos então as seguintes competências:

- **Pedro Laranjo:** *Product Owner*;
- **Tiago Jorge:** *Project Manager*;
- **Nuno Simões:** *Developer*, *Test Manager* e *Technical Manager*.

Capítulo 3 - Objectivos

3.1 - Objectivos

O estágio tinha como objectivo, desenvolver uma aplicação robusta e visualmente apelativa de visualização de métricas e alarmes de diversos nós da rede para posterior análise. A princípio previa-se desenvolver a aplicação tanto para Android, como para iOS, tendo-se decidido desenvolver somente para Android, aquando do desenvolvimento, conforme é explicado no Capitulo 7.

Seguidamente, apresenta-se aquela que era a lista de objectivos da aplicação, antes do início do desenvolvimento:

- **Autenticação:** A aplicação deverá permitir a autenticação de contas previamente criadas na versão web do software Tellai.
- **Visualização de métricas:** deverá ser possível visualizar as métricas recolhidas dos vários nós da rede do cliente em gráficos, tabelas e outros meios que sejam convenientes.
- **Visualização de alarmes:** a visualização dos alarmes deve ser possível por severidade e por Target sendo que deve também ser notificado o utilizador em tempo real, cada vez que é gerado um novo alarme.
- **Visualização de target:** Pretende-se que seja possível a consulta dos targets que o cliente tem na sua infra-estrutura, bem como os seus detalhes.
- **Parametrização da aplicação:** deve ser possível parametrizar na aplicação o utilizador e password que se autentica automaticamente, bem como se o utilizador pretende, ou não, receber no seu telemóvel, notificações em tempo real dos alarmes.

No que concerne aos objectivos cognitivos do elemento estagiário, pretendia-se que obtivesse um contacto mais fundamentado com o ambiente empresarial e de trabalho em equipa, adquirindo melhores capacidades, não só técnicas, mas também de gestão e organização, facto que considero ter sido cumprido.

Em relação ao ciclo de vida de desenvolvimento, apresentam-se de seguida as fases e os períodos de tempo que eram previstos, apresentando-se também o tempo real:

1. ***Inception*:** Definição da ideia e dos objectivos esperados, bem como o planeamento das fases seguintes e de como se procederiam.

   Previsto: 28-01-2013;

   Real: 28-01-2013.

2. ***Design*:** Esquematização dos ecrãs e seus fluxos por meio de mockups e posterior desenho da interface gráfica.

   Previsto: 22-02-2013;

   Real: 22-02-2013.

3. ***Development***: Fase de desenvolvimento da aplicação em bruto, corrigindo *Major Bugs* centrando principalmente na funcionalidade.

   Previsto: 19-04-2013;

   Real: 03-05-2013.

4. ***Stabilization***: Fase em que em paralelo ao desenvolvimento da aplicação, começam a ser lançadas versões para testes.

   Previsto: 31-05-2013;

   Real: 12-07-2013.

5. ***Deployment***: Finalização da aplicação e publicação da versão estável.

   Previsto: 14-06-2013;

   Real: 02-08-2013.

Capítulo 4 - Tecnologias

O presente capítulo serve para apresentar as tecnologias escolhidas, bem como os motivos da sua escolha.

4.1 - MySQL

Na necessidade de ter uma base de dados para guardar dados de suporte à aplicação, a opção recaiu sobre o MySQL. Embora a solução já existente tenha como SGBD o MySQL, o que influenciaria sempre a escolha, isso não invalidou que fossem feitos testes para melhor aferir a escolha.

Para a tomada de decisão, foi usado o *Benchmark* TPC-H, gerando 21 *queries* de teste com o software DBGEN (disponível em http://www.tpc.org/tpch/).

Os factores de decisão foram os seguintes, por respectiva ordem de importância: Preço de licenciamento, *Performance*, documentação disponível e aceitação no mercado.

Dos parâmetros preço de licenciamento, documentação disponível e aceitação no mercado obtiveram-se dois SGBDs muito semelhantes, sendo eles o MySQL e o PostgreSQL. Assim, restou a performance como elemento decisivo, recaindo a decisão para o MySQL, pelos resultados médios de 5 medições a cada *query* do TPC-H, ilustrados na figura 4.

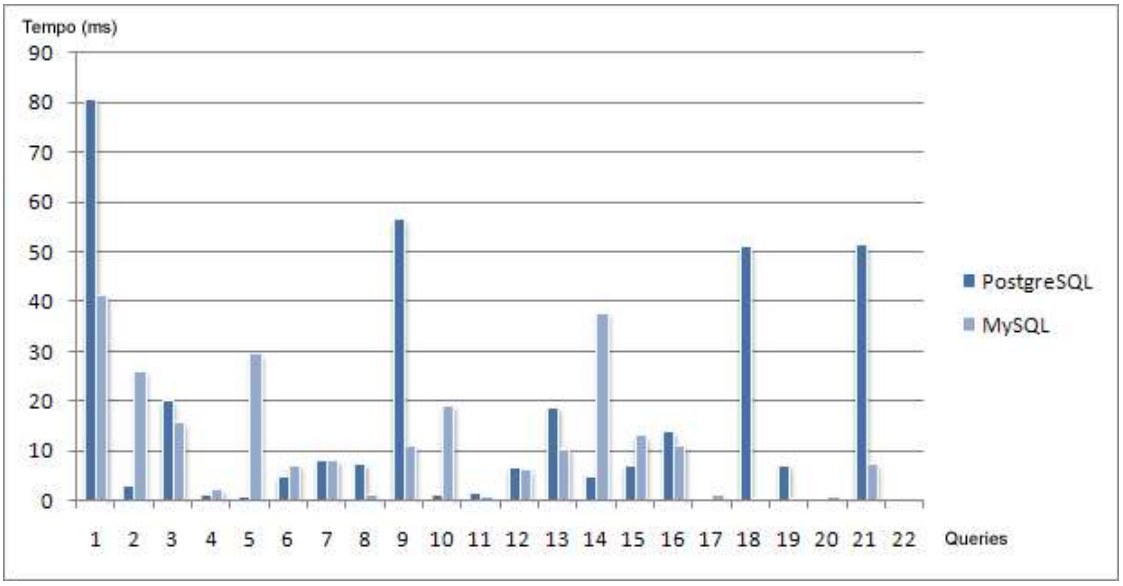

Figura 3 – Média aritmética das medições de cada query em ordem ao tempo

As tabelas com as medições completas, bem como o gráfico do desvio padrão poderão ser consultados no Anexo B.

4.2 - SQLite

A utilização do SQLite tornou-se mandatória para a utilização do TellaiMobile, sendo essa a forma nativa de armazenar dados no sistema operativo Android.

4.3 - .NET

A escolha para a componente ***TellaiMobile Web*** recai sobre a Framework .NET MVC4 da Microsoft. Esta escolha deve-se a factores como:

- **Escolha do Product Owner:** embora o *product owner* tenha dado liberdade de análise sobre as tecnologias a usar, demonstrou sempre uma forte tendência para o .NET, principalmente por ser a tecnologia usada na empresa, permitindo assim a futura integração desta aplicação com outras soluções desenvolvidas na empresa.

- **Curva de aprendizagem:** o estagiário revela uma razoável familiarização com a tecnologia, permitindo assim desenvolver uma solução de menor custo e maior robustez, potenciando assim o ROI associado ao projecto.

Para o desenvolvimento da aplicação móvel será igualmente utilizada a tecnologia .NET C#, pelos motivos enunciados em 4.7.

4.4 – MongoDB

A base de dados MongoDB é uma base de dados *NoSQL* com cada vez mais aceitação no mercado, quer pela sua comprovada performance, quer pela sua escalabilidade. Esta base de dados é actualmente usada para armazenar os alarmes, sendo que o TellaiMobile interage com esta, quer pelo módulo TellaiMobile Web aquando da visualização de alarmes, que para fazer *Push* (envio das notificações para os dispositivos móveis) das notificações em tempo real, desta feita por intermédio de uma aplicação desenvolvida sobre nodeJS.

4.5 – RabbitMQ

O RabbitMQ é um sistema de mensagens, já utilizado na aplicação Tellai, que implementa o paradigma "Produtor-Consumidor". Este sistema é robusto e de fácil utilização, sendo que foi utilizado para implementar a fila de alarmes enviados para o sistema de notificações GCM.

4.6 – Nodejs

O nodejs é uma plataforma relativamente recente, tratando-se de um servidor baseado em JavaScript, que permite construir aplicações rápidas e escaláveis. Esta plataforma foi usada para implementar o sistema de envio das notificações em tempo real para os dispositivos móveis. Este serviço está então encarregue de ler as mensagens (alarmes) da fila Rabbit, enviando-as para o sistema de notificações Google Cloud Messaging.

Foi ainda considerada a possibilidade de utilizar o PushSharp, mas visto o nodejs já estar a ser utilizado na solução Tellai, e ter um consumo de recursos muito reduzido, essa opção foi colocada de lado.

4.7 – Xamarin

Xamarin é uma Framework que permite desenvolver aplicações para sistemas Android e iOS usando .NET C#. Para desenvolver para o sistema Android geralmente usa-se Java e para iOS usa-se Objective C, logo Xamarin permite cortar no tempo de aprendizagem e de desenvolvimento, podendo reutilizar-se código nas duas plataformas, diferenciando-se então somente na componente da interface gráfica. Tem ainda outra grande vantagem, que é o *know-how* na tecnologia .NET, quer por parte do estagiário, que poderá partilhar a aprendizagem com a do desenvolvimento do módulo TellaiMobile Web, quer pelos restantes elementos da guberni, que têm já larga experiência em .NET, permitindo-lhes ter maior facilidade em desenvolver sobre o produto finalizado.

Convém ainda referir que antes da tomada de decisão foram efectuados também testes utilizando PhoneGap e KendoUI, visto ser uma metodologia que também permitia facilmente desenvolver para multi-plataforma sem elevado custo. No entanto, após alguns testes, esse caminho foi abandonado, uma vez que as aplicações desenvolvidas sobre PhoneGap se revelam muito lentas e pouco responsivas, não cumprindo o esperado de uma aplicação deste tipo.

4.8 - Ferramentas a Utilizar

A escolha das ferramentas a utilizar é sempre importante, pois delas depende directamente o tempo e qualidade de implementação da solução. Assim, foram definidos alguns parâmetros de suporte à escolha, sendo eles: a dimensão da comunidade de utilizadores e quantidade de documentação de suporte disponível, curva de aprendizagem associada à utilização da ferramenta e capacidade de adaptabilidade da ferramenta às necessidades do utilizador, por meio de *plugins* ou outras funcionalidades configuráveis.

Assim a escolha recaiu sobre as seguintes ferramentas:

- **Visual Studio:**
  Versão: 2012

Página Web: http://www.microsoft.com/visualstudio/ptb/visual-studio-update

- **MySQL:**
  Versão: 5.5.29
  Página Web: http://dev.mysql.com/
- **.NET:**
  Versão: 4.5
- **Mono for Android**
  Página Web: http://xamarin.com/monoforandroid

Capítulo 5 - Plano de Trabalho

De seguida apresenta-se aquela que era a calendarização prevista das várias fases do projecto, juntamente com a calendarização realmente realizada.

Como é visível na Figura 2 (ver imagem aumentada no Anexo A), o projecto dividiu-se em duas fases principais: a fase de investigação/planeamento e a fase de desenvolvimento. Não obstante, a fase de planeamento estendeu-se igualmente à fase de desenvolvimento, sendo impossível dissociar estes dois parâmetros ao adoptar a metodologia *Scrum*.

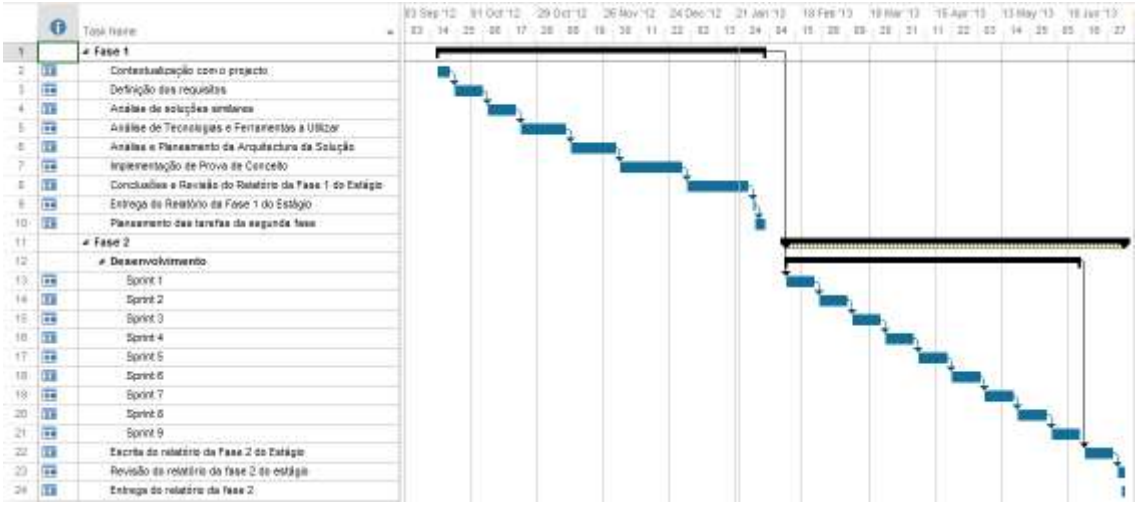

Figura 4 - Planeamento Global (imagem aumentada no Anexo A)

5.1 - Calendarização

A seguinte tabela descreve as tarefas da Fase 1 do estágio:

Tarefa	Início	Fim
Contextualização com o projecto: Tomada de conhecimento do âmbito do projecto, dos objectivos esperados, de especificidades da área de *IT management* e da equipa de trabalho. Primeiro contacto com a arquitectura do produto.	17-09-2012	21-09-2012
Definição dos requisitos: Especificação dos requisitos da solução em *user stories*.	24-09-2012	05-10-2012
Análise de soluções similares: Levantamento e análise de soluções que de alguma forma tenham funcionalidade semelhante à proposta.	08-10-2012	19-10-2012
Análise de Tecnologias e Ferramentas a utilizar: Escolha das tecnologias a usar na solução (*Framework*, Bases de Dados, etc.) e das ferramentas necessárias ao	22-10-2012	09-11-2012

desenvolvimento, testes e outras operações.		
Análise e Planeamento da Arquitectura da Solução: Especificação da infra-estrutura necessária à solução de *software* para executar em produção.	12-11-2012	30-11-2012
Implementação de Prova de Conceito: Primeiros testes de implementação para aferir a aplicabilidade das tecnologias escolhidas, na arquitectura em questão.	03-12-2012	28-12-2012
Conclusões e Revisão do Relatório da Fase 1 do Estágio	31-12-2012	25-01-2013
Entrega do Relatório da Fase 1 do Estágio	28-01-2013	
Planeamento das tarefas: Planeamento do conjunto de tarefas a efectuar na segunda fase.	29-01-2013	01-02-2013

A Fase 2, foi a fase de desenvolvimento do projecto, e estava previsto dividir-se em 9 *sprints* de duas semanas, sendo que acabou por se extender por 12 *sprints* de duas semanas e 1 *sprint* de uma semana. Posterior a essas 13 *sprints*, existiu três semanas para actualização do relatório de estágio. É de notar que o planeamento de cada *sprint* foi efectuado na correspondente reunião *Planning Game*, conforme explicado na próxima secção deste relatório.

A tabela seguinte elucida sobre a disposição temporal das várias *sprints* na fase 2 do projecto:

Tarefa	Início	Fim
Sprint 1	11-02-2013	22-02-2013
Sprint 2	25-02-2013	08-03-2013
Sprint 3	11-03-2013	22-03-2013
Sprint 4	25-03-2013	05-04-2013
Sprint 5	08-04-2013	19-04-2013
Sprint 6	22-04-2013	03-05-2013
Sprint 7	06-05-2013	17-05-2013
Sprint 8	20-05-2013	31-05-2013
Sprint 9	03-06-2013	14-06-2013
Sprint 10	17-06-2013	28-06-2013
Sprint 11	01-06-2013	12-06-2013
Sprint 12	15-07-2013	26-07-2013
Sprint 13	29-07-2013	02-08-2013

Escrita do relatório da Fase 2 do Estágio	15-08-2013	23-08-2013
Revisão do relatório da Fase 2 do Estágio	26-08-2013	30-08-2013
Entrega do Relatório Final de estágio	04-09-2013	

5.2 - Planeamento

5.2.1 - Responsabilidades

Seguidamente apresenta-se a lista de responsabilidades do estagiário enquanto elemento do projecto:

- Participação activa no planeamento do projecto;
- Definição da arquitectura da solução de *software*;
- Participação nas "*daily scrum meetings*";
- Elaboração do Plano de Testes;
- Levantamento dos requisitos em *user stories*;
- Elaboração dos Modelos de Dados;
- Codificação da solução de *software*;
- Elaboração do Manual de Utilizador;
- Desenho detalhado da solução;
- Testes e validação da solução;
- Formação em novas tecnologias e metodologias.

5.2.2 - Scrum e XP

No projecto, foram utilizadas as metodologias Scrum e XP, permitindo assim agilizar o processo de desenvolvimento.

Na implementação desta metodologia tivemos em conta os seguintes princípios:

- ***Planning Game***: Efectuados de duas em duas semanas, no fim de cada *Sprint*. Estas reuniões serviram para fazer a demonstração do produto gerado nessa *Sprint*, obtendo assim *feedback* por parte do cliente sobre o resultado esperado e obtido, planeando então assim a *sprint* seguinte.
- ***Simple Design***: Este é um dos princípios do XP e foi fundamental para o sucesso do projecto. O objectivo foi o de desenvolver sempre e somente os requisitos pedidos pelo cliente, de forma a evitar a consequente perda de tempo.
- ***Customer Tests***: é um conjunto de testes definidos pelo cliente juntamente com o estagiário, que avaliam a aceitação ou não de um determinado requisito. Estes testes

levaram a que se percebesse rapidamente no projecto que novas funcionalidades eram necessárias.

- ***Sustainable Pace*:** faz referência a um ritmo de trabalho saudável, procurando sempre não efectuar horas extra, a menos que introduza comprovado valor ao projecto. Este processo garante qualidade e bom ambiente de trabalho evitando a baixa produtividade associada ao cansaço.

- ***Stand-up Meetings*:** são reuniões diárias de curta duração onde se abordaram as tarefas realizadas, por realizar e dificuldades encontradas, procurando sempre resolver potenciais obstáculos ao desenrolar do projecto.

- ***Coding Standards*:** embora a equipa de desenvolvimento somente fosse constituída pelo estagiário, tornou-se importante o cumprimento de padrões de desenvolvimento para que no futuro outra pessoa possa desenvolver sobre o trabalho efectuado com facilidade.

- ***Refactoring*:** este processo de melhoramento contínuo do código desenvolvido ajuda à compatibilidade do código no processo de integração, minimiza a introdução de erros, melhora a legibilidade do código e evita a duplicação de funções.

- ***Continuous Integration*:** a integração contínua implica que as novas funcionalidades desenvolvidas sejam sempre integradas no projecto quando concluídas, minorando assim a possibilidade de conflitos.

Assim, o processo de gestão do projecto decorreu da seguinte forma:

- No princípio do projecto foram definidas as *stories* (agrupamento de tarefas) constituintes do projecto por forma a produzir o nosso *product backlog* (total de *stories* do projecto);

- Em cada *Planning Game* foram escolhidas as *stories* do *product backlog* que integraram a *sprint* correspondente, constituindo assim o nosso *sprint backlog* (*stories* que integram uma *sprint*);

- Diariamente e às 10h00 teve sempre lugar a *daily scrum meeting* (ou *Stand-up Meeting*) para registo do estado do projecto;

- No final de cada *sprint* existiu sempre a demonstração do produto desenvolvido e a análise dos desvios ao resultado previsto.

A figura 4 ilustra o desenrolar do processo, numa perspectiva temporal.

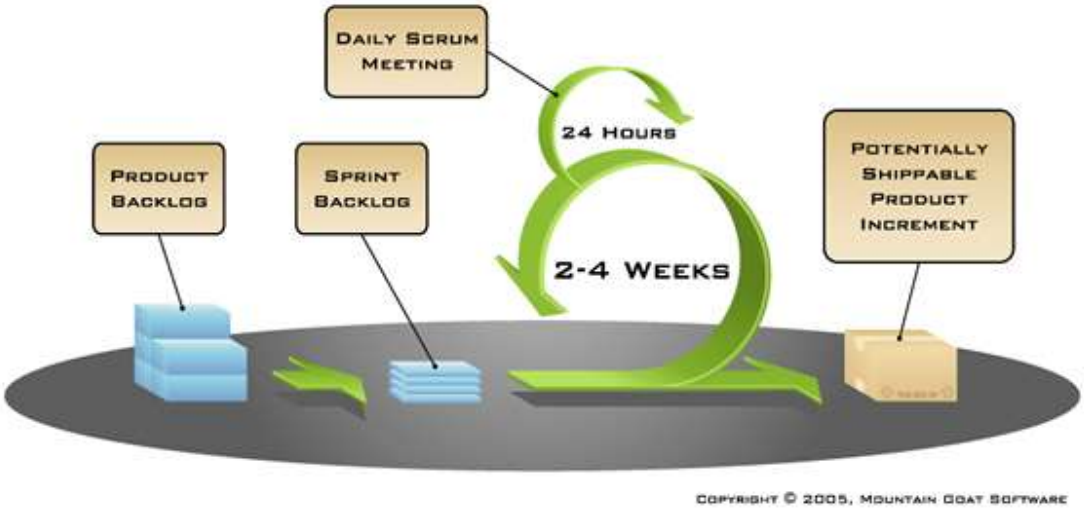

Figura 5 - Exemplo do processo Scrum

Capítulo 6 - Trabalho Realizado

6.1 - Cenários de Utilização

A aplicação serve para monitorizar infra-estruturas de qualquer dimensão, permitindo ter controlo de um vasto conjunto de métricas e alarmes, remotamente em dispositivos móveis.

Seguidamente definem-se diversos cenários de utilização da aplicação, para uma melhor compreensão do seu fluxo.

6.1.1 - Inicialização da Aplicação

Na inicialização da aplicação, é sempre mostrado um ecrã enquanto se processam algumas operações de arranque (Figura 6). Para se processar o arranque da aplicação TellaiMobile, é então instanciada a classe TellaiMobileApp que extende a classe Application do sistema Android. Nesta classe inicializa-se o sistema de logs, de notificação de erros e declara-se dois receptores de broadcasts. O primeiro receptor de *broadcast*, escuta por alterações no ConnectivityManager do sistema Android, ou seja, alterações à conectividade da rede, ao passo que o segundo escuta por actualizações de dados da aplicação (alarmes, métricas, targets, etc). Estes receptores têm importante papel em todo o desenrolar da aplicação, pois permitem que se verifique internamente alterações ao estado da ligação de rede, necessária ao correcto funcionamento do TellaiMobile, e permite também notificar os ecrãs da aplicação de que existem novos valores, e que como tal devem actualizar-se. Esta funcionalidade é possível graças à classe BroadcastReceiver que permite registar *handlers* que visam tratar sinais enviados dentro do sistema.

Figura 6 - Ecrã de inicialização da aplicação

É ainda instanciado, na inicialização da aplicação, um serviço que corre em paralelo na aplicação, para actualizar os valores das métricas. Este serviço executa de 30 em 30 segundos, e invoca um WebService na aplicação TellaiMobile Web, que lhe devolve todos os valores das métricas actuais, actualizando-os na base de dados do dispositivo móvel. Aquando da recepção de novas métricas, envia então um sinal de sistema em *broadcast*,

sendo que caso exista algum *handler* à escuta (algum ecrã a mostrar métricas), força a actualização desse ecrã.

6.1.2 – Login

Para ser efectuado o login, é em primeiro lugar registado o *handler* que verificar o estado da ligação à rede, uma vez que é um ponto fundamental, visto que sem ligação não haverá login. Seguidamente é obtida a GCM Key, elemento fundamental para o funcionamento da aplicação, sendo esta o elemento que identifica o dispositivo em todo o sistema Tellai, e também a chave que permite todas as comunicações do sistema Tellai para o dispositivo com a aplicação, através da GCM.

Após o registo do *handler* e da obtenção da chave, inicia-se o processo de autenticação. Assim, é verificado em primeiro lugar se já foi alguma vez efectuada autenticação com sucesso no dispositivo, sendo que em caso afirmativo se utiliza essas credenciais para autenticar, pedindo no entanto as credenciais, no caso desta condição não se verificar.

Figura 7 - Ecrã de login da aplicação

Na acção de autenticação, é então invocado um *WebService* na aplicação TellaiMobileWeb que verifica se o cliente é válido e se o utilizador introduziu correctamente as credenciais. Caso alguma destas verificações não seja satisfeita, o utilizador é notificado, pedindo novamente as credenciais (Figura 7). Já no caso de tudo estar correcto, são arquivadas as credenciais na base de dados para posteriores autenticações de forma automática. Após a autenticação com sucesso, o utilizador é direcionado para o ecrã de entrada da aplicação e são actualizados os alarmes e restantes parâmetros de configuração, como targets, monitors e métricas (Figura8).

Figura 8 - Actualização dos alarmes após login

Importa ainda referir que, sempre que ocorre o registo de um novo dispositivo (autenticação de novo dispositivo), ou a alteração de utilizador no dispositivo, é colocada uma mensagem numa fila (RabbitMQ), onde é indicada a alteração. Esta mensagem é depois consumida pela aplicação nodeJS que actualiza a sua lista de dispositivos ligados.

6.1.3 – Ecrã de Entrada

No ecrã de entrada é mostrado um resumo do estado actual do cliente, sendo indicado o número de alarmes nos estados critical, major, warning e normal, o número de targets e o número de métricas (Figura9). Neste ecrã é possível ir para, as configurações da aplicação, os alarmes do cliente e os valores de métricas. Existem ainda registados neste ecrã 3 *handlers* que se encarregam de actualizar a interface gráfica, no caso de haver: actualizações de alarmes, actualizações de targets e actualizações de métricas.

Figura 9 - Ecrã de entrada da aplicação

6.1.4 – Ecrã de Configurações

No ecrã de configurações (Figura 10), é indicada a conta com que o utilizador se encontra ligado, possibilitando efectuar o *logout*. É ainda indicada a versão de software, é possível consultar os termos e condições e é possível definir se a aplicação notifica a ocorrência de novos alarmes no *status bar* do dispositivo.

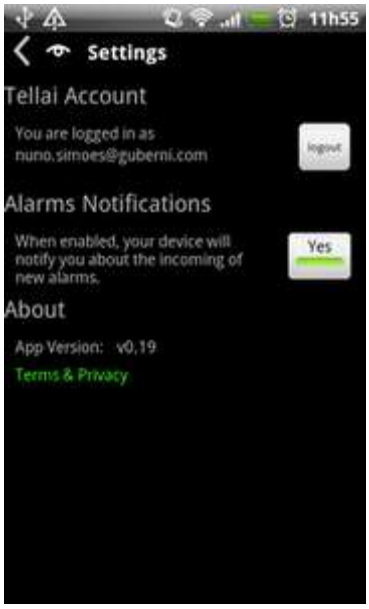

Figura 10 - Ecrã de configurações da aplicação

6.1.5 – Listagem de Alarmes

Na listagem dos alarmes podem ocorrer duas situações, ou não existem alarmes e é mostrada essa informação ao utilizador, ou existem e são listados todos os alarmes actualmente activos na infraestrutura (Figura 11). Em qualquer uma das situações o utilizador pode actualizar os alarmes por meio de um botão, quando já existem alarmes, ou tocando somente no ecrã, quando não existem alarmes. Não obstante, estes são sempre actualizados automaticamente aquando da ocorrência de novos alarmes, uma vez que são enviados em tempo real pela GCM.

Figura 11 - Listagem de alarmes na aplicação

Conforme dito anteriormente, sempre que surge um novo alarme na infraestrutura este é actualizado automaticamente no dispositivo. Para que tal ocorra, foi necessário construir um mecanismo que enviasse automaticamente as notificações para todos os dispositivos registados no cliente. Assim, existe na aplicação Tellai, um sistema denominado de AlarmManager, que tem como função analisar as métricas a entrar no sistema e de gerar alarmes sempre que necessário. Este mecanismo já existia, tendo sido alterado somente para colocar uma mensagem numa fila (Rabbit Queue) a indicar a ocorrência de um novo alarme, com o identificador deste. Essa mensagem é então consumida pela aplicação desenvolvida em nodeJS, que a envia para todos os dispositivos registados no cliente respectivo ao alarme. Após o envio, a mensagem é então recebida no dispositivo, o que inicializa o processo de inserir o alarme na base de dados, de enviar um sinal de sistema a indicar que há novos alarmes e colocar uma notificação na *status bar* do dispositivo, caso o utilizador tenha definido nas configurações da aplicação que pretende ser notificado.

No ecrã de listagem dos alarmes, é então mostrada a lista de todos os alarmes activos, conforme dito anteriormente, sendo que em cada entrada da lista é mostrado o nome do Target a que o alarme diz respeito, o objecto do alarme (e.g. uma partição do disco, um processo do sistema operativo, etc), o texto do alarme, a data de geração do alarme e o tempo em que já está activo.

Tocando no ecrã, numa entrada de alarme na listagem, são então mostrados os detalhes do alarme selecionado.

6.1.5 – Detalhes de Um Alarme

Nos detalhes de um alarme são mostradas várias informações, organizadas por separadores categorizados como: detalhes, acções, métricas, instruções e anotações. A navegação nestes separadores pode ser feita tocando no nome do separador desejado, ou arrastando os separadores de um lado para o outro (Figura 12).

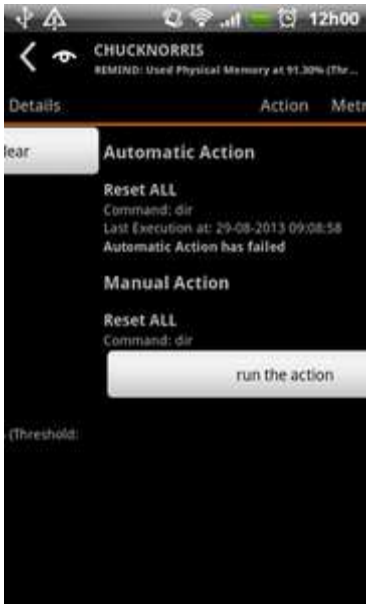

Figura 12 - Exemplo de navegação nos detalhes e um alarme

Detalhes

No ecrã de detalhes do alarme são mostrados o tipo e nome do target, o monitor e a métrica a que o alarme diz respeito, sendo mostrado ainda o texto do alarme (Figura 13).

Neste ecrã é ainda possível efectuar acções sobre o alarme como *own/disown*, em que o operador poderá marcar ou desmarcar esse alarme como seu, ou fazer *clear* ao alarme, indicando que a situação está resolvida.

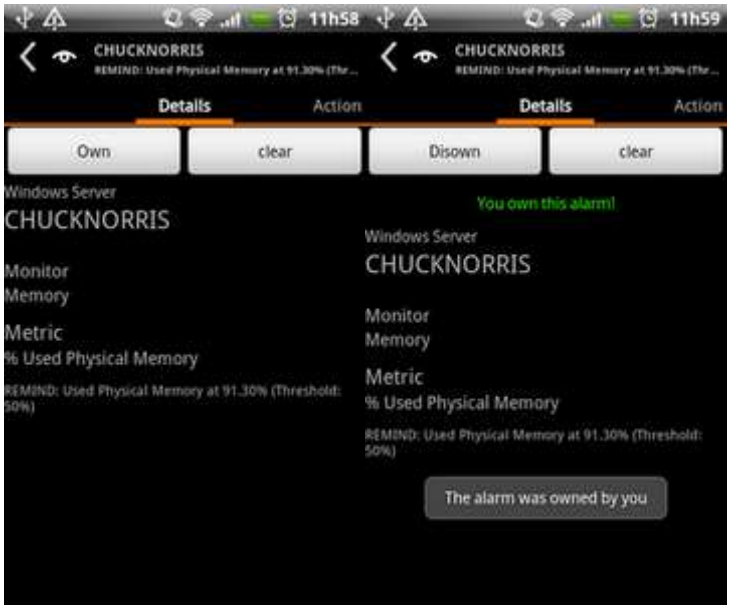

Figura 13 - Detalhes de um alarme e exemplo de operação sobre um alarme

Acções

Neste separador, são mostrados dois tipos de acções, caso estejam configuradas no alarme. No caso de nenhuma acção estar configurada, um ecrã é mostrado a indicar que não existem acções para executar.

Figura 14 - Ecrã de acções de um alarme

As acções podem então ser automáticas, caso esteja configurado para efectuar uma acção automaticamente, aquando da ocorrência do alarme, ou manuais, sendo que neste caso implica a execução por parte do operador por meio de um botão (Figura 14). Existem no entanto algumas verificações a efectuar, antes que o utilizador possa executar a acção. Em primeiro lugar, o utilizador tem de ter permissões para executar acções sobre alarmes. Passada esta primeira verificação, podem dar-se algumas situações em que o operador não

poderá executar acções manuais, mesmo que tenha permissões para isso, sendo elas: a acção já ter sido executada e estar configurada para não poder ser repetida (Figura 15), a acção ter falhado e estar configurada para não re-executar em caso de falha e o caso da acção ainda estar a executar (Figura 15). Em todas estas situações o utilizador não poderá executar acções manuais. Existe ainda a possibilidade de configurar a acção, para requerer um passo intermédio de confirmação da intenção de execução por parte do operador, antes de processar o pedido, sendo que nesse caso é mostrada uma mensagem de confirmação ao utilizador, podendo ele prosseguir ou cancelar.

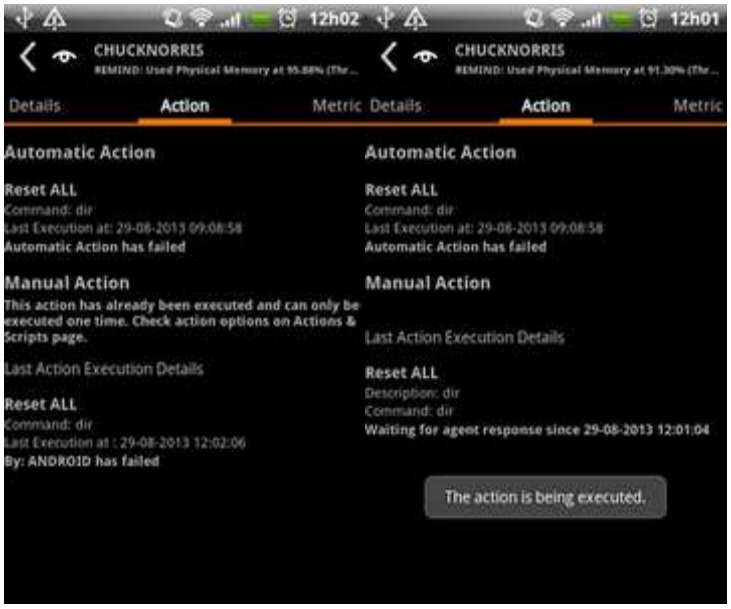

Figura 15 - Exemplo de acção executa e de acção a executar num alarme

Métricas

Para o utilizador poder ter uma melhor visualização da variação dos valores da métrica que gerou o alarme, é mostrado no separador métricas, um gráfico de linhas com os valores das últimas 4horas do par métrica/target correspondente (Figura 16).

Figura 16 - Gráfico com valores de métrica de um alarme

Instruções

Existe na aplicação Tellai a possibilidade de definir instruções num alarme, para leitura por parte do operador aquando da geração do alarme, sendo essas instruções visualizadas neste separador (Figura 17).

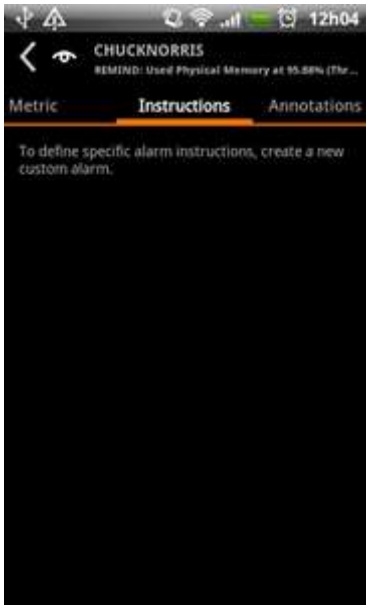

Figura 17 - Instruções de um alarme

Anotações

Podem ser adicionadas na aplicação Tellai, anotações aos alarmes, sendo que essa funcionalidade foi igualmente criada na aplicação TellaiMobile. Assim, no separador das anotações do alarme é possível introduzir novas notas no alarme e consultar as já existentes (Figura 18). Sempre que uma nova anotação é introduzida no dispositivo móvel, é invocado um WebService na aplicação TellaiMobile Web, para essa anotação ser introduzida na base de dados de alarmes (ClientDataStore) do cliente.

Figura 18 - Anotações de um alarme

6.1.7 – Valores de Métricas Filtrados por Target

Na segunda área da aplicação TellaiMobile, é possível visualizar os valores actuais das métricas.

Sendo que seria muito difícil visualizar todas as métricas sem qualquer filtragem, pela sua vasta quantidade, é mostrado ao utilizador um primeiro ecrã onde são listados todos os tipos de target (caso exista mais do que um) e após seleccionado o tipo de target, são listados todos os targets desse tipo seleccionado (caso exista mais do que um target para o tipo seleccionado) (Figura 19).

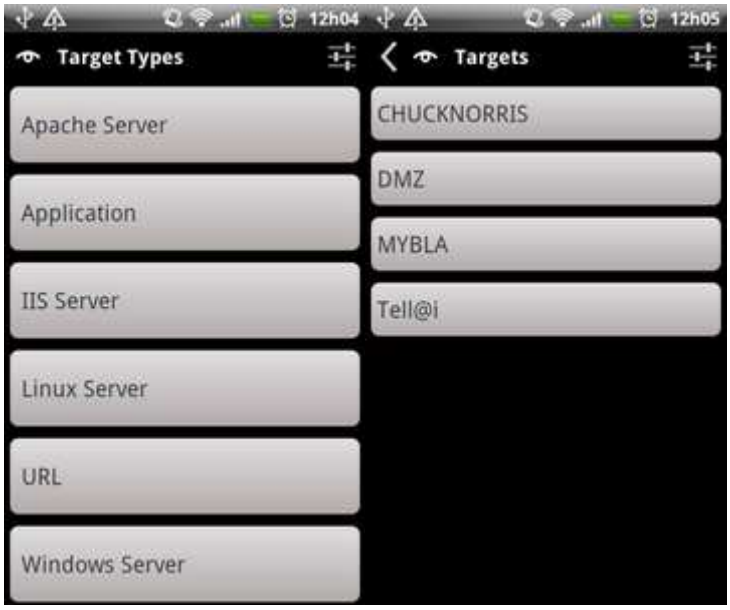

Figura 19 - Exemplo de navegação com tipos de target e targets

Após escolhido o target, são mostrados todos os valores actuais das métricas, categorizadas por monitor, podendo igualmente agrupar-se por localização da recolha da métrica, quando as métricas são *multi-location*, como o caso das métricas de HTTP, que podem ser recolhidas para o mesmo target de vários locais (Figura 20).

Figura 20 - Valores de métricas agrupadas por target

Esta parte da aplicação tem uma particularidade interessante em termos de desenvolvimento. Uma vez que os valores das métricas estão em constante actualização, implica refrescamentos constantes ao ecrã, o que se torna muito complicado de gerir na visualização por separadores, visto que cada actualização implica um redesenhar do ecrã e automaticamente voltar ao topo do ecrã. Como é normal o utilizador fazer *scroll* no ecrã para poder ver mais valores, voltar ao início sempre que o ecrã refrescava tornava-se incomportável. Assim, foi construída uma estrutura para guardar as coordenadas de *scroll* de cada separador, sendo que sempre que se processa um redesenhar do ecrã dos valores das métricas, verifica-se qual o separador actualmente visível e, após o redesenhar, força-se um *scroll* para o ponto em que o utilizador estava anteriormente.

6.1.8 – Valores de Métricas Fitlrados por Métrica

A visualização de valores de métricas, filtrado por métrica, é em todo igual à visualização de valores de métricas filtradas por target, diferindo na filtragem e na forma de agrupar os valores. Assim, ao utilizador é primeiramente disponibilizada a listagem de Monitors (caso exista mais do que um), a listagem de métricas associadas ao monitor escolhido (se houver mais do que uma no monitor escolhido), e após isso, os valores da métrica seleccionada, agrupados por target (Figura 21).

Figura 21 – Navegação por métrica e valores de métricas por métrica

6.1.8 – Valores da Métrica nas Últimas 4 Horas

Em todas as visualizações de métricas, é possível visualizar os valores das recolhas das últimas 4 horas para o par target/métrica correspondente, ao pressionar no valor actual da métrica (Figura 22).

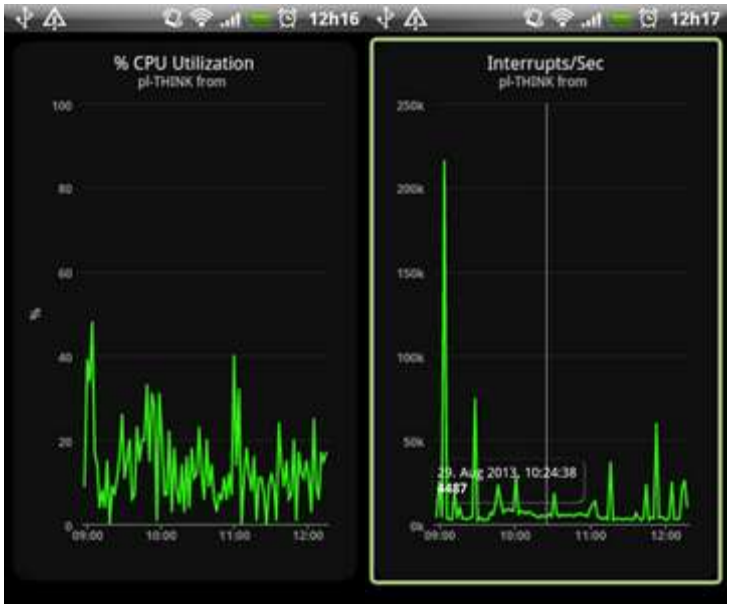

Figura 22 – Visualização dos valores de uma métrica nas últimas 4 horas e detalhes de um ponto no gráfico

6.2 - Requisitos de Software

6.2.1 – Ecrã de login (Ecrã 1)

O ecrã de login será somente visível caso o utilizador não tenha as credenciais já associadas à aplicação móvel. Caso isso se verifique, este será o primeira ecrã da aplicação, sendo aqui pedido ao utilizador que introduza as suas username e password, efectuando o login pressionando um botão.

Para uma representação visual, consultar o Mockup 1 do "Anexo C – Ecrãs do Software".

6.2.2 – Ecrã de loading (Ecrã 2)

Este ecrã somente servirá para dar um feedback visual ao utilizador, quando a aplicação necessita de processar algo que envolva algum tempo.

6.2.3 – Ecrã de Alarmes (Ecrã 3)

No ecrã de alarmes terá de ser possível visualizar o número de alarmes actualmente activos e somente para targets que o utilizador tenha permissões. Neste ecrã, os alarmes deverão ser agrupados por target, sendo mostrado o número do somatório de alarmes desse target. A severidade associada ao target, e que será mostrada será sempre, a pior das severidades, ou seja a mais gravosa, na escala de mais gravosa para menos gravosa respectivamente: critical, major e warning.

Caso o utilizador se encontre offline, deve dar essa indicação ao utilizador, bem como a data da última actualização dos dados.

Para uma representação visual, consultar o Mockup 2 do "Anexo C – Ecrãs do Software".

6.2.4 – Ecrã de listagem de Alarmes de um Target (Ecrã 4)

No ecrã de detalhes de listagem de alarmes de um target, devem ser listados todos os alarmes, do mais recente para o menos recente, sendo possível fazer as seguintes acções sobre os alames:

- ***Acknowledge*:** Utilizador indica que tomou conhecimento do alarme ficando atribuído a si, sendo então removido da listagem de alarmes activos;
- ***Own*:** Utilizador marca esse alarme como seu;
- ***Disown*:** Utilizador desmarca o alarme de si, caso este esteja associado a si;
- ***Clear*:** O alarme é removido da listagem de alarmes activos.

Estas acções sobre alarmes somente são possíveis se a aplicação estiver em modo online.

Para uma representação visual, consultar o Mockup 3 e Mockup 4 do "Anexo C – Ecrãs do Software".

6.2.5 – Ecrã de Métricas (Ecrã 5)

No ecrã de métricas deverá ser possível configurar, através do ecrã 6, quais as métricas e quais os targets que serão mostrados neste ecrã (ecrã 5).

Após configurado, mostra as métricas actuais para a configuração escolhida.

Caso o utilizador se encontre offline, deve dar essa indicação ao utilizador, bem como a data da última actualização dos dados.

Para uma representação visual, consultar o Mockup 5 do "Anexo C – Ecrãs do Software".

6.2.6 – Ecrã de Configuração do Ecrã das Métricas (Ecrã 6)

Neste ecrã devem ser mostradas todas as métricas disponíveis e targets, por forma a o utilizador poder escolher quais são mostradas no ecrã 5.

6.2.7 – Ecrã de Gráfico do par Métrica/Target (Ecrã 7)

Após carregar numa das amostragem do par Métrica/Target no ecrã 5, deve ser mostrado este ecrã, onde será mostrado um gráfico de linhas que mostre as ocorrências deste par num dado hiato de tempo.

Para uma representação visual, consultar o Mockup 6 do "Anexo C – Ecrãs do Software".

6.2.8 – Ecrã de Targets (Ecrã 8)

Deve ser possível visualizar uma listagem dos targets a que o utilizador tem acesso, agrupados por tipo. Deve ainda ser possível ver os detalhes de um target após seleccionar um dos targets, por meio do ecrã 9.

Caso o utilizador se encontre offline, deve dar essa indicação ao utilizador, bem como a data da última actualização dos dados.

Para uma representação visual, consultar o Mockup 7 do "Anexo C – Ecrãs do Software".

6.2.9 – Ecrã de Detalhes de um Target (Ecrã 9)

Neste ecrã deve ser possível visualizar os detalhes de um target.

Para uma representação visual, consultar o Mockup 8 do "Anexo C – Ecrãs do Software".

6.2.10 – Ecrã de Setup (Ecrã 10)

Neste ecrã poderá alterar-se a username e password que a aplicação utiliza para aceder aos dados. Deverá ainda ser possível definir se a aplicação recebe ou não notificações de novos alarmes em tempo real.

Para uma representação visual, consultar o Mockup 9 do "Anexo C – Ecrãs do Software".

6.2.11 – Armazenamento Local

A aplicação deve armazenar os dados localmente, por forma a poderem ser consultados em modo offline e também para optimizar a experiência de utilização da aplicação. Estes dados deverão ser actualizados quando o dispositivo móvel se liga à internet.

6.2.12 – Actualização de Métricas e Alarmes na Aplicação

As métricas devem ser actualizadas em períodos de tempo pré-definidos quando o utilizador se encontra ligado à internet.

6.2.13 – Notificação de Alarmes no Dispositivo Móvel

Sempre que um novo alarme é gerado, o dispositivo móvel deve receber uma notificação, não na aplicação, mas por meio do sistema de notificações nativa do sistema operativo, caso estas notificações estejam activadas no ecrã de setup da aplicação (ecrã 10).

6.2.11 – Widget de Alarmes

Deve ainda ser desenvolvida uma widget que permita ao utilizador associar um determinado target, contabilizado nesta os alarmes activos nesse dado target.

Para uma representação visual, consultar o Mockup 10 do "Anexo C – Ecrãs do Software".

6.3 - Infra-estrutura

O projecto final resultou em algumas alterações na infra-estrutura da solução geral, sendo que foi criado um conjunto de serviços que servem para a comunicação entre as aplicações instaladas nos dispositivos móveis e a solução Tellai. Assim, e por questões de organização, escalabilidade e segurança, foi criado o **TellaiMobile Web** que permite aos dispositivos móveis fazer pedidos de dados e uma pequena aplicação em **nodeJS** que serve para enviar dados em tempo real aos dispositivos móveis com notificações activadas via GCM. Esse **TellaiMobile Web** por sua vez comunica, já dentro da rede da solução, com o **Configuration Services** a fim de poder utilizar os dados armazenados nas bases de dados da aplicação.

Na figura seguinte é possível ver a constituição final da infra-estrutura após integração do projecto "Aplicação para Dispositivos Móveis de uma Solução de Monitorização de IT".

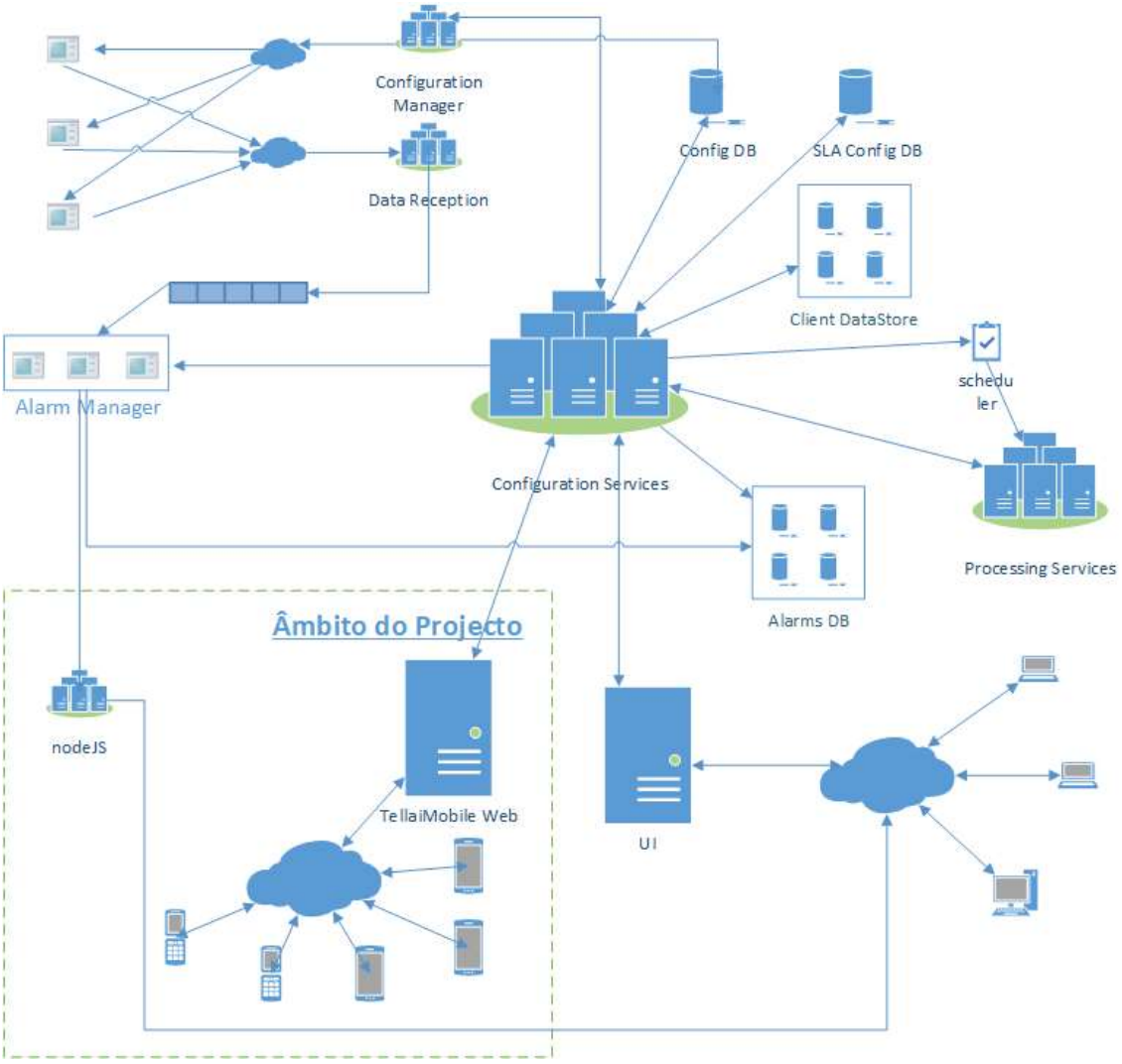

Figura 23 – Visão geral da infra-estrutura final

6.4 - Arquitectura de Software

A arquitectura da aplicação móvel interliga-se com algumas partes que já existiam na solução Tellai, mais concretamente **Configuration Services**, **ConfigDB**, **Client DataStore** e **AlarmsDB**.

No que concerne à arquitectura da Aplicação Móvel, temos os seguintes módulos:

- **TellaiMobile:** Diz respeito à aplicação móvel propriamente dita, que é instalada nos dispositivos móveis dos clientes. Esta aplicação móvel interage directamente com o módulo **TellaiMobile Web**, para fazer todos os pedidos de dados necessários ao seu correcto funcionamento. Esta aplicação tem ainda uma base de dados interna (SQLite), que funciona ao nível do sistema operativo do dispositivo.

- **TellaiMobile Web:** O módulo **TellaiMobile Web** funciona como intermediário entre os dispositivos móveis e os dados da aplicação. Por ele, passam os pedidos de autenticação e subscrição da aplicação, de alarmes, métricas e targets, bem como as acções sobre os alarmes.

- **Noitificações em Tempo Real:** Este módulo é constituído por 3 passos. No primeiro passo, o **Alarm Manager**, módulo já descrito anteriormente no capítulo "Estado da Arte" deste documento, envia para uma fila de mensagens, **Rabbit Queue**, os alarmes a serem enviados para os dispositivos móveis. Seguidamente, o servidor desenvolvido sobre nodeJS, carregando da **Config DB** a informação de que dispositivos estão registados e para quais devem ser enviados os alarmes, lê os alarmes existentes na fila e envia-os para o sistema de notificação GCM. O passo seguinte é o envio dessas mensagens por parte do GCM e interpretação dessas notificações por parte dos dispositivos móveis.

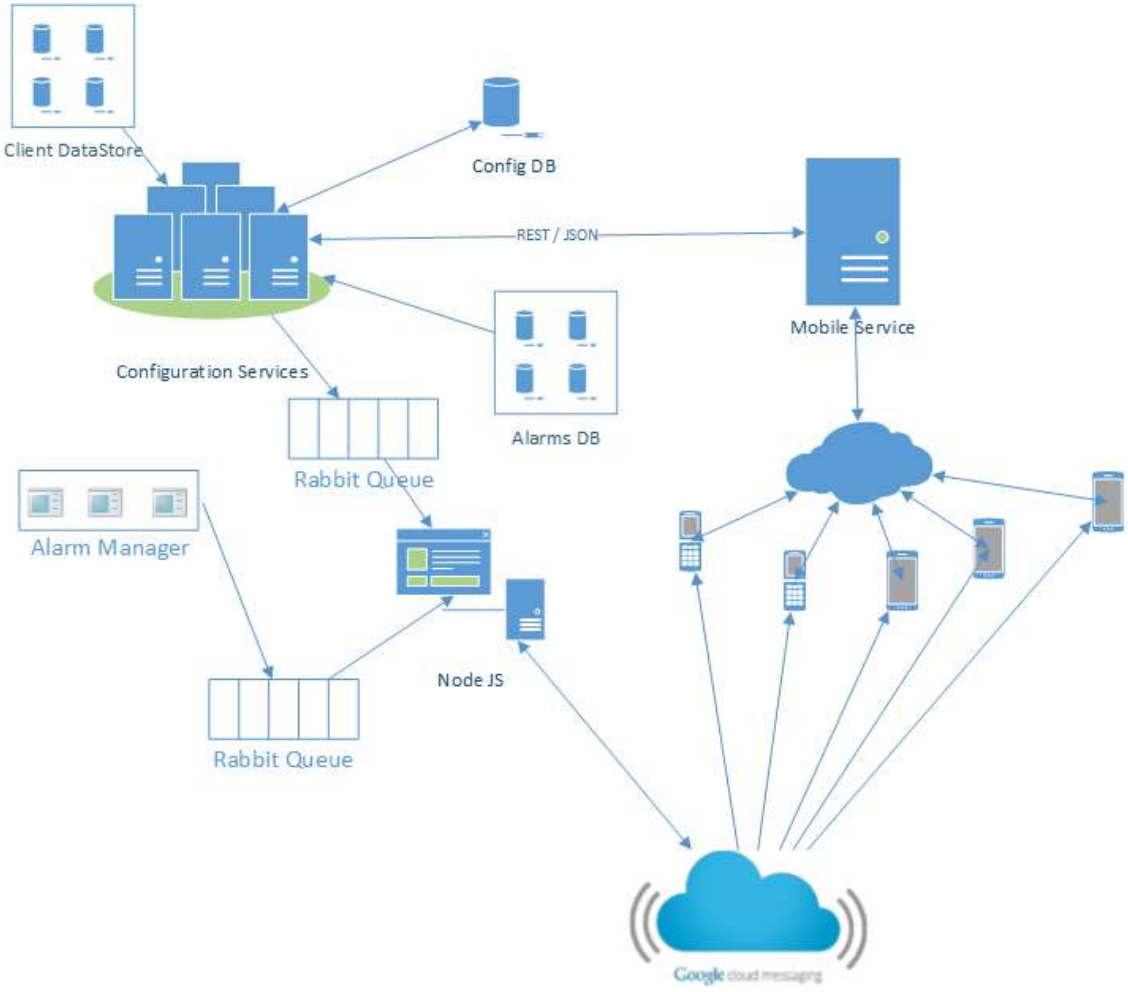

Figura 24 - Arquitectura da aplicação móvel

6.5 Organização de Código

TellaiMobile

A organização do código da aplicação TellaiMobile, foi toda direcionada, o máximo possível, para a reutilização nas futuras versões da aplicação em iOS e Windows Phone, uma vez que esse é um dos pontos fortes da utilização de Xamarin e que existe esse objectivo de desenvolver aplicação para esses sistemas operativos.

Assim, existem 5 projectos dentro da solução TellaiMobile:

- Mono.Android.Crasher:
- PushSharp.Client.MonoForAndroid.GCM
- TellaiMobile.Core.Android
- TellaiMobile.Logs
- TellaiMobileAndroid

Mono.Android.Crasher

Encontram-se no projecto Mono.Android.Crasher, todas as classes necessárias ao funcionamento do sistema de logging, que recolhe informação do dispositivo e envia por email, aquando de uma falha na aplicação.

Uma das dificuldades que surgiu aquando da fase de desenvolvimento foi o envio de relatórios de falhas. O Android tem um sistema embutido que permite o envio de relatórios de erros para a consola de desenvolvimento do projecto na Google, no entanto pretendia-se algo mais robusto, onde fosse possível receber num email diversas informações do estado do dispositivo aquando da falha. Não existindo nada nativo do sistema para essa função, efectuou-se alguma pesquisa até se encontrar uma livraria portada do ACRA para MonoAndroid disponível no GitHub, em https://github.com/soundnRg/Crasher, que permitia a recolha de informações do dispositivo, aquando de uma falha na aplicação, e enviava para uma folha de cálculo do Google. Este sistema não cumpria o requisito a 100%, mas era um bom ponto de partida, sendo que existia já nesta livraria um trabalho prévio para enviar o erro por email. Estando esta funcionalidade indicada como incompleta e não funcional, decidiu-se então tomar esta livraria como base e tentar colocar o envio por email a funcionar, o que se conseguiu ao fim de algumas horas de trabalho.

PushSharp.Client.MonoForAndroid

Para permitir o registo no Google Cloud Messaging e recepção de mensagens por parte do dispositivo, era necessária uma livraria que efectuasse esta gestão. Assim, optou-se por utilizar o PushSharp for MonoAndroid. Esta é uma versão portada pela Xamarin, do original PushSharp, para funcionar com o MonoAndroid.

TellaiMobile.Core.Android

Neste projecto reside uma boa parte da lógica da aplicação. É aqui que se faz a interação com a Base de Dados da aplicação TellaiMobile, com os *Web Services* da aplicação TellaiMobile Web e a declaração de todos os objectos necessários para mapear a base de dados da aplicação. Este projecto é como o nome indica, o coração da aplicação, sendo que está desenvolvido de forma a poder ser reutilizado para a versão de iOS e de Windows Phone facilmente, visto que está isolada do resto da aplicação em termos de dependências.

Este projecto divide-se então internamente em 4 partes:

1. **BL:** Contém todos os objectos que representam entidades da base de dados, e os managers, que são classes estáticas usadas para lógica de negócio associada às entidades.

2. **DAL:** Contém o DataManager, que é uma classe que organiza a interacção entre a lógica de negócio existente no **BL** e a Camada de Acesso a Dados existente no **DL** (descrito de seguida). Esta classe permite encadear diversas acções na base de dados, sem ter de as declarar repetidamente na camada de lógica de negócio, como por exemplo apagar registos antes de inserir novos, ou mesmo inserir registos em várias tabelas.

3. **DL:** Aqui encontram-se todas as classes que dizem respeito à camada de acesso a dados, sendo constituída por uma classe chamada SQLite, que é uma camada de persistência para SQLite desenvolvida pela Krueger Systems, Inc., e a classe TellaiMobileDatabase, que contém todos os métodos de CRUD necessários à interacção da aplicação com a base de dados SQLite, nativa do sistema android.

4. **SAL:** Com uma única classe, denominada TellaiMobileSiteParser, que é a classe responsável por todas as invocações a *WebServices* da solução TellaiMobile Web.

Na raíz deste projecto, existe ainda uma classe Constants, onde está centralizada toda a informação de URLs de WebServices, e demais parâmetros estáticos da aplicação.

TellaiMobile.Logs

Sendo necessária uma estrutura de logs para diversas operações da aplicação, criou-se este projecto para conter as estruturas necessárias. A razão da criação de um projeto aparte, prende-se somente pelo requisito de portabilidade.

Uma vez que praticamente todos os logs seriam escritos ao nível do projecto core da aplicação, e tendo em conta que este se manteve estanque quanto ao sistema (Android, iOS ou Windows Phone), tornou-se mandatório criar uma estrutura que pudesse ser usada na parte core, sem comprometer este nível de isolamento. Assim, existe neste uma interface denominada ITellaiLogFactory, que funciona basicamente como uma factory. Todas as invocações no projecto core são feitas sobre um objecto que extende esta interface, objecto esses que poderá ser o AndroidLogFactory ou o IosLogFactory, sendo que este último não está actualmente em uso, uma vez que não existe aplicação para iOS.

Tendo em conta esta abstracção, é da responsabilidade do projecto TellaiMobileAndroid (descrito seguidamente) invocar o objecto de Log como AndroidLogFactory. Este objecto é instanciado no arranque da aplicação, tendo um tempo de vida até à terminação da aplicação. Assim, sempre que é invocado um método do projecto core, passa-se por parâmetro a referência de memória do objecto AndroidLogFactory para ser usado. No sistema iOS irá funcionar da mesma forma, para o objecto IosLogFactory, respectivamente.

TellaiMobileAndroid

No projecto TellaiMobileAndroid reside toda a codificação que diz directamente respeito ao sistema Android, sendo maioritariamente a camada de visualização e alguma lógica associada a esta. Assim, este projecto divide-se em 5 áreas:

1. **Adapters:** Existem diversas situações, em que para popular estruturas complexas de visualização em listagem no sistema Android implica a utilização de um objecto de suporte que receba os dados a utilizar, e que os manipule para devolver cada uma das Views (objectos que representam estruturas da interface gráfica) que constituem essa estrutura global, já preenchida e pronta a mostrar no ecrã. Esses objectos denominam-se de Adapters e devem ser colocados na directoria adapters deste projecto.

2. **Fragments:** Os fragments poderão descrever-se como partes parciais de uma visualização de ecrã. No sistema Android é possível desenvolver-se ecrãs de visualização como um todo, ou como um conjunto de Fragments. Esta funcionalidade foi extremamente útil para desenvolver ecrãs em tabs, sendo que todos os fragments da aplicação residem nesta directoria.

3. **Resources:** Nas resources encontra-se, desde imagens utilizadas na aplicação, até aos ficheiros XML que definem os diversos layouts de ecrãs, passando por um conjuntos de ficheiros XML que definem valores fixos, como cores, textos e alguns outros estilos.

4. **Screens:** Na directoria screens, encontram-se todas as classes que extendem a class Activity do sistema Android. A class activity é semelhante aos fragments, tendo no entanto uma estrutura mais complexa, uma vez que são utilizadas para construir layouts completos.

5. **Support:** Contém diversos objectos criados para facilitar a lógica aplicacional, como por exemplo, verificação do estado de conectividade à rede, envio/recepção de sinais internos à aplicação, recepção de mensagens do GCM, etc.

Na raíz deste projecto existe ainda uma classe denominada TellaiMobile, que extende a classe Application do sistema Android, que serve para manipular diversos dados transversais a todos os ecrãs da aplicação.

TellaiMobile Web

A aplicação TellaiMobile Web, funciona como que um middleware entre toda a aplicação Tellai já existente e a aplicação TellaiMobile. Esta expõe diversos WebServices que são invocados pelos dispositivos móveis, para obter dados e efectuar operações sobre estes.

Para desenvolver estes WebServices, utilizou-se a tecnologia MVC4 da Microsoft, que se descreve pela estrutura seguidamente detalhada:

- **Controllers:** Os controllers são as classes que contém os métodos invocados directamente na chamada de cada URL. Estes métodos têm a lógica de negócio que é necessária para toda a execução invocada na chamada, sendo que normalmente funciona da seguinte forma: um controller, invoca um ou mais models para obter dados que depois são devolvidos para quem realizou a chamada ao controller. No caso dos gráficos, e uma vez que estes requerem a construção de uma view, estes dados são fornecidos à view que após construída, é devolvida a quem realizou a chamada ao controller.

- **DAL:** contém o objecto que representa cada cliente do sistema Tellai, mas de uma forma reduzida, tendo somente os parâmetros necessários ao funcionamento da aplicação móvel.

- **DL:** Ficheiros gerados pelo Entity Framework da Microsoft que funcionam como camada de persistência de dados da base de dados de configuração.

- **Models:** Os models são classes que permitem o acesso à camada de dados por parte dos controllers, sendo que no entanto, a grande generalidade dos Models não acedem directamente à camada de dados, mas sim a serviços expostos pela aplicação Tellai já existente. O resultado de cada chamada é então guardado numa cache aplicacional, que melhora substancialmente a performance da aplicação

TellaiMobile Web, ficando de fora desta cache todos os dados relativos a alarmes e métricas, uma vez que esta informação está em constante variação, não se justificando fazer cache destes.

- **Scripts:** Contém todo o código javascript utilizado pelas Views da solução TellaiMobile Web.
- **Views:** Nesta directoria encontra-se a view que permitirá visualizar gráficos na aplicação TellaiMobile.

Na solução TellaiMobile Web existe ainda um projecto de interfaces, com diversos objectos e enumeradores partilhados entre as aplicações TellaiMobile Web e TellaiMobile.

6.6 Testes de Usabilidade

Por forma a aferir o sucesso ou não do produto desenvolvido, em termos de usabilidade, efectuaram-se alguns testes com 20 voluntários, dos quais se apresentam os resultados seguidamente.
Para visualizar o teste em questão, consultar o Anexo D.

1 – Teve alguma dificuldade em instalar a aplicação no dispositivo?

Não	Sim
20	**0**

2 – Ao inicializar a aplicação, percebeu o que estava a acontecer?

Não	Sim
2	**18**

3 – O ecrã de entrada da aplicação era de percepção…

Muito complicada	Complicada	Razoável	Fácil	Muito fácil
0	**3**	**6**	**10**	**1**

4 – De uma forma geral, atingiu o caminho para o que pretendia ver?

Nunca	Poucas vezes	Algumas vezes	Muitas vezes	Sempre
0	**0**	**1**	**9**	**10**

5 – As entradas de alarme são de legibilidade…

Muito fraca	Fraca	Razoável	Boa	Muito boa
0	**0**	**6**	**13**	**1**

6 – Face à versão web, os detalhes dos alarmes são…

Nada familiares	Pouco familiares	Razoavelmente familiares	Bastante familiares	Totalmente familiares
0	**0**	**0**	**13**	**7**

7 – A consulta e execução das acções dos alarmes foi…

Nada perceptível	Pouco perceptível	Razoavelmente perceptível	Bastante perceptível	Totalmente perceptível
0	**1**	**7**	**1**	**11**

8 – Foi perceptível o porquê de não poder executar uma acção, quando isso ocorreu?

Nunca	Ao fim de algum tempo	Imediatamente
1	**5**	**14**

9 – Os gráficos de valores de métricas são de utilização e percepção…

Nada perceptível	Pouco perceptível	Razoavelmente perceptível	Bastante perceptível	Totalmente perceptível
0	**0**	**2**	**6**	**12**

10 – A leitura de anotações, já existentes, de alarmes foi perceptível?

Nunca	Ao fim de algum tempo	Imediatamente
0	**0**	**20**

11 – A introdução de novas anotações nos alarmes foi perceptível?

Nunca	Ao fim de algum tempo	Imediatamente
0	**0**	**20**

12 – O ecrã de configurações é acessível de forma…

Muito complicada	Complicada	Razoável	Simples	Muito simples
0	**0**	**2**	**4**	**14**

13 – A consulta de valores de métricas é...

Muito complicada	Complicada	Razoável	Simples	Muito simples
1	**3**	**8**	**4**	**4**

14 – Como foi a experiência de refrescamento dos valores das métricas?

Bloqueou constantemente	Bloqueou algumas vezes	Nem notei
0	**3**	**17**

Capítulo 7 - Implementação Prevista vs Implementação Realizada

Comparando aquilo que era os requisitos/objectivos iniciais, com o produto final, facilmente se percebe que houve mudanças várias. À primeira ideia, pode-se pensar que a introdução de 4 sprints se deveu a um mau planeamento, mas não. Após o inicio da implementação, e após as primeiras versões de demonstração, começou-se a ter uma percepção mais palpável do que era o software, levando assim a algumas mudanças que se enumeram, de forma abreviada, seguidamente:

- **Visualização de Métricas:** a princípio, o que se pretendia era poder consultar os últimos valores de cada métrica para cada target. Após o início do desenvolvimento, percebeu-se que em termos de usabilidade não iria funcionar, levando ao desenvolvimento de uma estrutura muito mais complexa, de filtragem e agrupamento visual, já descrito anteriormente no Capítulo 6, na secção, 6.1.7 e 6.1.8.

- **Visualização de Alarmes:** conforme se pode verificar no Anexo C, onde se encontram os mockups de ecrãs desenhados aquando do planeamento, a princípio pretendia-se visualizar em primeiro lugar o número de alarmes activos em cada target, permitindo depois ver a listagem de todos os alarmes activos desse dado target. Esse princípio foi abandonado, sendo que se preferiu mostrar todos os alarmes activos num só ecrã, por uma questão de familiarização com a funcionalidade na versão web. Após pressionar na entrada do alarme, permitiu-se ver todos os detalhes do alarme, ver o gráfico de ocorrências da métrica, consultar e executar acções, visualizar as instruções e consultar e adicionar anotações nos alarmes. Nenhuma destas funcionalidades estava prevista anteriormente, tendo sido implementadas todas.

- **Visualização de Targets:** esta funcionalidade foi abandonada, ficou no entanto pensada para uma versão futura da aplicação.

- **Widget:** a widget não se revelou uma mais-valia, face às novas necessidades implementadas, logo ficou pensada para uma versão futura da aplicação.

- **iOS:** com o tempo começou-se a perceber que para desenvolver uma aplicação realmente fiável, não seria possível desenvolver a aplicação para iOS nesta fase, sendo que irá ser o próximo passo no projecto.

Capítulo 8 - Conclusão

Com este estágio, foi-me possível concluir que a gestão de infra-estruturas IT, embora sendo uma área muito antiga, constitui ainda um grande potencial de exploração, quer pela sua abrangência, quer pela grande complexidade dos sistemas envolvidos.

Foi então um grande desafio participar deste projecto. Desde a aprendizagem de diversas terminologias e metodologias inerentes à gestão de IT, até ao conhecimento da solução Tellai, tudo foi novo e promoveu uma nova etapa da minha formação na área da informática.

Talvez então dos maiores riscos no processo fosse mesmo assimilar todo este novo conhecimento, principalmente a compreensão da solução existente, uma vez que falhas no conhecimento do funcionamento da solução poderiam ter custos muito elevados na fase de desenvolvimento, podendo mesmo invalidar todo o projecto. Esta fase foi então ultrapassada, podendo avançar com confiança e diligência para a fase de desenvolvimento, a fim de congregar o conhecimento apreendido durante o curso de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra com todo o conhecimento proveniente deste contacto no meio empresarial.

No que concerne ao processo de desenvolvimento, deu para perceber como é o ritmo do desenvolvimento de um projecto a nível empresarial, e de como os requisitos podem mudar rapidamente. Este projecto foi óptimo para tomar alguma experiência sobre como lidar com essa mudança de necessidades e de perceber como se processa a integração de novos softwares em software já existente. Este foi de facto um grande desafio, que me agradou imenso e me deu enorme regozijo.

Em jeito de agradecimento e finalização deste documento não poderia deixar de agradecer à guberni pela disponibilidade e boa vontade na fomentação da minha aprendizagem, deixando um especial reconhecimento ao Eng. Tiago Jorge e Eng. Pedro Laranjo. Agradeço também à minha namorada, demais família e amigos. O meu muito obrigado.

Referências

- Aplicações Móveis: http://en.wikipedia.org/wiki/Mobile_app e
- http://en.wikipedia.org/wiki/Mobile_application_development
- APNS: http://developer.apple.com/library/mac/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html e

  https://github.com/neoziro/node-apns
- Cloud Computing: http://en.wikipedia.org/wiki/Cloud_computing
- Embeed Databases: http://en.wikipedia.org/wiki/Embedded_database
- GCM: http://android.amolgupta.in/2012/07/google-cloud-messaging-gcm-tutorial.html,

  http://androidmyway.wordpress.com/2012/07/09/gcm-demo/,

  https://gist.github.com/3079447 e http://developer.android.com/google/gcm/gs.html
- Gestão de IT: http://en.wikipedia.org/wiki/Information_technology_management
- MySQL vs. PostgreSQL: http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL
- Mobile Frameworks: http://operationproject.com/2011/adventures-in-html5-part-one/#.UQWwm7-EyLR
- MongoDB: http://www.mongodb.org/
- MPNS: https://github.com/jeffwilcox/mpns
- .NET: http://blog.xamarin.com/eight-reasons-c-sharp-is-the-best-language-for-mobile-development/.
- Nodejs: http://nodejs.org/
- PhoneGap: http://phonegap.com/
- PushSharp: http://www.youtube.com/watch?v=MytQ6vqrE5g
- RabbitMQ: http://www.rabbitmq.com/
- SaaS: http://en.wikipedia.org/wiki/Software_as_a_service,

  http://www.computerworld.com.pt/2009/02/25/aplicaes-saas-fazem-valer-baixos-custos-de-implementao/ e

  http://b2bmagazine.consumidormoderno.uol.com.br/dicas/aplicac-es-empresariais-devem-crescer-4-5-em-2012

- Scrum: http://pt.wikipedia.org/wiki/Scrum
- SQLite: http://pt.wikipedia.org/wiki/SQLite
- TPC-H: http://www.tpc.org/tpch/
- Xamarin: http://xamarin.com/
- XP: http://pt.wikipedia.org/wiki/Scrum