2014

Instituto Politécnico de Coimbra

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

# Crowd Sensing para Campanhas de Solidariedade

MESTRADO EM INFORMÁTICA E SISTEMAS

AUTOR | David João Almeida dos Santos e Silva

ORIENTADORA |
Prof.ª Doutora Ana Cristina da Costa Oliveira Alves

Coimbra, dezembro 2014


Departamento
de Engenharia Informática

# *Crowd Sensing* para Campanhas de Solidariedade

Projeto apresentado para a obtenção do grau de Mestre em Informática e Sistemas

**Autor**
**David Silva**

**Orientadora**
**Ana Alves**
ISEC

**Coimbra, dezembro de 2014**


Departamento
de Engenharia Informática

# *Crowd Sensing* para Campanhas de Solidariedade

Projeto apresentado para a obtenção do grau de Mestre em Informática e Sistemas

**Autor**
**David Silva**

**Orientador**
**Ana Alves**
ISEC

**Coimbra, dezembro de 2014**

# Agradecimentos

Existem trabalhos simples, feitos por uma única pessoa, usando unicamente os seus conhecimentos adquiridos ao longo do tempo e com esforço pessoal, mas existem também aqueles trabalhos mais complexos, que mesmo que sejam feitos por uma pessoa, terão sempre apoio de muitas outras pessoas, as quais nem sempre têm a mesma visibilidade.

Este projeto é um bom exemplo não de um, mas de vários trabalhos executados ao longo de vários meses, sempre com o apoio de várias pessoas a quem estou bastante grato e às quais pretendo agradecer e realçar para que as mesmas não fiquem esquecidas. É também a última etapa para terminar a minha formação no Mestrado de Informática e Sistemas, no Instituto Superior de Engenharia de Coimbra, ao qual estou automaticamente grato pela formação que me têm dado ao longo destes últimos anos, quer em Mestrado, quer em Licenciatura.

Objetivamente queria agradecer à minha orientadora, professora Doutora Ana Cristina Oliveira Alves, pelo seu empenho, disponibilidade e interesse em levar este projeto sempre para a frente.

O meu também profundo agradecimento à Cáritas Diocesana de Coimbra pela sua disponibilidade, empenho e interesse em utilizar este projeto como uma ferramenta resultante da sua própria tentativa de modernização tecnológica, com especial agradecimento a Ana Paula Cordeiro, Carina Dantas, Carlos Ribeiro e Pedro Balhau como representantes da mesma instituição e pelo constante *feedback* dado.

Por último, mas não minorando, queria agradecer à minha família e à minha namorada pelo apoio que me têm dado, quer nos bons, quer nos maus momentos.

## Resumo

Este projeto industrial teve duas vertentes distintas, uma referente a um estudo tecnológico acerca da aplicabilidade de tecnologias de alto desempenho e alta disponibilidade em *Web Services* e outra direcionada a uma aplicação prática destas tecnologias para ajudas humanitárias.

Na fase de estudo tecnológico, foram analisadas várias frameworks para construção de *Web Services*, diferentes tipos de base de dados e diferentes tipos de bibliotecas para auxiliar na obtenção de códigos de produtos (EAN), análise esta que inclui um estudo de desempenho, disponibilidade e confiança, bem como apreciações sobre cada uma. A prioridade foi sempre recorrer a tecnologias open-source por forma a evitar problemas de licenciamento, permitindo que estes resultados possam ser aproveitados por alguém interessado, minimizando custos e transmitindo mais transparência.

A fase de aplicação prática tinha como visão a criação de uma plataforma facilmente adaptável, que incorpore o conceito de *Mobile Crowd Sensing* com vista a facilitar e centralizar o processo de inventariação, em tempo real e de fácil consulta, por forma a incentivar as pessoas a contribuir de forma equilibrada com o que é realmente mais prioritário. Isto deu lugar a uma aplicação cliente-servidor com um conjunto de funcionalidades mais afinadas aos objetivos da instituição Cáritas Diocesana de Coimbra, que decidiu acolher este projeto. Para conseguir este fim, estas aplicações recorrem a tecnologias recentes, tais como Android, *Web Services* e bases de dados não relacionais.

Este projeto foi desenvolvido segundo uma metodologia ágil: na definição de requisitos, na arquitetura, no desenvolvimento e testes, sempre apresentando os resultados à medida que eram obtidos como forma de receber feedback e permitindo encaminhar o projeto na direção pretendida.

A nível de experiência adquirida, este projeto permitiu conhecer o estado atual da tecnologia, dos seus prós e contras, uma mais-valia para decisões em aplicações futuras. Além disso, aumentou igualmente o leque de ideias para inovação na própria Cáritas, abrindo portas a novas aplicações integradas e de implementação de projetos de outras áreas.

**Palavras-chave:** Mobile Crowd Sensing, Android, NoSql, aplicações móveis, campanhas solidárias

# Abstract

This project has two separate strands, one refers to technological study about the applicability of high performance and high availability technologies in *Web Services* and the other is directed to a practical application of these technologies to solidary campaigns for goods collecting.

In this technological study, several frameworks for building *Web Services*, databases of different types and libraries to assist in obtaining product codes (EAN) and data were analysed, this includes a study of performance, availability and reliability, as well as appraisals for each one. The priority was always to use open source technologies, in order to avoid licensing issues, allowing these results to be used by anyone interested, minimizing costs and relaying more transparency.

The practical implementation phase's main goal was to create an easily adaptable platform that incorporates the concept of Mobile Crowd Sensing to ease and centralize the inventory's process, in real time and in an easy way to use, in order to encourage people to contribute in balanced portions, with what is really of a higher priority. This led to an application client-server with a collection of functionalities refined in accordance with the objectives established by the institution Cáritas Diocesana de Coimbra, the host of this project. To achieve this goal, these applications use the latest technologies such as Android, Web Services and non-relational databases.

This project was developed following an agile methodology, with cyclical definition phase of requirements, architecture design, development, testing and presentation of release's result, as a way to get feedback and allow a more efficient development of the project.

About the experience acquired, this project provided a state of art of technologies, with hers pros and cons, a very important thing to make decisions on future applications. Moreover, it also increased the range of ideas for innovation in Cáritas itself, opening doors to new integrated applications and implementations of other projects in some more areas.

**Keywords:** Mobile Crowd Sensing, Android, NoSQL, mobile applications, solidarity campaigns.

# Índice

# Índice de Figuras

# Índice de Tabelas

# Abreviaturas

ADT – Android Development Tools

API – Application Programming Interface

ART – Android Runtime

CORBA – Common Object Request Broker Architecture

CRUD – Create Read Update Delete

CSS – Cascading Style Sheets

CSV – Comma-separated Values

DBMS – DataBase Management System

DEIS – Departamento de Engenharia Informática e Sistemas

EAN – International Article Number (originalmente European Article Number)

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IDC – International Data Corporation

IDE – Integrated Development Environment

IP – Internet Protocol

ISEC – Instituto Superior de Engenharia de Coimbra

J2EE – Java Platform Enterprise Edition

JDK – Java Development Kit

JRE – Java Runtime Environment

JSON – JavaScript Object Notation

JVM – Java Virtual Machine

MVC – Model View Controller

MVP – Model View Presenter

MVVM – Model View ViewModel

NDK – Native Development Kit

NoSQL – Not Only SQL

RAM – Random Access Memory

REST – Representational State Transfer

RPC – Remote Procedure Call

SDK – Software Development Kit

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

SVN – Apache SubVersion

TCP – Transmission Control Protocol

UI – User Interface

UTF – Unicode Transformation Format

UX – User eXperience

WORA – Write Once Run Anywhere

WSDL – Web Services Description Language

XML – eXtensible Markup Language

# 1. Introdução

Em pleno século XXI, a fome e a escassez de bens essenciais são males que afetam uma grande parte da população mundial. Embora normalmente estejam associados ao subdesenvolvimento, estes atingem também bastantes pessoas nos países desenvolvidos. Felizmente existe quem, pelos mais variados motivos, esteja disposto a ajudar.

Na tentativa de reduzir estes e outros males, criaram-se grupos e instituições de apoio coletivo e organizado, em que uma das tarefas é a angariação voluntária de bens alimentares e outros géneros de bens de primeira necessidade.

Apesar destas angariações serem de extrema ajuda, as mesmas levantam 2 problemas:

- Discrepância entre necessidade e oferta – Embora toda a ajuda seja bem-vinda, existem bens mais prioritários, prioridades essas que podem mudar em função de angariações anteriores, altura do ano, fenómenos naturais (p. ex.: cheias, fogos, terramotos, …) ou até mesmo em função da recolha atual uma vez que os temas das campanhas de um determinado tipo (p. ex. alimentar, higiene, material escolar) não são específicas a um bem. Neste último caso, é vantajoso para o voluntário (tanto para quem doa como para quem está a recolher) saber em tempo real que categorias específicas de bens já foram doadas em que quantidade (p. ex. cadernos, lápis, canetas).
- Heterogeneidade geográfica – Havendo pessoas a doar bens em todo o país, e tendo em conta a dispersão populacional, é de imaginar que esta doação não seja homogénea. Mesmo a nível regional existe uma discrepância muito grande em termos de quantidade e tipo de doações, algo que se for possível estimar, permite uma melhor otimização de recursos humanos e logísticos.

Tendo em mente estes 2 problemas, surgiu a ideia de uma abordagem *Crowd Sensing* unido às tecnologias móveis [1], para investigar e ajudar a entender os padrões e distribuição de doações, ao mesmo tempo permitindo ter uma ideia em tempo real da discrepância entre a necessidade e o total de bens doados, por forma a sensibilizar as pessoas para ajudarem com o que está em falta.

A abordagem feita vai ao encontro do conceito de código aberto (*Open Source*) e tecnologias móveis, por forma a ser aplicável em qualquer parte sem limitação de licenças e poder ser melhorada por outros interessados.

## 1.1. Objetivos do Projeto

Dados os problemas realçados, as objetivos primários que este projeto pretende colmatar são:

- Desenvolver um serviço para agregação de informação para inventariação em tempo real de bens recolhidos;
- Criar uma aplicação para plataforma Android que permita a catalogação e visualizar a informação recolhida em tempo real;

- Afinação da solução para uma instituição de solidariedade nacional, neste caso a Cáritas Diocesana de Coimbra.

Além destes, também se definiu os seguintes objetivos secundários que pretendem também ser uma mais-valia para a globalidade do projeto em si e para trabalhos futuros:

- Pesquisar, testar, avaliar e eventualmente melhorar tecnologias recentes nas áreas de:
  - Reconhecimento de códigos de barras em plataformas móveis;
  - Encapsulamento de informação e chamada remota a procedimentos através de protocolos de comunicação;
  - Criação de *Web Services* para recolha e agregação de informação;
  - Utilização de um modelo de base de dados mais otimizado para acessos concorrentes em tempo real para guardar a informação recolhida.
- Análise dos dados recolhidos de forma a apresentar a participação temporal/espacial da campanha, por forma a entender padrões e fazer um melhor planeamento de campanhas futuras, à imagem do que já foi feito em outros projetos de Mobile Crowd Sensing [2] [3] [4].

Com estes objetivos primários e secundários cumpridos, obter-se-á uma base sólida pronta para desenvolvimento e melhorias futuras, com outras instituições quer a nível nacional, quer a nível internacional.

## 1.2. Cáritas Diocesana de Coimbra

Tal como se apresenta em [5], a Cáritas Diocesana de Coimbra é uma Instituição Particular de Solidariedade Social que apoia de forma transversal as comunidades nos âmbitos social, saúde, educação e pastoral, em 5 distritos da região Centro de Portugal.

Implantada desde a década de 50, a Cáritas sempre procurou acompanhar e responder, subsidiariamente, aos problemas das comunidades, utilizando uma metodologia que privilegia o diálogo, a cooperação e o trabalho em rede. Atualmente foca a sua intervenção na procura de estratégias inovadoras e economicamente sustentáveis, que permitam a prestação de respostas com qualidade, adequadas às necessidades emergentes, mantendo como enfoque o humanismo, profissionalismo e o rigor técnico e científico.

Consciente da sua responsabilidade social a Cáritas tem, ao longo dos tempos, desenvolvido a sua ação de forma a privilegiar as pessoas, famílias e grupos social e economicamente mais carenciados contando, atualmente, com perto de 120 respostas sociais na área geográfica da Diocese de Coimbra.

- Abrange as seguintes áreas principais, entre outros projetos de caráter mais pontual:
- Respostas sociais:
  - Educação (Infância, Tempos Livres);
  - Saúde (Ambulatório, Internamento);

    - Ação Social, Família e Comunidade (Crianças e Jovens em risco, Idosos, VIH/sida, Toxicodependência, Sem-abrigo, Intervenção Comunitária).
- Serviços – Formação, Clínica, Lavandaria, Colónia de férias;
- Ação Pastoral.

A dimensão da ação Cáritas e a sua natureza são expressão da responsabilidade social que tem interiorizado ao longo das últimas cinco décadas e lhe permitem ser referencial dinamizador e transformador de toda a sociedade em prol do bem comum.

A Cáritas tem como Missão ser um instrumento da Igreja, na área geográfica da diocese de Coimbra, para promover e defender a dignidade humana à imagem de Jesus Cristo.

É Visão da Cáritas ser uma referência diocesana e nacional pela qualidade e capacidade de ser pioneira nos serviços que presta à comunidade de forma próxima, reflexiva e sustentável.

A Cáritas assenta a sua missão nos seguintes Valores essenciais:

- Humanização - A Cáritas, na defesa/promoção da dignidade humana, desenvolve uma intervenção centrada na pessoa e na comunidade, salvaguardando os respetivos "direitos, liberdades e garantias";
- Profissionalismo - A Cáritas, no trabalho que desenvolve, pauta-se eticamente pelo rigor técnico, competência e consistência;
- Compromisso - A Cáritas leva a cabo a sua missão com determinação, persistência, empreendedorismo, disponibilidade, entrega, entreajuda e lealdade;
- Transparência - A Cáritas projeta a sua intervenção a partir de uma leitura da realidade, de modo a que a mesma possa ser sinal visível da sua visão;
- Caridade - A Cáritas vincula a sua ação à dimensão do amor ao próximo, na assistência, promoção, desenvolvimento e transformação de estruturas, pelos profissionais e voluntários;
- Universalidade - A Cáritas acolhe todas as pessoas independentemente da nacionalidade, etnia, religião ou proveniência social e olha para todas as problemáticas como provocação à sua ação;
- Criatividade - A Cáritas faz face às múltiplas problemáticas existentes e emergentes, procurando inovar nas respostas com flexibilidade e transdisciplinaridade.

### 1.2.1. Estrutura orgânica da Cáritas Diocesana de Coimbra

A Cáritas Diocesana de Coimbra pode ser organicamente representada segundo o diagrama presente na Figura 1.

Os setores mais relevantes para este projeto foram sobretudo a Equipa Informática e de Inovação.

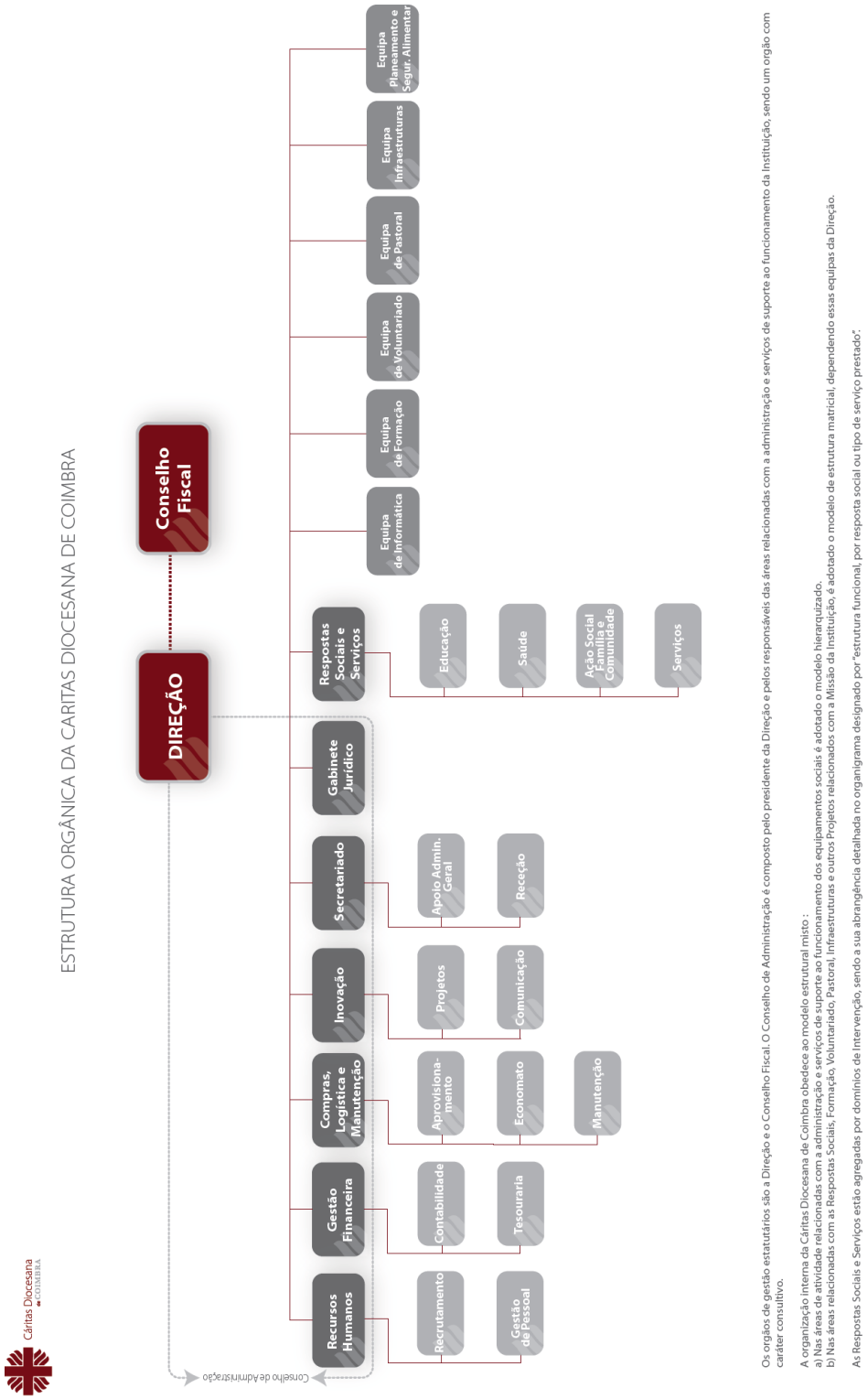



*Figura 1 - Estrutura orgânica da Caritas Diocesana de Coimbra*[1]

[1] http://www.caritas.pt/site/coimbra/images/pastapdf/institucional/organica.pdf [Últ. acesso a 2014-12-12]

### 1.2.2. Distribuição geográfica das respostas

Contrariamente ao que o nome possa refletir, a Cáritas Diocesana de Coimbra abrange não só o concelho de Coimbra, mas também os concelhos de Aveiro, Leiria, Santarém e Viseu.

A sua dispersão de centros de apoios pode ser vista no panfleto informativo da Figura 2.

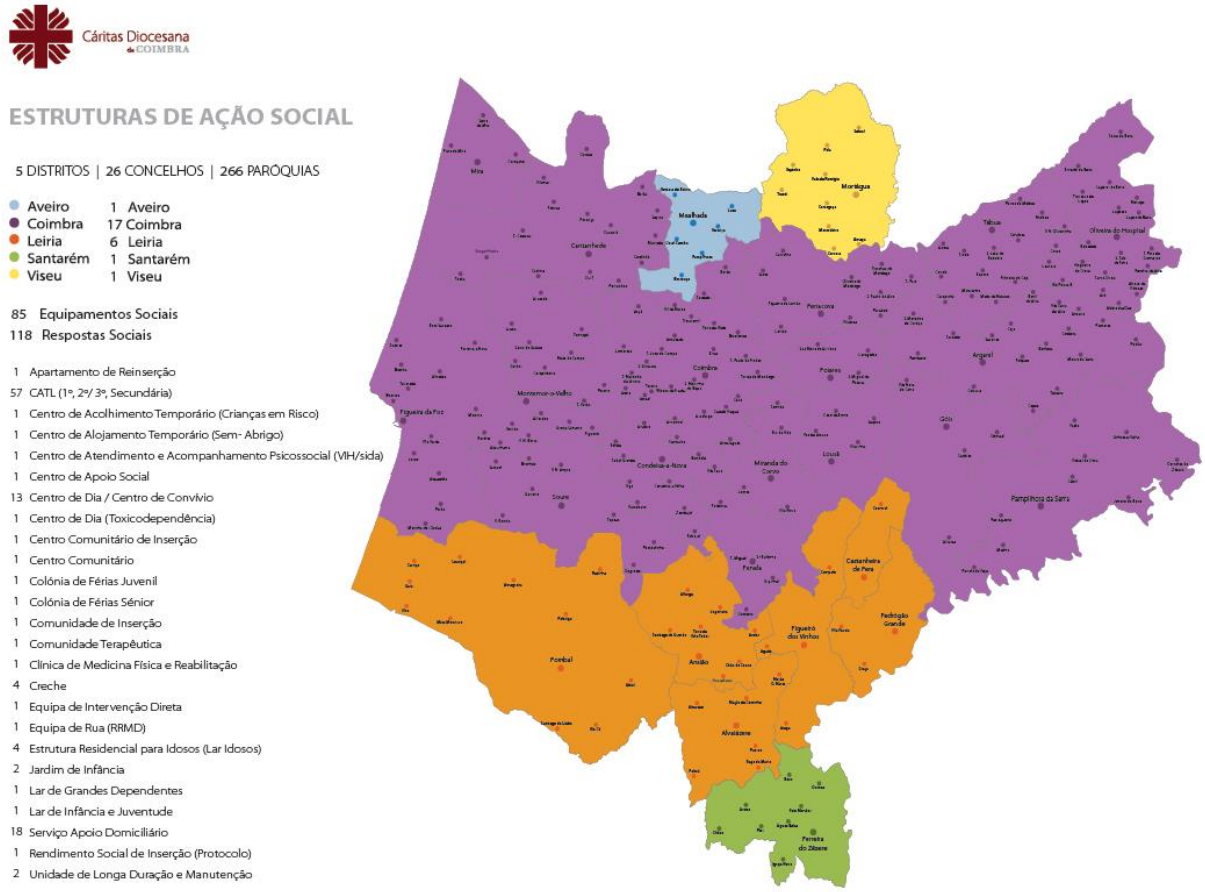



*Figura 2 - Mapa de abrangência da Caritas Diocesana de Coimbra*[2]

## 1.3. Calendarização

Associado à proposta inicial de projeto, havia uma lista de tarefas e as datas estimadas da sua conclusão. As tarefas inicialmente propostas foram as seguintes:

- T1 – Estudo e caracterização das tecnologias envolvidas

[2] http://www.caritas.pt/site/coimbra/images/pastapdf/2013/mapadiocese%20final-01.jpg [Último acesso a 2014-12-12]

- T2 – Análise de requisitos
- T3 – Desenvolvimento
- T4 – Testes (Arquitetura/Funcionais/Usabilidade)
- T5 – Refinamentos e correções
- T6 – Desenvolvimento da interface de visualização
- T7 – Teste em campo e Recolha de dados
- T8 – Análise espacial e temporal dos dados
- T9 – Documentação (manuais, artigo e dissertação)

A disposição das tarefas foi feita de acordo com os dados da Tabela 1.

| | Meses | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Tarefas** | | N | | N+1 | | N+2 | | N+3 | | N+4 | | N+5 | | N+6 | | N+7 | | N+8 | |
| T1 | | | | | | | | | | | | | | | | | | | |
| T2 | | | | | | | | | | | | | | | | | | | |
| T3 | | | | | | | | | | | | | | | | | | | |
| T4 | | | | | | | | | | | | | | | | | | | |
| T5 | | | | | | | | | | | | | | | | | | | |
| T6 | | | | | | | | | | | | | | | | | | | |
| T7 | | | | | | | | | | | | | | | | | | | |
| T8 | | | | | | | | | | | | | | | | | | | |
| T9 | | | | | | | | | | | | | | | | | | | |
| **Metas** | INI | | | M1 | M2 | | | | | | | M3 | | | | M4 | | | M5 |

*Tabela 1 - Calendarização de tarefas original*

Dado que se optou por uma metodologia ágil e tendo em conta que o começo tardio do projeto levou a que houvesse coincidência com datas de férias e consequentemente maior lentidão e menor indisponibilidade para receber feedback, foi necessário fazer um reajustamento do plano, ficando o plano de tarefas de acordo com a Tabela 2.

| | Meses | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Tarefas** | | N | | N+1 | | N+2 | | N+3 | | N+4 | | N+5 | | N+6 | | N+7 | | N+8 | N+7 | |
| T1 | | | | | | | | | | | | | | | | | | | | |
| T2 | | | | | | | | | | | | | | | | | | | | |
| T3 | | | | | | | | | | | | | | | | | | | | |
| T4 | | | | | | | | | | | | | | | | | | | | |
| T5 | | | | | | | | | | | | | | | | | | | | |
| T6 | | | | | | | | | | | | | | | | | | | | |
| T7 | | | | | | | | | | | | | | | | | | | | |
| T8 | | | | | | | | | | | | | | | | | | | | |
| T9 | | | | | | | | | | | | | | | | | | | | |
| **Metas** | INI | | | | M1 | | | | | | | | | | | | | | M4 | M5 |

*Tabela 2 - Calendarização de tarefas adaptada*

Em paralelo com as tarefas e em datas combinadas, foram feitas demonstrações do trabalho realizado, nomeadamente validação de documentos e demonstração da aplicação desenvolvida. Estas datas estão evidenciadas na Tabela 3.

| **Data demonstração** | **Elementos apresentados/discutidos** |
|---|---|
| 28/02/2014 | • Apresentação de proposta<br>• Análise de aspetos essenciais à aplicação a desenvolver |
| 18/04/2014 | • Apresentação de *mockup* funcional para ideias (*release* 1)<br>• Apresentação da arquitetura, organização e catalogação de dados |
| 03/06/2014 | • Enumeração de dados necessários para o projeto<br>• Debate acerca do método de tratamento de introdução manual e introdução automática de produtos |
| 24/07/2014 | • Demonstração do *release* 2<br>• Proposta de integração inserção agrupada de produtos<br>• Discussão quanto aos aspetos de User Experience, User Interface e *Guidelines* no geral |
| 31/07/2014 | • Demonstração de *mockups* não funcionais feitos de acordo com o proposto na reunião anterior |
| 01/09/2014 | • Demonstração do *release* 3 |
| 30/09/2014 | • Apresentação do *release* 4<br>• Apresentação da interface administrativa |
| 18/12/2014 | • Apresentação do release 5<br>• Teste em campo da aplicação nas instalações da Caritas |

*Tabela 3 - Datas das apresentações e apreciações das Releases*

## 1.4. Estrutura do Relatório

Este relatório está dividido em vários capítulos, cada um dedicado a um tema em concreto.

O primeiro capítulo faz uma apresentação do projeto, as razões que o levaram a ser executado e o que pretende alcançar, juntamente com uma apresentação da instituição de solidariedade que acolheu este projeto. Também está presente um mapa de calendarização original e real da evolução e demonstrações feitas.

O segundo capítulo introduz a problemática da crescente necessidade de alto desempenho e alta disponibilidade dos serviços baseados em web, bem como uma apresentação de frameworks e sistemas que prometem fazer face a esta necessidade.

O terceiro capítulo apresenta o conceito de aplicações móveis, dispositivos embebidos, o sistema Android e como a sua existência tem modificado a forma das pessoas usarem os seus *smartphones* e outros dispositivos que até hoje eram de uso muito restrito.

O quarto capítulo abrange toda a investigação realizada, descrevendo as comparações feitas entre frameworks de *Web Services*, protocolos de comunicação, bases de dados

e bibliotecas de reconhecimento de códigos EAN, juntamente com uma lista de pontos positivos e pontos negativos de cada uma.

O quinto capítulo inclui o desenvolvimento das aplicações em si, descrevendo as metodologias usadas, os requisitos, desenhos de arquiteturas e ciclos de desenvolvimento associados a cada *release*, ferramentas e linguagens usadas.

No sexto capítulo é feita a análise das metodologias aplicadas, do desenvolvimento das diferentes partes do projeto e de possível trabalho futuro.

O sétimo capítulo contém todas as referências do material consultado ou utilizado para a conceção deste trabalho, incluindo os estudos feitos e documentação auxiliar na fase de desenvolvimento.

# 2. Frameworks de Alto Desempenho/Disponibilidade

Desde o seu aparecimento na década de 80, a Internet veio abrir portas a um conjunto de informação muito mais vasto do que era possível aceder anteriormente. Inicialmente usado para fins militares e académicos, em pouco tempo expandiu-se de uma forma exponencial a um público abrangente. Hoje em pleno ano de 2014, já se estima que existam mais de 2,8 mil milhões de utilizadores em todo o mundo, o que representa quase 40% da população mundial[3].

Este aumento deve-se ao crescente interesse das pessoas de obterem e partilharem informação, o que tem levado a um aumento dos meios de acesso à mesma, tornando-se um ciclo vicioso. Há uma década atrás, era apenas possível aceder à internet através de linha telefónica ou linhas dedicadas num computador de secretária. Atualmente já é possível através de linhas ADSL, GSM, fibra ótica e outros meios de alta velocidade, não só através de dispositivos fixos, mas especialmente por meios móveis tais como computadores portáteis, telemóveis, *smartphones*, *tables*, consolas de jogos e mais recentemente dispositivos sensoriais que abrangem o novo e crescente conceito de IoT (*Internet-of-Things)*[4] [5].

Este aumento incessante de acessos e informação é uma realidade cada vez mais presente. A meio do ano 2014, a Google já contabilizava 100 mil milhões de pesquisas mensais[6], o Twitter cerca de 500 milhões de *tweets* diários[7] e a Cisco estimava que um total de 62 Exabytes ($10^{18}$) fossem transmitidos na Internet mensalmente, prevendo que este valor chegue a 132 Exabytes em 2018[8]. Como se isto não fosse bastante, é preciso garantir a disponibilidade contante desta informação, já que como muitas empresas e serviços estão apoiados na Internet, uma simples falha pode acarretar perdas gigantescas. Com valores desta magnitude, mesmo com o avanço do *hardware*, torna-se complicado servir tanta informação, em parte devido a limitações da arquitetura do hardware em si e também pela forma como o software é pensado.

Apesar de só recentemente este assunto estar a ganhar grande destaque, já há muito que é debatido. De facto já desde o ano 2000 que vários autores publicam artigos referentes a este problema do crescente aumento do número de utilizadores, referindo especificamente o problema C10K[9], e mais recentemente o problema C10M[10]. Estes problemas referem-se ao desafio de conseguir servir 10 mil e 10 milhões de utilizadores em simultâneo, algo que apesar de não ser óbvio à primeira vista, apresenta um conjunto de desafios tecnológicos.

[3] http://www.internetworldstats.com/stats.htm [Último acesso a 2014-12-12]

[4] http://whatis.techtarget.com/definition/Internet-of-Things [Último acesso a 2014-12-12]

[5] http://www.gartner.com/newsroom/id/2636073 [Último acesso a 2014-12-12]

[6] http://phandroid.com/2014/04/22/100-billion-google-searches/ [Último acesso a 2014-12-12]

[7] https://about.twitter.com/company [Último acesso a 2014-12-12]

[8] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html [Último acesso a 2014-12-12]

[9] http://www.kegel.com/c10k.html [Último acesso a 2014-12-12]

[10] http://highscalability.com/blog/2013/5/13/the-secret-to-10-million-concurrent-connections-the-kernel-i.html [Último acesso a 2014-12-12]

## 2.1. Estado da Tecnologia

Uma das formas para fazer face ao problema do crescente número de utilizadores e à necessidade de disponibilizar cada vez mais informação eficientemente, passa precisamente pelo modo como esta é processada e disponibilizada. É precisamente nesta área onde entram as frameworks de alto desempenho e alta disponibilidade, permitindo a construção de serviços que quando bem construídos e organizados, permitem atender um número de utilizadores e uma quantidade de informação até então impossíveis com os mesmos recursos. Dentro desta categoria entram os *Webservices*, sistemas de gestão de base de dados (DBMS) e quaisquer outros serviços associados.

O termo Webservice refere-se ao extrato de software encarregue de receber os pedidos de clientes remotos, executar alguma ação especifica e enviar alguma informação, tanto dados como indicadores de falha e sucesso. Dentro destes, as frameworks usadas para os implementar, usam uma arquitetura baseada em pools de *threads*, algo que apesar de ser uma otimização em relação à antiga criação de uma *thread* por cliente, continua a ser bastante dispendiosa a nível de recursos. Por muito que a tecnologia avance, os próprios acessos a *hardware* são sempre uma parte lenta do acesso à informação, sendo assim um dos principais *bottlenecks* no acesso aos dados. Para contrapor esta limitação, as frameworks de alto desempenho usam uma estratégia diferente, tirando proveito de uma arquitetura *event-driven* e uma API não-bloqueante, geralmente aliada ao padrão de *software* reativo. Na prática isto significa ter uma única *thread* que está ciclicamente à espera de novos eventos, podendo ser estes um novo pedido ou a indicação de novos dados requisitados, diminuindo assim os tempos mortos devido aos tempos de espera pelo *hardware*, diminuindo a memória usada e dividindo melhor o tempo de processador por cada cliente.

Outro ponto fundamental destas tecnologias é a sua capacidade de estar distribuída, seja por vários processadores, máquinas ou mesmo em diferentes localizações geográficas, permitindo ter vários servidores espalhados ao nível de uma *Wide Area Network* (WAN) como forma de redundância a falhas e estarem mais próximos dos pontos onde há maior afluência de tráfego, aumentando o *throughput* geral.

Embora se detalhe melhor nos capítulos seguintes, pode referir-se que atualmente a framework para criação de Webservices que se encontra mais em voga, devido às suas capacidades, é certamente o Node.js[11]. Mesmo tendo sido lançada inicialmente em 2009 e correndo em JavaScript, o facto de correr sobre o motor de JavaScript V8 da Google e cada vez mais *plugins* e drivers de acesso, faz com que tenha um uso crescente.

Além dos Webservices, outro elemento que tem recebido especial importância tem sido os DBMS distribuídos e de alta disponibilidade. Das quais, de destacar as bases de dados relacionais com replicação e as mais recentes bases de dados NoSQL, as quais serão devidamente debatidas no seguimento deste relatório.

De entre os vários DBMS, os que têm sido debatidos recentemente serão certamente o Cassandra e MongoDB.

[11] http://nodejs.org/ [Último acesso a 12 de Dezembro de 2014]

## 2.2. Exemplos de Utilização

Apesar de serem novas, algumas empresas já estão a tirar partido destas tecnologias mais recentes e a comprovar os seus benefícios. De seguida são detalhadas para as tecnologias anteriormente apresentadas alguns cenários de utilização nas empresas de software que as adotaram.

### 2.2.1. Node.js

Tratando-se de uma das tecnologias de alto desempenho mais faladas da atualidade, o Node.js destaca-se por ser uma framework feita em JavaScript, que corre sobre a Engine V8, a mesma usada pelo Google Chrome, o que consequentemente leva a que o código para Node.js seja escrito inteiramente em JavaScript. Contrariamente ao que possa levar a crer, é possível obter desempenhos excelentes, devido a um sistema de pré-compilação para código-máquina que a própria Engine disponibiliza.

Das empresas que já estão a tirar proveito destacam-se:

- Groupon
- LinkedIn
- Microsoft
- PayPal
- Rakuten
- SAP
- Walmart
- Yahoo!

### 2.2.2. Cassandra

Focada no armazenamento de grandes quantidades de informação, espalhada por vários servidores, este sistema de base de dados não relacionais oferece grande robustez, boa prevenção de falhas, replicação assíncrona e baixa latência. É também conhecida pelo seu alto nível de *throughput* de dados, em detrimento de escritas e leituras mais demoradas. É sobretudo uma base de dados orientada a colunas, tendo também uma componente *key-value* para organização de linhas de registo.

O acesso externo à informação é feito via CQL, uma alternativa ao SQL, estando disponíveis drivers para acesso nas linguagens Java (JDBC), Python (DBAPI2), Node.js (Helenus) e Go (gocql).

Das empresas que estão a tirar proveito deste sistema de base de dados destacam-se:

- Digg
- Netflix
- Reddit
- SoundCloud

### 2.2.3. MongoDB

Destaca-se por ser um sistema de gestão de base de dados (DBMS) totalmente open-source e sem grandes empresas a dirigir. Este DBMS não relacional é um dos mais usados para pequenas e medias empresas de desenvolvimento de *software* quando optam por soluções NoSQL. Esta popularidade justifica o facto de ser um dos DBMS NoSql com mais suporte nas diversas linguagens e sistemas.

Ao contrário dos típicos DBMS relacionais e da maioria dos DBMS NoSQL, o MongoDB é orientado a documentos, guardando a informação numa estrutura semelhante a JSON. A sua capacidade de balanceamento de carga, replicação e a possibilidade de agregação de dados, juntamente com um foco em sistemas de menor porte, tornam-no um bom sistema para soluções onde não seja possível dispor de muitos servidores, mas mesmo assim se pretenda ter redundância e alta disponibilidade de dados oferecidos por vários servidores.

Apesar de não ser dos sistemas com maior performance, este é usado com alguma recorrência na fase de prototipagem de vários sistemas e por vezes por importantes empresas nalguns dos seus projetos. Das mais conhecidas, pode-se enumerar:

- SAP
- Sourceforge
- Foursquare
- eBay
- CERN
- Shutterfly

# 3. Framework Android

Focado especialmente nos dispositivos móveis, Android é um sistema operativo baseado no *kernel* Linux e atualmente desenvolvido pela Google. É encontrado especialmente em *smartphones* e *tablets*, mas também é possível encontrar em *smart TV's*, *smart-watch's*, consolas de jogos e outros dispositivos domésticos.

A interface com utilizador geralmente é feita através de ecrã táctil, oferecendo por isso uma experiência de utilização simples e intuitiva. Isto, aliado ao facto de ser um sistema operativo de utilização livre, são alguns dos pontos que o levam a ser um dos sistemas operativos de maior sucesso da atualidade.

O seu nível de popularidade tem crescido igualmente entre companhias que pretendem dispositivos baratos, bastante personalizáveis e de fácil desenvolvimento, o que levou a um grande aumento de dispositivos no mercado, permitindo baixar os preços e aliciando ainda mais a sua aquisição por parte do utilizador final.

Devido a ser um sistema operativo que está constantemente a ser desenvolvido, é costume o lançamento de novas versões. Estas alterações podem ser na forma de uma atualização de uma aplicação ou um completo *upgrade* a todo o sistema Android.

Até à data de escrita desde documento, as versões existentes são as que se pode verificar na Tabela 4 [6].

| **Versão** | **API** | **Nome** | **Lançamento** |
|---|---|---|---|
| 1.0 | 1 | - | 23/09/2008 |
| 1.1 | 2 | - | 09/02/2009 |
| 1.5 | 3 | Cupcake | 27/04/2009 |
| 1.6 | 4 | Donut | 15/09/2009 |
| 2.0 | 5 | Eclair | 26/10/2009 |
| 2.0.1 | 6 | Eclair | 03/12/2009 |
| 2.1 | 7 | Eclair | 12/01/2010 |
| 2.2 - 2.2.3 | 8 | Froyo | 20/05/2010 |
| 2.3 – 2.3.2 | 9 | Gingerbread | 06/12/2010 |
| 2.3.3 – 2.3.7 | 10 | Gingerbread | 09/02/2011 |
| 3.0 | 11 | Honeycomb | 22/02/2011 |
| 3.1 | 12 | Honeycomb | 10/05/2011 |
| 3.2 | 13 | Honeycomb | 15/07/2011 |
| 4.0 – 4.0.2 | 14 | Ice Cream Sandwich | 18/10/2011 |
| 4.0.3 – 4.0.4 | 15 | Ice Cream Sandwich | 16/12/2011 |
| 4.1 | 16 | Jelly Bean | 09/07/2012 |
| 4.2 | 17 | Jelly Bean | 13/11/2012 |
| 4.3 | 18 | Jelly Bean | 24/07/2013 |
| 4.4 | 19 | KitKat | 31/10/2013 |
| 4.4W | 20 | KitKat | - |
| 5.0 | 21 | Lollipop | 03/11/2014 |

*Tabela 4 - Versões públicas do Sistema Operativo Android*

## 3.1. Aceitação no Mercado

Em 2013 foi o sistema operativo pré-instalado para dispositivos móveis com mais vendas a nível mundial, fator este que aliado ao facto de estar presente em dispositivos de diversas marcas, levaram a que o Android já na altura fosse e continue a ser o sistema operativo dominante no mercado. Tal como se pode ver na Figura 3, este apresentou uma subida notória ao longo dos últimos anos e que de acordo com o estudo [12] feito pela IDC (International Data Corporation), no 3º quadrimestre de 2014 o Android já contava com um sharing de 84% entre os dispositivos em utilização.

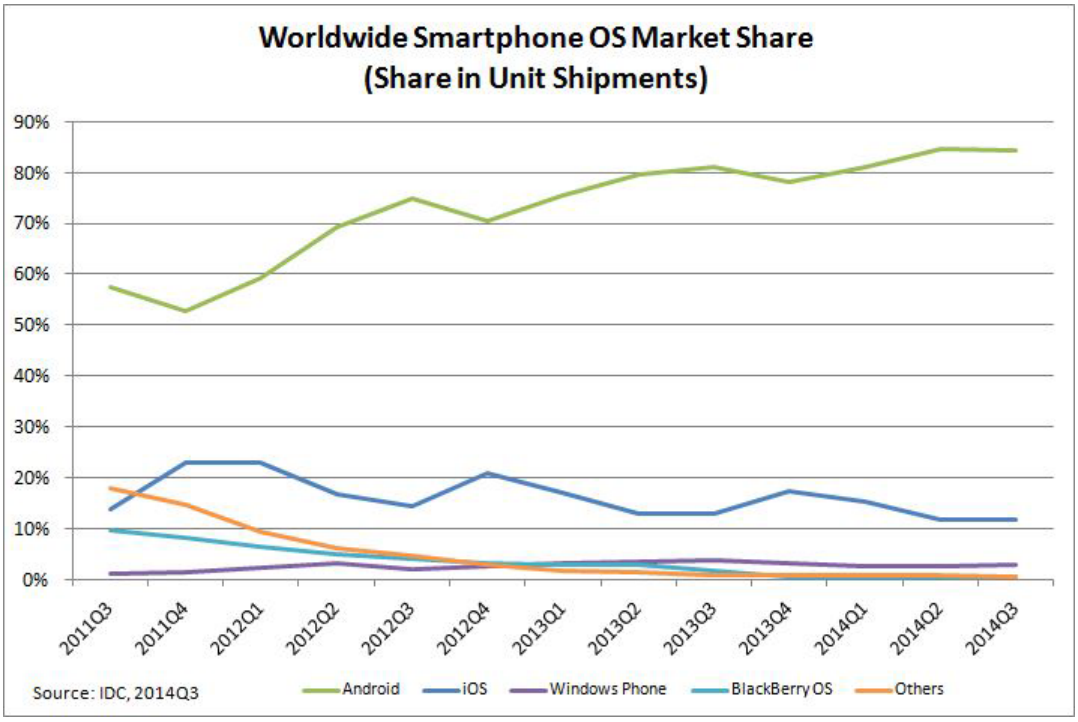



*Figura 3 - Vendas de dispositivos móveis por Sistema Operativo*

Estima-se também que o número de vendas de dispositivos com sistema operativo Android, entre 2012 e 2014, tenha sido próximo do número de computadores pessoais atualmente existentes em todo o mundo. Esta vendas tornavam previsível que outros números tomassem o mesmo rumo, como foi o caso de em julho de 2013 a Google Play ter anunciado a meta de 1 milhão de aplicações publicadas na sua plataforma[13], bem como um total de 50 mil milhões de *downloads* das mesmas. Já em 2014 a Google anunciou a existência de mais de 1 mil milhões de utilizadores de Android[14] ativos mensalmente.

Como forma de dar um indicador da subdivisão de plataformas, a Google disponibiliza semanalmente um gráfico com o nível de popularidade das versões de Android mais usadas

[12] http://www.idc.com/prodserv/smartphone-os-market-share.jsp [Último acesso a 12 de Dezembro de 2014]

[13] http://mashable.com/2013/07/24/google-play-1-million/ [Último acesso a 12 de Dezembro de 2014]

[14] http://tek.sapo.pt/noticias/negocios/android_vai_ter_mais_mil_milhoes_de_utilizado_1358499.html [Último acesso a 12 de Dezembro de 2014]

a nível mundial. Os valores que se podem ver na Tabela 5 e Figura 4[15] são os divulgados no momento de escrita deste relatório.

| Versão | Codename | API | Distribuição |
|---|---|---|---|
| 2.2 | Froyo | 8 | 0.70% |
| 2.3.x | Gingerbread | 10 | 11.40% |
| 4.0.3 | Ice Cream Sandwich | 15 | 9.60% |
| 4.0.4 | | | |
| 4.1.x | Jelly Bean | 16 | 25.10% |
| 4.2.x | | 17 | 20.70% |
| 4.3 | | 18 | 8.00% |
| 4.4 | KitKat | 19 | 24.50% |

*Tabela 5 - Utilização das várias versões de Android (Tabela)*

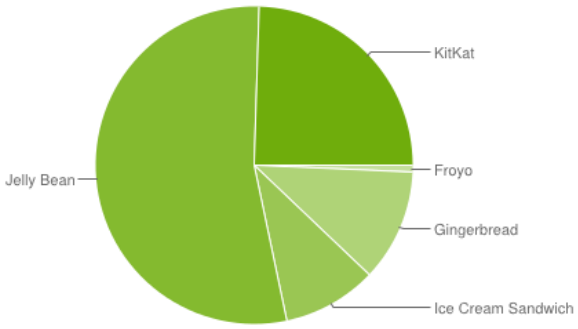



*Figura 4 - Utilização das várias versões de Android (Figura)*

## 3.2. Desenvolvimento de Aplicações

Um dos fatores de sucesso do Android foi ter conseguido despertar o interesse tanto de utilizadores como de programadores de todo o mundo. De facto um estudo conduzido pela Developer Economics no 3º trimestre de 2014 revelou que 74% dos programadores de tecnologias móveis programam para Android[16], números estes que também se refletem no numero crescente de aplicações disponibilizadas.

Normalmente o desenvolvimento de aplicações Android pode ser feito de 5 formas:

- Android SDK[17] – Escrito segundo a notação de Java 6 e em parte compatível com este, é considerado o método padrão de desenvolvimento de aplicações Android. Tal como a maioria das implementações de Java, permite chamadas JNI para execução em código-máquina.
- Android NDK – Normalmente uma forma complementar e associada ao Android SDK, permite criar bibliotecas ou aplicações completas para Android em C, C++ e outras linguagens compiladas em código-máquina. Dado a sua complexidade e a incompatibilidade do código-máquina em arquiteturas diferentes, o seu uso só é encorajado para situações em que traga benefícios notórios à aplicação.
- HTML/JavaScript[18] – Embora não seja um método nativo dos dispositivos móveis, é possível criar aplicações recorrendo quase inteiramente a conteúdo Web. Recorrem com frequência à *engine* WebKit, e mais recentemente à própria *engine* do Chromium. É possível obter um grau de compatibilidade entre dispositivos suficientemente alto, ao ponto de haver diversas aplicações a usarem extensivamente este método e inclusive tenham sido lançado livros a explicar os passos detalhadamente, como por exemplo [7].

[15] Dados retirados de https://developer.android.com/about/dashboards/index.html a 12 de Dezembro de 2014

[16] http://www.developereconomics.com/reports/developer-economics-q3-2014/ [Último acesso a 12 de Dezembro de 2014]

[17] http://developer.android.com/sdk/index.html [Último acesso a 12 de Dezembro de 2014]

[18] http://developer.android.com/guide/webapps/index.html [Último acesso a 12 de Dezembro de 2014]

- Cordova / PhoneGap – Trata-se de um misto entre Android SDK, Android NDK e HTML. Estas frameworks permitem escrever aplicações em HTML, JavaScript e CSS, com a capacidade de chamar funcionalidades do sistema, graças a uma camada intermédia que interliga as chamadas de JavaScript às chamadas nativas. Este método de desenvolvimento está em expansão, principalmente pelo facto das aplicações escritas poderem ser convertidas para diferentes plataformas quase de forma direta.
- Linguagens "não oficiais" / emuladores – Dado a sua flexibilidade e o grande número de aplicações disponíveis, é possível encontrar compiladores e interpretadores de linguagens normalmente não encontradas em dispositivos móveis (p. ex. Perl, Python, AIR, PHP) e até inclusive é possível encontrar emuladores que permitam correr código desenhado para sistemas diferentes (p. ex. programas para ATARI, ZX Spectrum, MSX, MSDOS).

Exceto para linguagens não-oficiais, normalmente o desenvolvimento de aplicações Android implica ter o Android SDK. Este além de permitir o desenvolvimento comum em Java, também engloba um conjunto de bibliotecas, *debuggers*, emuladores, documentação, exemplos e tutoriais. Este SDK é suportado nas plataformas Windows, Mac e Linux, permitindo completa liberdade ao programador. As IDE's suportadas são muito variadas, apesar de oficialmente ser suportado o Eclipse através do *plugin* ADT e o Android Studio (baseado no IntelliJ IDEA).

As aplicações instaláveis para Android têm a extensão ".apk", semelhante ao formato ".jar" ou ".zip". No seu interior contém ficheiros ".dex" (bytecode compilado), ".so" (código-máquina resultante de código C++), resources e ficheiros de configuração.

Apesar da programação em Android ser normalmente feita em Java, isto não significa que aplicações Java SE sejam compatíveis, devido sobretudo ao Android usar a *runtime* Dalvik ou Android Runtime (ART) em vez da Java Virtual Machine. O mesmo acontece com aplicações nativas de GNU/Linux, que apesar do Android correr sobre um *kernel* Linux, o uso da biblioteca Bionic em vez da GNU C Library faz com que a maioria das funcionalidades essenciais para aplicações de GNU/Linux não estejam disponíveis. Apesar destas incompatibilidades não serem o cenário ideal, a utilização destas *runtimes* específicas permite poupar recursos e obter grandes desempenhos, algo vital num dispositivo embebido de uso comum, em detrimento de outras funcionalidades não tão importantes.

## 3.3. Tecnologias associadas

Tratando-se de uma implementação distinta de java, a framework Android tem outras diferenças associadas. Existe um conjunto de ficheiros e classes chave que são encontrados em qualquer aplicação, os quais vão ser enumerados e devidamente explicados no decorrer deste subcapítulo.

### 3.3.1. AndroidManifest.xml

AndroidManifest.xml trata-se de um ficheiro em formato xml que deve estar obrigatoriamente na raiz de todas as aplicações Android. Descreve valores que afetam globalmente a aplicação (p.ex. versão mínima de Android suportada), define quais os componentes (p.ex. Activities) que podem ser executados e regras específicas para cada componente.

Contrariamente à maioria dos paradigmas de programação, em Android não existe uma função main() indicadora do ponto de partida da aplicação. Esta função é desempenhada pelo AndroidManifest.xml, que na maioria das aplicações, associa um ícone *launcher* ao arranque de uma Activity. Similarmente é também este ficheiro que define as permissões a serem oferecidas à aplicação, normalmente apresentadas aquando da instalação da mesma.

Um exemplo deste ficheiro retirado do programa de demonstração HelloWorld pode ser visto na Figura 5.

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.my_domain.app.helloactivity">

  <application android:label="@string/app_name">

    <activity android:name=".HelloActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
```

*Figura 5 - Exemplo de ficheiro AndroidManifest.xml*

### 3.3.2. Strings.xml

Tratando-se o Android de um sistema desenhado para ser utilizado por pessoas de todo o mundo, é fundamental que suporte a internacionalização.[19] Isto oferece não só uma tradução direta de palavras, mas sim uma completa adaptação das aplicações conforme a língua pretendida.

Um dos elementos mais importante para o suporte a internacionalização é certamente o ficheiro strings.xml, localizado obrigatoriamente em `res/values/strings.xml`, opcionalmente também noutras pastas para associar a um atributo específico do dispositivo de destino. É neste ficheiro onde são armazenados todos os textos a serem apresentados na aplicação, em formato UTF-8[20], assim como indicações de como o texto deve ser apresentado sobre determinadas circunstâncias.

Um exemplo deste ficheiro retirado do programa de demonstração Hello World pode ser visto na Figura 6.

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

*Figura 6 - Exemplo de ficheiro strings.xml*

[19] https://developers.google.com/international/i18n [Último acesso a 12 de Dezembro de 2014]

[20] http://tools.ietf.org/html/rfc3629 [Último acesso a 12 de Dezembro de 2014]

### 3.3.3. Activity

As *Activities* são um elemento que permite a interação com o utilizador, fazendo um elo entre a parte funcional e a parte gráfica da aplicação. É à *Activity* quem cabe a tarefa selecionar os elementos gráficos a serem apresentados e a definição de propriedades dos mesmos. Por padrão, a framework Android permite que uma *Activity* lance outra *Activity*, em que a *Activity* que lança é colocada em *background* numa pilha de atividades, só sendo possível voltar atras terminando a *Activity* que se encontra no cimo da pilha.

Do ponto de vista de programação, uma *Activity* trata-se de uma classe Java que estende a classe *Activity* e tem um ciclo de vida de acordo com o apresentado na Figura 7.[21]

Até ao lançamento do Android 3.0, este era o único meio de apresentar elementos gráficos e fazer interação com o utilizador. A partir desta altura foram adicionados *Fragments*.

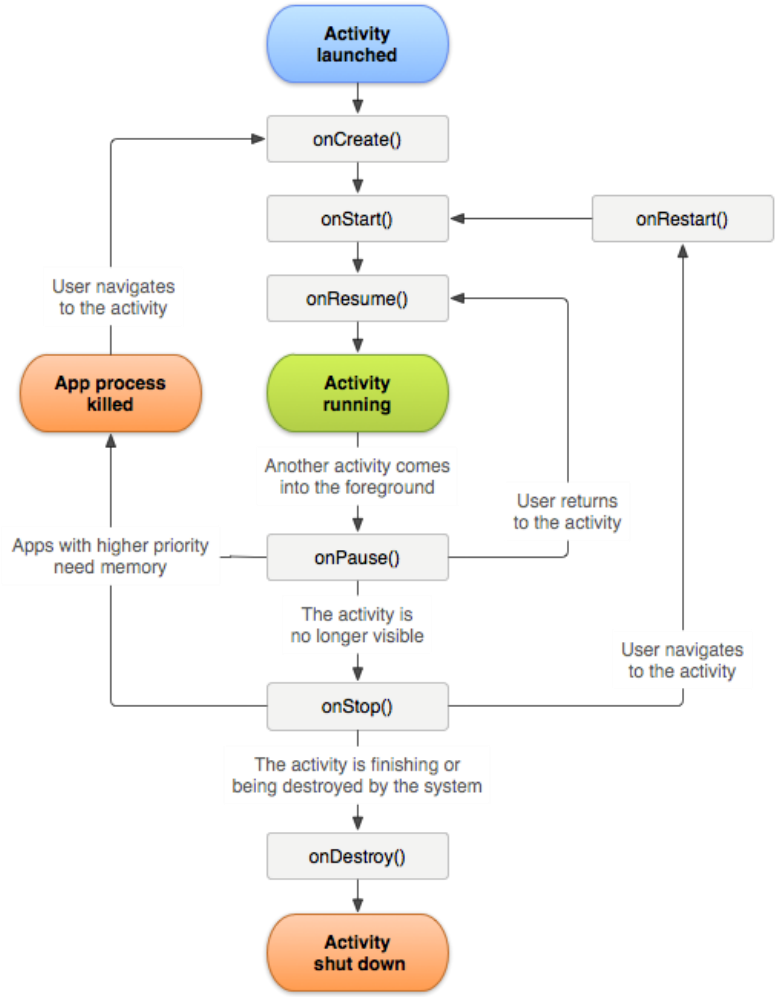



*Figura 7 - Ciclo de vida das Activities*

[21] http://developer.android.com/reference/android/app/Activity.html [Último acesso a 12 de Dezembro de 2014]

### 3.3.4. Fragment

Um *Fragment* representa uma porção da interface visual contido numa *Activity*, podendo ser apresentado um único *Fragment* por *Activity*, ou vários em simultâneo. Cada *Fragment* é visto como uma unidade independente, com o seu próprio ciclo de vida, os seus próprios eventos e pode ser adicionado ou removido a qualquer momento da *Activity*.

Este conceito foi adicionado no Android 3.0 com o objetivo de suportar interfaces visuais mais dinâmicas em dispositivos de ecrã de maiores dimensões, tal como o caso dos tablets. É atualmente recomendada a sua utilização para conter os elementos visuais das aplicações, em vez das *Activities*.

Do ponto de vista de programação, cada *Fragment* é uma classe que estende a classe *Fragment*, deve conter um único construtor vazio e sem argumentos. Além disso tem o seu próprio ciclo de vida (Figura 8) e este é diretamente afetado pelo ciclo de vida da própria *Activity* (p.ex. um pause seguido de *resume* na *Activity*, induz a um *onDestoryView*, *onCreateView* e todos os métodos intermédios).[22]

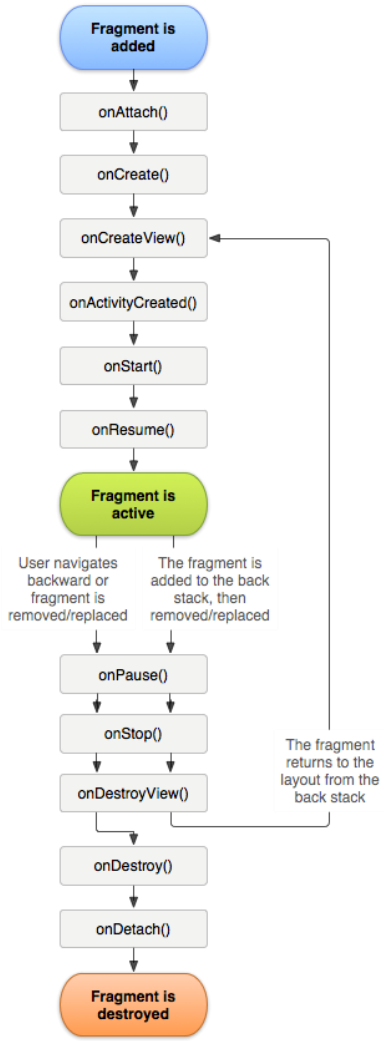



*Figura 8 - Ciclo de vida de um Fragment*

[22] http://developer.android.com/guide/components/fragments.html [Último acesso a 12 de Dezembro de 2014]

### 3.3.5. Intent

Um *Intent* é uma forma abstrata de descrever uma ação a ser executada, normalmente usado para lançar *Activities*, tanto da própria, como de outras aplicações.

Geralmente é constituído pelos campos:

- *action* – Define a ação a ser tomada (p. ex. lançar *Activity*, abrir uma URL no browser padrão, abrir uma imagem no visualizador de fotos)
- *data* – Define qualquer informação adicional, no contexto da *action* escolhida (p. ex. URL da pagina a abrir no *browser* padrão)

É igualmente usado para descrever eventos recebido do sistema, quando previamente declarados no AndroidManifest.xml (p. ex. coneção do carregador, receção de SMS, etc).

Para uma lista completa de usos possíveis, a melhor fonte será certamente a página oficial do Android Developer dedicado a *Intents*[23].

### 3.3.6. HttpClient

*HttpClient* é uma biblioteca para suporte de comunicação HTTP, embutido na framework Android desde a versão 1.0. Teve origem na biblioteca *HttpClient*, resultado de um subprojecto de *Jakarta Commons*, atualmente substituída pelo projeto Apache *HttpComponents*.[24] A sua compatibilidade com Java (JVM) e Android, fazem desta uma biblioteca bastante utilizada para aceder a recursos remotos, através do protocolo HTTP, e consequentemente outros serviços que tirem partido deste mesmo protocolo.

### 3.3.7. WebView

Além da possibilidade de abrir páginas através do *browser* padrão com recurso a *Intents*, é igualmente possível apresentar o conteúdo das páginas diretamente na aplicação. Para isto é necessário recorrer a uma *WebView*, um componente que tira partido da *engine WebKit* ou *Chromium* para apresentar eficientemente páginas web, tanto em acesso local como remoto. Associado às *engines* que correr por detrás da *WebView*, são igualmente suportados CSS e JavaScript, sendo costume fazer-se um misto entre código nativo e código JavaScript, com forte ênfase aos elementos gráficos providenciados pelo próprio HTML e CSS.

A utilização deste componente como classe resume-se ao exemplo da Figura 9[25].

```
WebView webview = new WebView(this);
setContentView(webview);
webview.getSettings().setJavaScriptEnabled(true);
webview.loadUrl("http://developer.android.com/");
```

*Figura 9 - Exemplo de uso de WebView*

[23] http://developer.android.com/reference/android/content/Intent.html [Últ. acesso a 12 de Dezembro de 2014]

[24] http://hc.apache.org/httpclient-3.x/ [Último acesso a 12 de Dezembro de 2014]

[25] http://developer.android.com/reference/android/webkit/WebView.html [Últ. acesso a 12 de Dezembro de 2014]

### 3.3.8. Localização

Um dos aspetos que permite melhorar a experiência de utilização de um dispositivo móvel, é este ter a capacidade de sentir o meio físico e tomar uma ação de acordo com certos parâmetros (ou em inglês, ser *Context-Aware*). Uma dessas formas de sentir, é precisamente ter capacidade de se localizar no espaço.

Pensado principalmente para dispositivos móveis, o Android é capaz de fazer geo-referenciação na forma de coordenadas cartográficas, usando a norma WGS84[26]. Para obter esta localização, o Android dispõe dos seguintes meios:

- GPS – Tira partido do recetor GPS presente em muitos dispositivos móveis, especialmente smartphones
- Rede – Com recurso a uma ligação de dados, é feito um reverse-lookup das redes Wifi e identificador da estação transmissora GSM, para obter as coordenadas aproximadas
- Hibrido – Também conhecido por Assisted GPS (AGPS), tira partido do conjunto GPS e rede sempre que possível, fazendo uma melhor gestão de consumo energético, com uma boa precisão, mesmo dentro de edifícios.
- Passivo – Embora não seja um método de localização fidedigno, permite que uma aplicação eventualmente receba uma localização, sem que seja necessário forçar o sistema dispensar recursos de forma prioritária

Embora não seja regra, por norma, e de acordo com a documentação de suporte [8], estes meios de localização apresentam as qualidades e defeitos presentes na Tabela 6.

| **Meio** | **GPS** | **Rede** |
|---|---|---|
| **Pros** | Mais fiável em *outdoor* | Bastante fiável *indoor* e em cidades<br>Consumo energético baixo |
| **Contras** | Fraco dentro de edifícios<br>Alto consumo de energia | Fraco-médio desempenho *outdoor* |
| | | |
| **Meio** | **Hibrido** | **Passivo** |
| **Pros** | Fiável tanto *indoor* como *outdoor*<br>Boa gestão de consumo de energia | O menor consumo de energia |
| **Contras** | Consumo energético baixo-médio | Não tem forma de prever quando obterá a localização |

*Tabela 6 - Comparação de meios de obtenção de localização em Android*

[26] https://confluence.qps.nl/pages/viewpage.action?pageId=29855173 [Últ. acesso a 12 de Dezembro de 2014]

# 4. Trabalho de Investigação

Nos últimos anos tem existido um aumento exponencial de utilizadores de dispositivos móveis, especificamente *smartphones* e *tablets*. Este aumento contribuiu para que houvesse uma maior procura e oferta de soluções para estes mesmos dispositivos e para outros serviços associados.

Dado também ao aumento de número de utilizadores de internet a nível mundial e ao aumento da quantidade de dados por estes produzidos, tem-se procurado novas soluções para tratar e armazenar dados, através de armazenamento e processamento distribuído, novos métodos de tratamento de informação e acima de tudo, sistemas que permitam um alto grau de disponibilidade.

Com tantas tecnologias novas, torna-se difícil saber o que realmente as distingue e qual as mais indicadas para cada solução em concreto, inclusive para este projeto. Por estes motivos, considerou-se uma mais valia fazer um trabalho de investigação com vista a ter uma noção das tecnologias disponíveis, qual a sua eficiência para a função proposta e decidir qual a que mais se enquadra dentro do âmbito do projeto.

Os pontos avaliados foram:

- Bibliotecas de leitura de códigos de barras
- Protocolos de comunicação
- Frameworks para construção de Web Services
- Sistemas de Gestão de Base de Dados

Estes pontos serão descritos com mais detalhe no seguinte subcapítulo.

## 4.1. Áreas de investigação

Aos 4 pontos mencionados na introdução deste capítulo, foi aplicado um conjunto de trabalhos de pesquisa e testes de performance e disponibilidade. Os resultados estão descritos nas seguintes secções.

### 4.1.1. Bibliotecas de leitura de códigos de barras

Códigos de barras é uma representação de informação passível de ser lida por uma máquina através de um meio ótico, normalmente usada para identificação de artigos. Foi criado por Joseph Woodland e Bernard Silver.[27]

Originalmente surgiram como um conjunto de pontos semelhantes a código morse, tendo evoluído para um sistema de barras de espaçamento e largura variável, sendo este o método usado para codificar a informação, muitas vezes também designado código de barras linear ou de uma dimensão (1D). Mais recentemente foram criados códigos de barras mais complexos, usando retângulos, pontos e outras formas geométricas, num formato em duas dimensões (2D). Embora não limitada a estas, os sistemas de códigos de barras 1D e

[27] http://hub.aa.com/en/aw/so-woodland-bar-code-bernard-silver-drexel-university [Último acesso a 12 de Dezembro de 2014]

2D têm elevada importância nas áreas retalhistas, industrial, automobilística e farmacêuticas, especialmente para identificação de peças e produtos.

Para o reconhecimento de código de barras ser possível e viável, é necessária a capacidade de reconhecimento de padrões de forma automatizada, rápida e eficiente. Embora este reconhecimento seja facilmente obtido com recurso a computadores comuns, isto nem sempre é trivial usando dispositivos móveis. Neste sentido têm-se feito estudos de algoritmos para atingir este fim, usando o mínimo de recursos possível [9] [10]. Graças a formas similares de captação de código de barras, é possível que dispositivos móveis façam a captação de códigos de barras com tempos de resposta excelentes. Existem várias ferramentas que usam estas funcionalidades e inclusive existem projetos que tiram partido desta e de outras tecnologias semelhantes para identificação [11] [12] [13].

Especificamente na área retalhista, existe uma entidade reguladora designada GS1[28], com o intuito de definir o tipo de código de barras para cada aplicação e reservar frações de códigos, com o intuito de impedir colisões de códigos de produtos diferentes. Entre estes, os tipos de código de barras usados com mais frequência encontram-se presentes na Tabela 7.

| Nome | Capacidade (Caracteres) | Exemplo |
|---|---|---|
| EAN-13 | 13 |  |
| EAN-8 | 8 |  |
| UPC-A | 13 |  |
| UPC-E | 8 |  |

*Tabela 7 - Exemplos de códigos de barras usados para fins retalhistas*

Apesar de por norma os equipamentos de leitura serem compatíveis com ambos, os formatos UPC-A e UPC-E são normalmente encontrados em produtos americanos, enquanto os formatos EAN-8 e EAN-13 (e possíveis variações dos mesmos) são usados no resto do mundo.

[28] http://www.gs1.org/ [Último acesso a 12 de Dezembro de 2014]

Ao contrário do que é usado na indústria automóvel, estes sistemas de códigos de barras são destinados apenas à identificação dos produtos, e não à sua serialização, que na prática significa que 2 produtos iguais, vendidos em estabelecimentos comerciais diferentes, têm obrigatoriamente o mesmo identificador de código de barras.

Embora não haja nenhuma restrição tecnológica, ao longo deste trabalho, o código de barras em formato EAN-13 será o ponto central de estudo, visto ser o mais comum para fins retalhistas e mais fácil de obter em território nacional. Tratando-se de um formato de utilização mundial, o qual é de domínio publico, é evidente que existirá um conjunto de ferramentas já disponíveis que permita a leitura e reconhecimento destes códigos. A pesquisa associada a isto está descrita nos seguintes subcapítulos.

#### *4.1.1.1. Tecnologias Disponíveis*

Com uma rápida pesquisa por leitores de códigos de barras usados nas superfícies comerciais, chega-se à imediata conclusão que um simples leitor tem custos notáveis, que seriam de todo o interesse minimizar. Através de pesquisas no âmbito deste projeto, chegou-se à conclusão que uma forma de reduzir os custos associados seria a utilização do próprio smartphone como dispositivo de captura e identificação de códigos de barra, para identificação dos respetivos produtos.

Com alguma pesquisa e analisando aplicações que fazem uso de leitura de código de barras, foi identificado um número razoável de bibliotecas para reconhecimento de códigos EAN e outras variantes em tempo real. Usando como critério de filtragem ser compatível com Android, ser de livre utilização para fins não comerciais e não aparentar ser um projeto abandonado, foi de imediato reduzido o número de bibliotecas, para algo muito mais compacto e viável de testar exaustivamente, mais especificamente as bibliotecas ZXing, Scandit Barcode Scanner e zBar.

Para testar estas mesmas bibliotecas, optou-se por começar por experimentar demos ou aplicações que façam uso direto das mesmas, todas elas distribuídas gratuitamente através da plataforma Google Play. Após usar as demos e pesquisar um pouco acerca da utilização das mesmas, de imediato se tirou as seguintes conclusões de cada uma:

**ZXing**

Identificador da aplicação: com.google.zxing.client.android

Criador: ZXing Team

Data da última modificação: 2/3/2014

Versão de Android necessária: 1.5

Pontos fortes:

- Suporte de um grande número de formatos de códigos de barras 2D e 3D
- Rápido reconhecimento de códigos de barras na presença de boa iluminação e boa qualidade de imagem
- Fácil de usar através de Intent como aplicação separada na plataforma Android

Pontos fracos:

- Dificuldades em reconhecimento na ausência de iluminação adequada ou no uso de uma câmara de baixa resolução
- Algumas leituras erradas de códigos de barras EAN
- Não é tão trivial incluir numa aplicação como sendo um *package* único
- O uso como aplicação externa via Intent não permite leituras continuadas

**Scandit Barcode Scanner**

Identificador da aplicação: com.scandit.demoapp

Criador: Scandit Inc.

Data da última modificação: 24/1/2014

Versão de Android necessária: 2.2

Pontos fortes:

- Boa velocidade de leitura
- Lida com baixas iluminações e má qualidade de imagem
- Bastante fácil de incorporar numa aplicação
- Permite múltiplas leituras continuadas com códigos de diferentes tipos

Pontos fracos:

- Suporte relativamente reduzido de formatos de códigos de barras suportados
- Não é uma ferramenta completamente gratuita, tendo custos associados caso se use em projetos comerciais ou se pretenda suporte a mais formatos
- Limite de instalação em 1.000.000 dispositivos por api key
- Com boas condições de iluminação e boa qualidade de imagem, tem desempenho inferior ao ZXing.

**zBar**

Identificador da aplicação: com.theappsembly.simplebookshelf

Criador: Jeff Brown

Data da última modificação: 02/02/2013

Versão de Android necessária: 2.2

Pontos fortes:

- Boa velocidade de leitura
- Permite múltiplas leituras continuadas com códigos de diferentes tipos
- Disponível também para desktop e iOS

Pontos fracos:

- Suporte relativamente reduzido de formatos de códigos de barras
- Focada maioritariamente em iOS
- As bibliotecas para Android não são oficiais e a maioria teve o seu desenvolvimento abandonado

Após os testes anteriores, partiu-se para a criação de pequenos protótipos para validar e avaliar o processo e a facilidade de integração das bibliotecas numa aplicação real.

As bibliotecas Zxing e Scandit Barcode Scanner permitiram criar exemplos de programas simples, funcionais e de fácil compreensão. No caso da Zxing, pode optar-se por instalar como programa externo (chamado via Intent), facilitando muito o processo de utilização da mesma.

Embora a aplicação exemplo do Scandit Barcode Scanner que se encontra no Google Play requeira no mínimo Android 2.2, foi possível criar um exemplo que requer no mínimo Android 2.1, algo que quase no fim do projeto foi alterado com a versão mínima a passar para 2.3.

Devido à complexidade da documentação e à nítida falta de foco na plataforma Android, a biblioteca zBar foi imediatamente posta de parte.

#### *4.1.1.2. Discussão*

Desde o início da fase de pesquisa, a biblioteca Zxing mostrou ser a melhor aceite pela maioria dos programadores da plataforma Android, devendo-se isto sobretudo por ser uma biblioteca inteiramente open source e com suporte a uma vasta gama de códigos 1D e 2D. A esmagadora maioria das aplicações Android que necessitam de fazer leitura de código de barras usam esta biblioteca.

A biblioteca Scandit Barcode Scanner não é open-source e tem uma gama de códigos suportados muito mais restrita (especialmente na versão gratuita). O seu ponto forte no entanto é a velocidade de leitura em condições e resolução menos favoráveis e a possibilidade de ler um código EAN 13 mesmo com grandes inclinações da câmara e imagem mal focada.

Por último não foi possível ver o potencial da biblioteca zBar, devido à sua complexidade e falta de material de suporte para a arquitetura Android.

### 4.1.2. Protocolos de comunicação

Fruto da proliferação das redes de comunicação e da necessidade de interação entre máquinas de arquiteturas diferentes, os protocolos de comunicação foram desde os primórdios uma forma de assegurar a eficiente troca de dados. Com o aumento do número de sistemas que devem operar entre si e o aumento da complexidade que isso implicava, começaram a surgir protocolos específicos para facilitar a transferência de dados e chamada a procedimentos remotos. Como era de esperar, isto teve forte impacto na forma de construir aplicações e foi uma das chaves do sucesso para a comunicação em rede em

ambiente empresarial e industrial. Foi também alvo de muitos estudos e melhorias ao longo dos anos [14] [15].

Com a chegada da Internet e mais recentemente com a massificação de tecnologias Web e dispositivos móveis, estes mostraram mais uma vez serem essenciais, mas não devidamente adaptados às necessidades. Isto levou ao aparecimento de mais protocolos, cada um focado numa área específica, geralmente a requerer muitos poucos recursos. Com isto, na atualidade existe um grande número de protocolos para permitir a comunicação entre máquinas diferentes. No contexto deste projeto, visto que existe a necessidade de comunicação com dispositivos móveis, deu-se prioridade aos que requerem menor recursos, sejam de processamento fácil, compatíveis entre diversos sistemas, suportem falhas de comunicação sem grandes problemas e tenham um baixo *overhead* no processo de comunicação.

Associado especificamente a este *overhead* e à baixa capacidade de processamento, existe uma grande discussão sobre o tema JSON vs. XML, onde nitidamente XML se encontra a cair em desuso [16].

Fez-se então uma pesquisa por vários projetos que requeriam comunicação de dados [17] [13] e bibliotecas que as providenciassem, obtendo-se a seguinte lista de protocolos possíveis:

- JSON-RPC
- REST
- CORBA
- WebSockets
- Eventsource
- XML-RPC
- SOAP

Estes protocolos serão analisados no seguimento deste relatório.

#### *4.1.2.1. Resultados Obtidos*

A seleção do protocolo de comunicação baseou-se principalmente em pesquisa e por familiaridade devido à utilização em trabalhos curriculares. Como resultado, foi possível extrair as informações apresentadas na Tabela 8.

| Protocolo | Detalhes |
|---|---|
| **Nome:** JSON-RPC<br>**Função:** RPC<br>**Encapsulamento:** JSON<br>**Suporte Android:** Sim<br>**Suporte nativo Android:** Não | • Protocolo simples, leve e fácil de implementar<br>• Baixo overhead<br>• Permite receção de notificações do servidor<br>• Dispõe de muitas bibliotecas para Android |
| **Nome:** REST<br>**Função:** RPC/Comm server-client<br>**Encapsulamento:** JSON/XML<br>**Suporte Android:** Sim<br>**Suporte nativo Android:** Sim | • É dos protocolos mais usados para API's públicas<br>• Compatibilidade com a maioria das linguagens<br>• Protocolo leve, extremamente simples<br>• É *sessionless*, requerendo o uso de mecanismos extras de autenticação<br>• Não suporta chamadas assíncronas, sendo necessário o uso de funções bloqueantes |
| **Nome:** CORBA<br>**Função:** RPC<br>**Encapsulamento:** RAW<br>**Suporte Android:** Não<br>**Suporte nativo Android:** Não | • Usado para RPC entre plataformas diferentes.<br>• Muito usada em aplicações *Enterprise*<br>• Não tem suporte nativo em Android |
| **Nome:** WebSockets<br>**Função:** Comm server-client<br>**Encapsulamento:** RAW<br>**Suporte Android:** Sim (v4.4)<br>**Suporte nativo Android:** Sim (v4.4) | • Nova tecnologia associada ao HTML5 e javascript<br>• É facilmente adaptável e simples de implementar<br>• Permite chamadas assíncronas em background.<br>• O suporte em Android está presente apenas a partir da versão 4.4 via WebView |
| **Nome:** Eventsource<br>**Função:** Comm server-client<br>**Encapsulamento:** RAW<br>**Suporte Android:** Sim (v4.4)<br>**Suporte nativo Android:** Sim (v4.4) | • É um mecânico para receber eventos de um servidor de forma assíncrona<br>• Está normalmente associado a HTML5 e javascript<br>• Serve para simplificar a receção de eventos vindos do servidor |
| **Nome:** XML-RPC<br>**Função:** RPC<br>**Encapsulamento:** XML<br>**Suporte Android:** Sim<br>**Suporte nativo Android:** Não | • Antecedente do que hoje é conhecido por SOAP<br>• Similar ao JSON-RPC, mas com encapsulamento XML<br>• Maior *overhead* que JSON-RPC<br>• Tecnologia antiga |
| **Nome:** SOAP<br>**Função:** RPC/Troca mensagens<br>**Encapsulamento:** XML<br>**Suporte Android:** Sim<br>**Suporte nativo Android:** Não | • Tecnologia de RPC que usa XML e HTTP para troca de mensagens<br>• Habitualmente associada a J2EE.<br>• Compatível com Android através de bibliotecas<br>• Tem um grande *overhead* à informação. |

*Tabela 8 - Conclusões tiradas da comparação de vários protocolos de comunicação e encapsulamento*

#### *4.1.2.2. Discussão*

Tendo em conta os dados obtidos através da pesquisa e o conhecimento obtido de projetos passados, optou-se assim por se utilizar inteiramente o protocolo REST com uma estrutura JSON, por forma a otimizar o mais possível o tráfego, facilitar as operações de *debug* e aumentar o leque de bibliotecas para tratamento de mensagens.

### 4.1.3. Frameworks para construção de Webservices

Graças à massificação do uso de aplicações móveis e tecnologias web, os Webservices têm sido um dos elementos que merecem mais atenção no desenvolvimento de sistemas de informação e muitas tecnologias novas têm surgido nos últimos anos. Para auxiliar nesta evolução, muitos estudos comparativos são feitos com alguma regularidade e é comum encontrar *websites* criados especificamente para assegura uma comparação contínua ao longo do tempo, em condições semelhantes e muitas vezes com capacidade de voto ou de comentário, para que seja possível discutir acerca da implementação e não unicamente das especificações.

Também associado à massificação das tecnologias Web, existe uma adoção de crescente de frameworks que tirem partido de um padrão reativo para servir pedidos, em vez dos típicos padrões de processos-*worker*, criação de *thread* ou *pool* de *threads*, com a premissa de maximizar o número de pedidos servidos, sem que haja congestão ou esgotamento de recursos. Devido à natureza deste projeto, revelou-se interessante estudar alguma documentação [18] [19], consultar a bibliografia [20], diversos fóruns de discussão sobre o tema e conduzir um estudo comparativo, para validar se justificava-se o uso de uma destas frameworks recentes e ter logo à partida uma decisão formada e bem fundamentada. Este estudo está descrito nos seguintes subcapítulos.

#### *4.1.3.1. Tecnologias disponíveis e experiências efetuadas*

Após uma pesquisa em vários fóruns de discussão, *sites* dedicados ao tema e alguns artigos científicos [21], a escolha e testes efetuados recaíram sobre as frameworks presentes na Tabela 9.

| Nome | Linguagem | Website oficial |
|---|---|---|
| ribs2 | C/C++ | https://github.com/Adaptv/ribs2 |
| Cowboy | Erlang | https://github.com/ninenines/cowboy |
| Yaws | Erlang | http://yaws.hyber.org/ |
| Glassfish | Java | https://glassfish.java.net/ |
| Jetty | Java | http://eclipse.org/jetty/ |
| Netty | Java | http://netty.io/ |
| vert.x | Java | http://vertx.io/ |
| Tomcat | Java | http://tomcat.apache.org/ |
| Node.js | JavaScript | http://nodejs.org/ |
| Mojolicious | Perl | http://mojolicio.us/ |
| Apache+PHP | PHP | http://httpd.apache.org/ |
| nginx+PHP | PHP | http://nginx.org/ |
| ReactPHP | PHP | http://reactphp.org/ |
| asyncore | Python | https://docs.python.org/2/library/asyncore.html |
| CherryPy | Python | http://www.cherrypy.org/ |
| Django | Python | https://www.djangoproject.com/ |
| Tornado | Python | http://www.tornadoweb.org/en/stable/ |
| Twisted | Python | https://twistedmatrix.com/trac/ |
| EventMachine | Ruby | https://github.com/eventmachine/eventmachine |

*Tabela 9 - Lista de frameworks para criação de webservices estudadas*

Para testar as várias frameworks, optou-se por criar uma simples aplicação servidor em cada. O método de teste consiste em fazer pedido GET a um endereço e avaliar o tempo de chegada de resposta e o número de respostas não recebidas. Todas estas respostas são precisamente iguais, sendo idênticas a uma resposta típica de um método GET de HTTP, contendo precisamente o seguinte conteúdo:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 2
Connection: close

OK
```

Para fazer estes pedidos e contabilizar as respostas recebidas, optou-se por utilizar uma ferramenta muitas vezes usada para testar a performance de servidores Web, o Apache Benchmark[29]. Como parâmetros de testes optou-se pelos seguintes parâmetros:

- Enviar um total de 100.000 de pedidos, já que além de ser um valor comum, demonstrou permitir obter uma baixa oscilação de resultados.

[29] http://httpd.apache.org/docs/2.2/programs/ab.html [Último acesso a 12 de Dezembro de 2014]

- Utilizar 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000 e 10000 como número de conexões concorrentes, representa o número de pedidos em simultâneo a serem enviados para o programa servidor
- Repetir cada teste 10 vezes e fazer a média aritmética dos resultados

#### *4.1.3.2. Resultados Obtidos*

Após terminar os testes e feito algum trabalho de organização dos resultados, foi possível extrair os tempos de respostas e as taxas de erro das diferentes frameworks.

Quanto aos tempos obtidos, tal como constatado na Tabela 10, as frameworks ribs2 e vert.x foram as que conseguiram melhores tempos, seguidas da Netty, Tomcat, asyncore e EventMachine. A mesma situação pode ser demonstrada pelo gráfico na Figura 10, juntamente com a existência de valores não lineares, os quais podem ser explicados por pormenores da implementação das várias frameworks.

| | **Time/req (ms) GET simples 100000 requests** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Pedidos concorrentes** | | | | | | | | | | | | |
| **Nome** | **1** | **2** | **5** | **10** | **20** | **50** | **100** | **200** | **500** | **1k** | **2k** | **5k** | **10k** |
| ribs2 | 0.14 | 0.12 | 0.11 | 0.11 | 0.10 | 0.10 | 0.11 | 0.11 | 0.12 | 0.12 | 0.13 | 0.19 | 0.28 |
| Cowboy | 0.44 | 0.36 | 0.33 | 0.31 | 0.28 | 0.24 | 0.23 | 0.24 | 0.25 | 0.24 | 0.27 | 0.35 | 0.45 |
| Yaws | 0.47 | 0.38 | 0.32 | 0.30 | 0.30 | 0.30 | 0.30 | 0.30 | 0.33 | 0.32 | 0.37 | 0.09 | 0.10 |
| Glassfish | 0.40 | 0.30 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.26 | 0.27 | 0.29 | 0.32 | 0.46 | 0.51 |
| Jetty | 0.95 | 0.95 | 0.21 | 0.21 | 0.19 | 0.18 | 0.18 | 0.18 | 0.18 | 0.19 | 0.21 | 0.30 | 0.53 |
| Netty | 0.26 | 0.18 | 0.18 | 0.18 | 0.18 | 0.18 | 0.18 | 0.18 | 0.18 | 0.18 | 0.19 | 0.21 | 0.27 |
| vert.x | 0.20 | 0.13 | 0.11 | 0.10 | 0.10 | 0.11 | 0.11 | 0.11 | 0.12 | 0.13 | 0.14 | 0.17 | 0.26 |
| Tomcat | 0.24 | 0.17 | 0.18 | 0.19 | 0.19 | 0.19 | 0.20 | 0.20 | 0.21 | 0.22 | 0.24 | 0.29 | 0.39 |
| Node.js | 0.46 | 0.37 | 0.35 | 0.35 | 0.35 | 0.35 | 0.35 | 0.35 | 0.35 | 0.36 | 0.08 | 0.09 | 0.10 |
| Mojolicious | 6.21 | 4.32 | 3.22 | 2.72 | 2.53 | 2.35 | 2.37 | 2.41 | 2.44 | 2.43 | 2.39 | 2.29 | 2.22 |
| Apache+PHP | 0.54 | 0.44 | 0.39 | 0.39 | 0.38 | 0.38 | 0.38 | 0.39 | 0.43 | 0.57 | 0.60 | 0.73 | 1.17 |
| nginx+PHP | 0.51 | 0.38 | 0.36 | 0.35 | 0.34 | 0.34 | 0.34 | 0.60 | 1.58 | 1.81 | 0.83 | 0.90 | 0.92 |
| ReactPHP | 0.51 | 0.43 | 0.42 | 0.40 | 0.39 | 0.40 | 0.43 | 0.49 | 0.59 | 0.62 | 0.61 | 0.82 | 0.93 |
| asyncore | 0.23 | 0.15 | 0.14 | 0.14 | 0.14 | 0.14 | 0.14 | 0.15 | 0.21 | 0.30 | 0.29 | 0.45 | 0.65 |
| CherryPy | 1.58 | 1.70 | 1.73 | 1.74 | 1.73 | 1.74 | 1.75 | 1.92 | 2.34 | 2.66 | 5.44 | 6.28 | 5.83 |
| Django | 2.73 | 2.64 | 2.82 | 2.84 | 2.84 | 2.85 | 2.85 | 2.87 | 3.25 | 3.37 | 3.49 | 3.54 | 3.46 |
| Tornado | 0.63 | 0.55 | 0.53 | 0.52 | 0.52 | 0.53 | 0.53 | 0.54 | 0.54 | 0.54 | 0.55 | 0.56 | 0.75 |
| Twisted | 1.09 | 0.98 | 0.93 | 0.91 | 0.90 | 0.90 | 0.90 | 0.91 | 0.91 | 0.91 | 0.94 | 1.05 | 1.25 |
| EventMachine | 0.29 | 0.22 | 0.17 | 0.16 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.16 | 0.22 | 0.40 |

*Tabela 10 - Resultado dos testes aos tempos de resposta*

| **Legenda** | |
|---|---|
| | Melhores tempos |
| | Tempos aceitáveis |

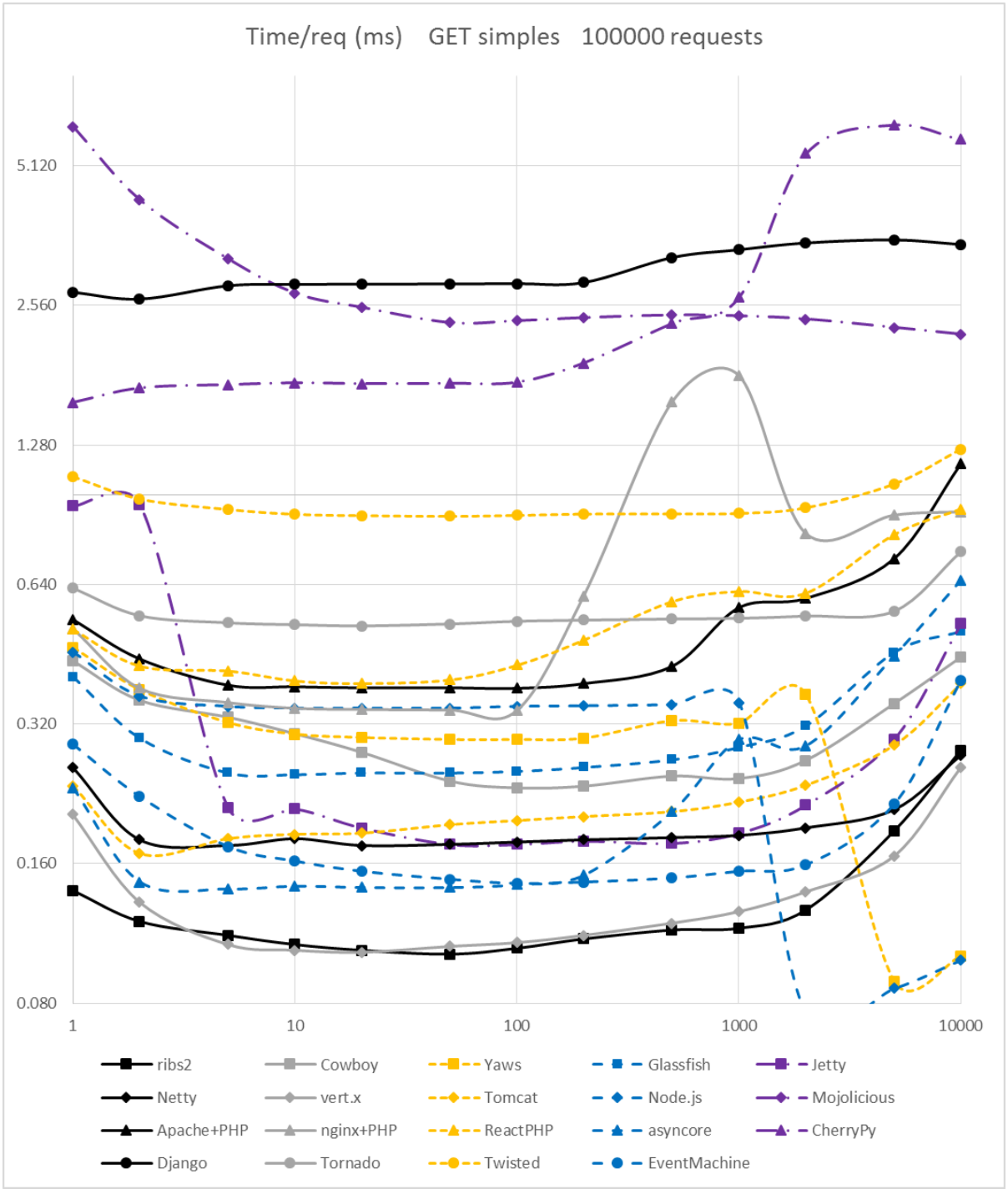



*Figura 10 - Resultado dos testes de desempenho às diferentes frameworks*

Tal como nos tempos, também foi possível obter os mesmos dados referentes às falhas de acesso ao serviço em situações de stress da máquina. Como se pode ver pela Tabela 11, as melhores frameworks foram Netty e Tomcat, seguidas de vert.x e até certo ponto, Node.js. O mesmo se pode verificar no gráfico da Figura 11, juntamente com o facto de nas falhas, as flutuações serem muito mais raras.

| | **Pedidos falhados GET simples 100000 requests** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Pedidos concorrentes:** | | | | | | | | | | | | |
| **Nome** | **1** | **2** | **5** | **10** | **20** | **50** | **100** | **200** | **500** | **1000** | **2000** | **5000** | **10000** |
| ribs2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 351 | 371 |
| Cowboy | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 46 | 285 | 1439 | 3698 |
| Yaws | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 219 | 382 | 200261 | 200000 |
| Glassfish | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 132 | 1086 | 1296 | 1298 | 5023 |
| Jetty | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 142 | 0 | 0 | 413 | 9645 |
| Netty | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| vert.x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 311 | 2390 |
| Tomcat | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Node.js | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 200124 | 200679 | 201542 |
| Mojolicious | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 660 | 3453 | 9089 | 24903 | 41701 |
| Apache+PHP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 528 | 2291 | 7052 | 12806 |
| nginx+PHP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 806 | 91134 | 30732 | 31264 | 42868 |
| ReactPHP | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 300 | 1067 | 2608 | 4627 | 11078 | 21605 |
| asyncore | 0 | 0 | 0 | 0 | 0 | 50 | 112 | 216 | 478 | 1263 | 2293 | 7115 | 17932 |
| CherryPy | 0 | 0 | 0 | 0 | 0 | 12 | 121 | 382 | 1801 | 4373 | 10262 | 37750 | 66388 |
| Django | 0 | 0 | 0 | 0 | 0 | 69 | 427 | 973 | 2129 | 3972 | 7653 | 71588 | 111728 |
| Tornado | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 177 | 2711 | 3826 |
| Twisted | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 179 | 789 | 1212 | 3157 | 10912 |
| EventMachine | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 51 | 56 | 2 | 0 | 2464 |

*Tabela 11 - Resultado dos testes às falhas de resposta*

| **Legenda** | |
|---|---|
| | Sem erros |
| | Erros apenas com alta concorrência |
| | Poucos erros com média ou alta concorrência |

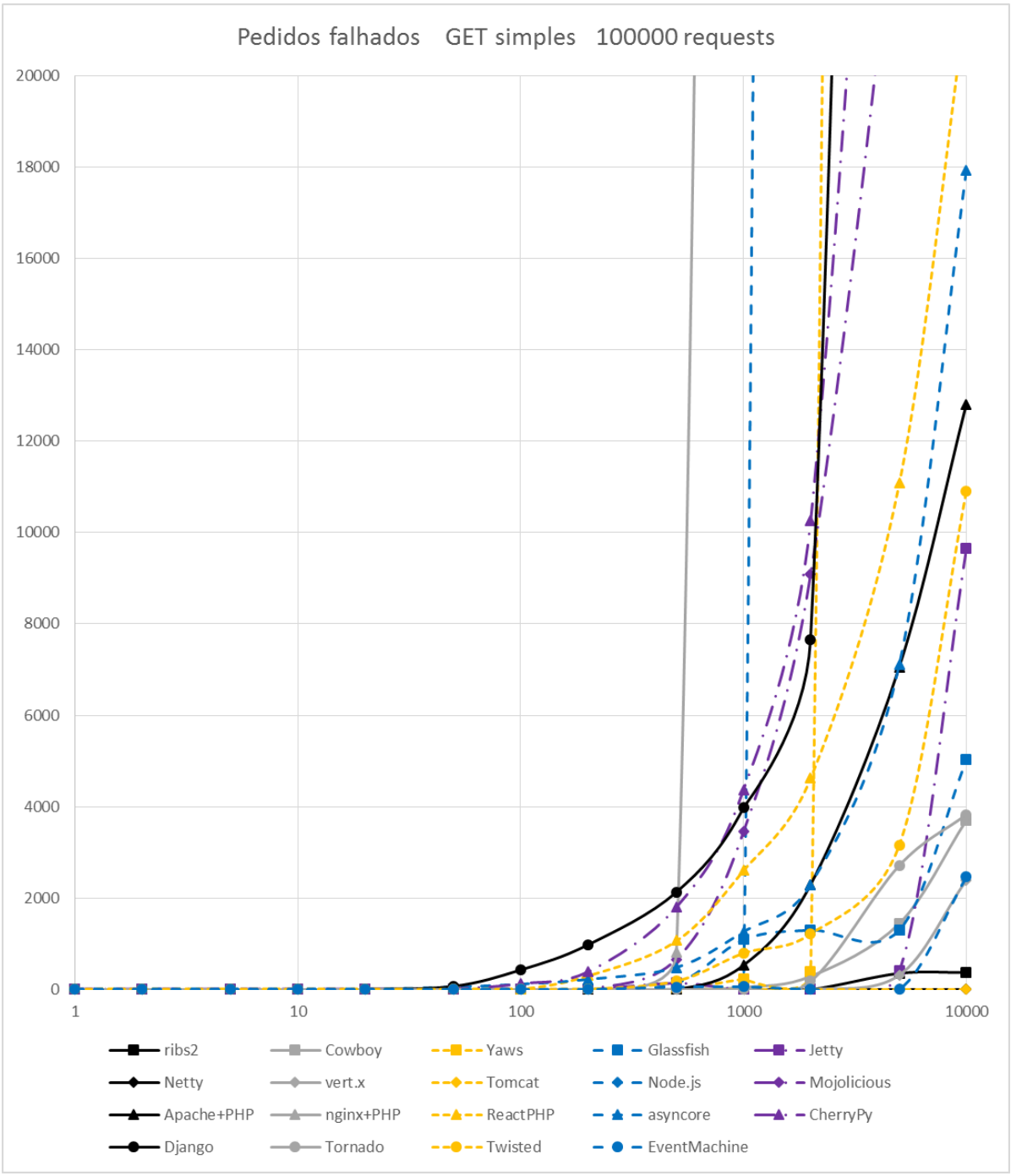



*Figura 11 - Resultados dos testes de fiabilidade às diferentes frameworks*

#### *4.1.3.3. Discussão*

Com base nos resultados dos testes efetuados, a nível de desempenho pode-se afirmar que as frameworks ribs2 e Vert.x são as melhores no que toca aos tempos de resposta, estando quase a nível de empate. De seguida (e com tempos de resposta superiores a 50%), estiveram as frameworks Netty, Tomcat, asyncore e EventMachine.

Quanto à fiabilidade, as frameworks Netty e Tomcat são as melhores posicionadas, tendo conseguido dar resposta a todos os pedidos efetuados. Imediatamente a seguir vem a

framework ribs2 com uma taxa de erro muito baixa, seguida da Vert.x e por fim da Node.js. Um pormenor a salientar é que a partir da barreira dos 2000 pedidos em simultâneo, o servidor em Node.js simplesmente termina, ficando o serviço indisponível.

Embora as frameworks ribs2, Vert.x, Netty e Tomcat tenham bastante qualidade, evidentemente teve que se aplicar algum processo de seleção. Optou-se pela Vert.x, pois assim iria:

- Permitir que tanto o servidor como o cliente fossem desenvolvidos na mesma linguagem (Java);
- De acordo com a documentação e exemplos vistos, iria simplificar o desenvolvimento do servidor;
- Permitiria adquirir novos conhecimentos numa nova framework e num método de desenvolvimento diferente do que já tinha sido experimentado.

### 4.1.4. Sistema de Gestão de Bases de dados

A mesma massificação que tem vindo a afetar as aplicações móveis e tecnologias web, tem vindo a influenciar de igual modo como os dados são guardados e geridos. Com a criação cada vez maior de informação e com uma necessidade de acesso rápido, em sistemas estáveis, distribuídos e replicados, não admira que esta seja uma das grandes áreas que sofreu e continuará a sofrer mais alterações ao longo dos anos.

Uma resposta a estas necessidades vem sobre a forma das bases de dados NoSQL, que tratando-se de uma alternativa às clássicas bases de dados relacionais, estas dividem-se em muitas subcategorias, cada uma com o seu domínio específico e com as suas vantagens e desvantagens para um dado problema. Estes DBMS têm aparecido em grande número nos últimos anos, o que tem motivado a diversos estudos nos últimos anos, quer por porte de investigadores independentes, quer por parte de grandes empresas, como forma de procurar soluções para os seus problemas e inovação para novos sistemas [22] [23] [24].

Como era de esperar, é recorrente encontrar comparativos entre os diversos DBMS, tal como [25] [26] [27] [28] [29] [30] [31] [32], ou até mesmo livros dedicados ao tema [33] [34]. Também é comum encontrar relatos de projetos que adotaram o caminho das DBMS NoSQL e que agora são um exemplo de sucesso [35], alguns deles vindos de empresas conceituadas, as quais fazem questão de afirmar que a adoção de uma base de dados NoSQL contribuiu para a melhoria global do sistema [36].

Para o desenvolvimento deste projeto, entendeu-se relevante fazer uma pesquisa e verificar se seria benéfico a utilização de alguma destas novas tecnologias, tanto para servidor como para cliente.

Muitos destes DBMS estão preparados para tratar grandes quantidades de dados, em máquinas de elevados recursos de processamento e memoria, algo que rapidamente descartou qualquer hipótese de vir a ser usado para o lado cliente. De facto a própria framework Android já dispõe de suporte a SQLite, a qual já deu provas de ter um desempenho excelente em ambientes com pouca concorrência de acesso e em dispositivos de recursos inferiores [37], motivos suficientes para a Google e muitos programadores recomendarem fortemente o seu uso.

Por outro lado, o servidor necessita de uma base de dados com suporte a bastante concorrência e que consiga aproveitar efetivamente os recursos disponíveis. Dado a vasta oferta de DBMS's, optou-se por refinar a busca dando maior relevância a bases de dados gratuitas, de livre utilização e que ofereçam boas prestações e fiabilidade em paralelo com a framework escolhida. Tratando-se este de um projeto académico, onde existe maior liberdade para efetuar testes e adquirir mais conhecimentos, optou-se também por aceitar o uso de tecnologias mais recentes, especificamente as bases de dados NoSQL, como forma de comprovar a sua eficiência e facilidade de aplicar num projeto.

#### *4.1.4.1. Tecnologias Disponíveis*

Após uma pesquisa em vários *websites,* fóruns*, blogs* e alguns livros dedicados ao tema, em especial [26] e [33], optou-se por escolher uma das seguintes bases de dados:

- Couchbase
- MongoDB
- Neo4J
- OrientDB
- Memcached
- Oracle NoSQL
- Redis
- Riak
- DB2
- MS SQL
- MySQL
- Oracle
- PostgreSQL
- Elasticsearch
- Solr
- Cassandra
- Hbase

Estes DBMS's serão analisados e comparados no seguimento desta seção, sempre no contexto em que se enquadra este projeto.

#### *4.1.4.2. Modelos de base de dados*

Apesar de uma base de dados servir principalmente para guardar informação, o modo como esta é guardada pode ser muito variado. Dependendo do modo como isto é feito, haverá consequências notáveis na forma como estes dados são enviados e recebidos pela base de dados, bem como a velocidade destas operações. Com isto, as bases de dados podem ser classificadas de acordo com os seguintes modelos:

- Relacional – O modelo clássico, onde é feita a normalização dos dados nas diversas tabelas, conseguindo uma redução do espaço ocupado e permitindo um tratamento de dados mais flexível. É um exemplo de ACID[30], nomenclatura dada a base de dados que garantem atomicidade, consistência, isolamento e durabilidade dos dados,

[30] http://dev.mysql.com/doc/refman/5.6/en/mysql-acid.html [Último acesso a 12 de Dezembro de 2014]

algo que apesar de ser excelente, dificulta o conceito de replicação sem perdas de performance.

- *Document* – Centrado no conceito de armazenar documentos, este modelo serve precisamente para guardar dados encapsulados de acordo com um *standard,* normalmente XML, JSON, BSON e PDF. A cada documento é associado um ID único em toda a base de dados e por vezes é possível fazer algum processamento extra, tais como ordenações e agregações.
- *Graph* – Usado para dados que possam ser relacionados entre si através de grafos, podem ser vistos como estruturas semelhantes a listas ligadas. Usadas sobretudo para guardar representações sociais e topologias de redes.
- *Key-Value* – Usando um *array* associativo, *map* ou dicionário, este modelo associa uma chave única a cada dado que se pretenda guardar. É dos modelos mais simples e rápidos, muitas vezes os dados são guardados em memória para permitir ainda mais performance. O acesso aos dados implica conhecer previamente a chave associada.
- *Search Engine* – Tirando proveito de estruturas binárias e *hashmaps*, este modelo pretende otimizar os dados para procuras rápidas, em detrimento da performance de inserções, modificações e eliminações.
- *Wide column* – Em parte semelhante ao modelo Key-Value, este modelo guarda a informação através de agrupamentos (similares a registos) de pares key-value (similares a colunas). A cada par key-value é adicionado um valor representativo do espaço temporal em que o valor foi modificado, para que os mecanismos de replicação possam identificar quais os valores mais recentes e os que precisam ser atualizados.

#### *4.1.4.3. Resultados Obtidos*

Recorrendo principalmente a pesquisas, livros e estudo dos diferentes modelos de bases de dados relacionais e não relacionais, foi possível extrair os dados que constam na Tabela 12.

| | | | | Performance sobre registos | |
|---|---|---|---|---|---|
| **Nome** | **Modelo** | **Open Source** | **Popularidade** | **Inserção** | **Procura** |
| Couchbase | Document | Sim | Média | Boa | Boa |
| MongoDB | Document | Sim | Alta | Boa | Boa |
| Neo4J | Graph | Sim | Média | Média | Má |
| OrientDB | Graph | Sim | Média | Média | Má |
| Memcached | Key-value | Sim | Média | Muito boa | Má/Não otimizado |
| Oracle NoSQL | Key-value | Não | Média | Muito boa | Má/Não otimizado |
| Redis | Key-value | Sim | Alta | Muito boa | Má/Não otimizado |
| Riak | Key-value | Sim | Baixa | Muito boa | Má/Não otimizado |
| DB2 | Relacional | Não | Baixa | Boa | Boa-Média |
| MS SQL | Relacional | Não | Alta | Boa | Boa-Média |
| MySQL | Relacional | Sim | Alta | Boa | Boa-Média |
| Oracle | Relacional | Não | Alta | Boa | Boa-Média |
| PostgreSQL | Relacional | Sim | Alta | Boa | Boa-Média |
| Elasticsearch | Search Eng. | Sim | Baixa | Má | Muito boa |
| Solr | Search Eng. | Sim | Baixa | Má | Muito boa |
| Cassandra | WideColumn | Sim | Alta | Boa | Boa |
| Hbase | WideColumn | Parcial | Alta | Boa | Boa |

*Tabela 12 - Resultado da pesquisa das bases de dados a usar*

| **Legenda** | |
|---|---|
| | Melhores resultados |
| | Bons resultados |
| | Resultados satisfatórios |
| | Maus resultados/A evitar |

Na escolha da base de dados a usar, optou-se pela seguinte ordem de critérios:

1. Excluir todas as bases de dados que não sejam de livre utilização
2. Análise de popularidade tendo em conta artigos recentes e pesquisas na Internet
3. Análise de desempenho espectável primeiro para armazenamento de registos (escrita) e de seguida para pesquisa de registos (procura)
4. Análise do enquadramento e prestação estimada com o equipamento que será usado para este projeto

Até ao ponto 3, mesmo não representando as soluções ideais, as bases de dados com melhor classificação eram:

- Cassandra
- MongoDB
- MySQL
- PostreSQL

Dentro das bases de dados NoSQL decidiu-se optar pelo MongoDB, pois mesmo com um desempenho inferior quando comparado com o Cassandra [25], este DBMS consegue tirar melhor rendimento a sistemas de recursos inferiores e onde o volume de dados não se enquadre somente dentro do conceito BigData [26]. Além disso, o MongoDB lida melhor com dados de estrutura diversificada, típico de software em processo de prototipagem e faz uso de interface JSON para troca de dados, tal como a framework escolhida para o Webservice.

No grupo das bases de dados relacionais, o MySQL é a mais bem colocada por ser um dos DBMS mais usados em alojamento e serviços *online* e por ter um desempenho ligeiramente superior em comparação com o PostgreSQL, em troca de um menor número de funcionalidades e não conformidade com ACID, algo que neste caso específico não apresenta ser problema [38].

Embora tanto MongoDB como MySQL aparentem ter capacidade de suportar este projeto, optou-se por dar prioridade ao MongoDB numa primeira fase e posteriormente adicionar suporte a MySQL para permitir tratamento adicional nos dados, esta decisão foi tomada especialmente por haver demasiados problemas nos drivers de acesso Vert.x – MySQL aquando a escrita deste relatório, não havendo garantias que fosse possível seguir esse caminho sem o risco de ocorrência de entraves maiores ao projeto.

## 4.2. Conclusões da Investigação

Com a investigação feita nesta fase foi possível aumentar os conhecimentos em diferentes áreas e aumentar o grau de confiança nas escolhas tomadas.

Apesar de não ser a escolhida, constatou-se que a biblioteca Scandit Barcode Scanner é uma boa alternativa à ZXing, oferecendo bom desempenho em *hardware* inferior. No entanto optou-se numa primeira fase por usar a ZXing, usando assim apenas tecnologias *open source* e aumentando o leque de código que podem ser reconhecidos.

Quanto aos protocolos de comunicação, o REST mostrou ser o mais apropriado para acesso a API's públicas e multiplataforma, a sua facilidade de implementação, baixo *overhead* de dados e fácil adaptabilidade a cada caso, foram os pontos mais importantes.

A nível de frameworks para criação de Webservices, foi possível ver e experimentar exemplos das suas implementações, nas mais variadas linguagens. Ribs2 e Vert.x apresentaram os melhores desempenhos, enquanto Tomcat e Netty apresentaram a maior fiabilidade. Optou-se por eleger a Vert.x como framework a utilizar, devido em parte ao seu alto desempenho e por ser uma tecnologia em bastante desenvolvimento e cada vez com mais suporte.

Por fim houve também um contacto com DBMS de diferentes tipos. Foi possível comparar as bases de dados relacionais e não relacionais, não só a nível de desempenho, mas inclusive o próprio processo de estruturação dos dados. Deste grupo acabou por se escolher o MongoDB e possivelmente numa fase posterior também o MySQL para facilitar possíveis tratamentos de dados.

# 5. Aplicação Desenvolvida

Dado que a criação de um sistema de catalogação e análise estatística em plataforma móvel era um dos objetivos primários deste projeto, achou-se que faria todo o sentido reservar este capítulo especificamente para este tema.

A maior parte do desenvolvimento foi no sentido de ir ao encontro das necessidades de uma instituição solidariedade específica, neste caso a Cáritas Diocesana de Coimbra, que desde o primeiro contacto, mostrou grande interesse em ideias de inovação, dando sugestões, solicitando demonstrações e dando feedback do encaminhamento global, tudo isto sem descartar a filosofia de um sistema de alto desempenho e alta disponibilidade para uso mais alargado.

Optou-se desde cedo uma abordagem ágil, pois só assim seria possível construir um sistema minimamente satisfatório para a instituição, tendo sido produzidos diversos artefacto, e um total de 5 releases. Foi ainda possível fazer alguns testes com servidores geograficamente separados, e embora para a última Release só se tenha usado uma topologia com um único servidor de Webservice e base de dados (Figura 12), é perfeitamente possível com um mínimo de configuração obter um sistema com múltiplos servidores de forma redundante e base de dados sincronizadas (Figura 13).

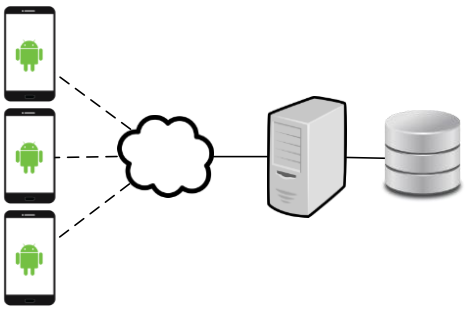


*Figura 12 - Topologia mínima*

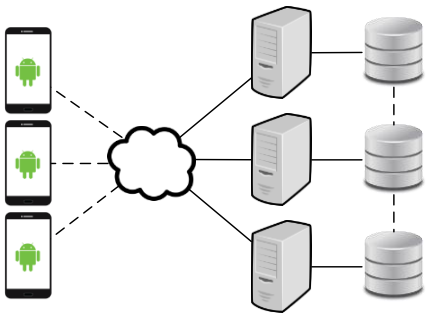


*Figura 13 - Topologia recomendada*

Embora tenham sido apresentadas várias releases, o nome da aplicação nunca foi oficialmente planeado, acabando por ficar simplesmente “Recolha de bens”, a qual será referenciada várias vezes ao longo deste capítulo.

## 5.1. Aplicações semelhantes

Antes do arranque da fase de desenvolvimento deste projeto, efetuou-se uma pesquisa por aplicações que seguissem uma ideologia semelhante. Embora não fosse possível encontrar nenhuma aplicação precisamente nos mesmos propósitos, foi possível agrupar as seguintes aplicações segundo a sua função principal de acordo com o apresentado na Tabela 13.

Dado que de todas estas nenhuma permite a catalogação centralizada, indicação das necessidades em tempo real, nem nenhuma é distribuída em código livre, conclui-se que este projeto certamente terá uma componente de inovação. A maioria destas aplicações tem também foco principal no continente americano, espera-se assim que com isto seja apresentada uma alternativa adaptada á realidade do continente europeu.

| Nome da aplicação | Recolha alimentar | Recolha de outros bens | Dados centrali-zados | Incentivo à competitividade do voluntariado | Compo-nente móvel | Estatísticas em tempo real | Registo espacial | *Open Source* |
|---|---|---|---|---|---|---|---|---|
| **Recolha de bens** | X | X | X | X | X | X | X | X |
| Foodbank [50] | X | | X | | X | | | |
| Wood Buffalo Food Bank [49] | X | | X | | X | | | |
| Roadrunner [48] | X | | | | X | | | |
| Donations [47] | | X | | | X | | | |
| Charitable Donations Log Lite [46] | | X | | | X | | | |
| Dell Employee Volunteer [45] | | X | X | X | X | | | |
| Food Bank [44] | X | | X | X | X | | | |
| St. Mary's Food Bank [43] | X | | X | X | X | | | |
| Justgive [42] | X | X | X | | | | | NA |
| Gofundme [41] | X | X | X | | | | | NA |
| Everyclick [40] | X | X | X | | | | | NA |
| Globalgiving [39] | X | X | X | | | | | NA |

*Tabela 13 - Comparação a outros projetos semelhantes*

## 5.2. Metodologias Aplicadas

Apesar da Cáritas Diocesana de Coimbra ter grande vontade de inovação, saber os problemas que enfrenta, saber o que precisa para os ultrapassar e ter um domínio acima da média nas tecnologias de informação, esta é uma instituição em que o desenvolvimento de aplicações informáticas claramente está fora do seu âmbito. Por outro lado, o mestrando não tem qualquer conhecimento da realidade e necessidades desta instituição. Com uma discrepância tão grande entre as duas partes, optou-se por uma metodologia ágil, iterativa e com uma boa componente de teste pela parte da instituição.

O método de trabalho aplicado resume-se ao apresentado na Figura 14.

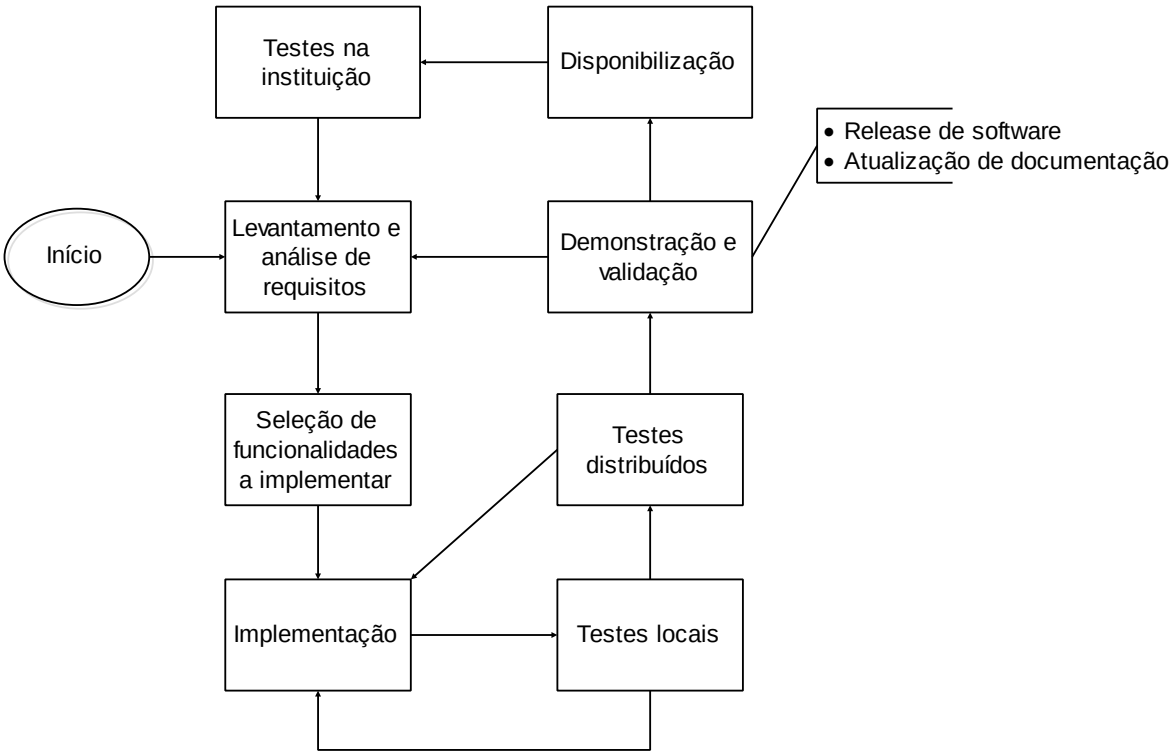



*Figura 14 - Esquema de aplicação das várias etapas de desenvolvimento*

Em resumos temos as seguintes fases:

- Levantamento e análise de requisitos
- Seleção de funcionalidades a implementar
- Implementação
- Testes locais
- Testes distribuídos
- Demonstração e validação
- Disponibilização
- Testes na instituição

Este processo será melhorado internamente e vai estender-se até à aplicação ser aprovada para uso e não tiver mais melhorias nem correções pendentes.

## 5.3. Ferramentas

Embora se tenha experimentado um número considerável de ferramentas, apenas uma pequena porção destas foi efetivamente utilizada, as quais serão descritas neste subcapítulo.

### 5.3.1. Eclipse

Atualmente mantido pela Eclipse Foundation, é uma IDE focada especialmente para Java, mas com suporte a um grande número de linguagens e frameworks. O seu sistema de *plugins* suporta adição de novas funcionalidades, automação de tarefas e integração com ferramentas em vista a facilitar e melhorar o trabalho entre equipas de desenvolvimento.

É também uma das IDE's suportadas pela Google para desenvolvimento Android, permitindo tanto programar, como construir a interface gráfica das aplicações.

O aspeto comum desta ferramenta pode ser vista na Figura 15.

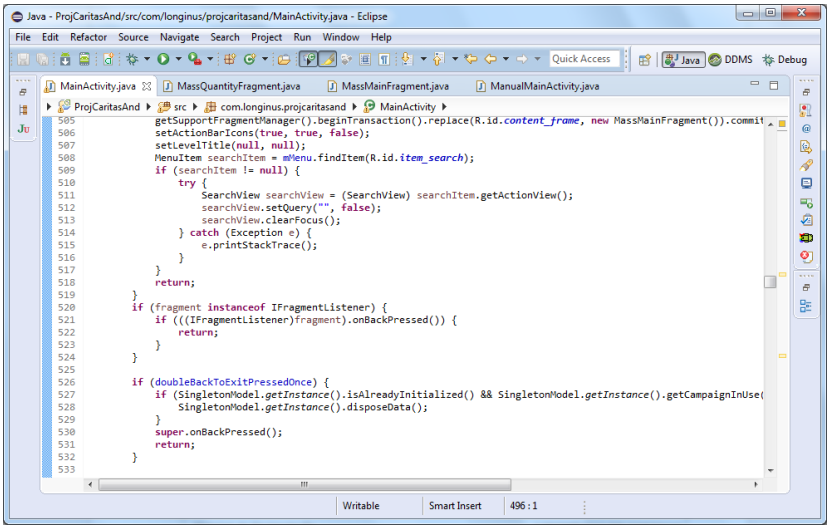



*Figura 15 - User Interface da IDE Eclipse*

### 5.3.2. Android SDK

Disponibiliza tudo que é necessário para o desenvolvimento de plataforma Android, nomeadamente bibliotecas, utilitários, emuladores, documentação e exemplos. Embora seja recomendado programar sempre para a última versão (opcionalmente com retro compatibilidade), é possível instalar apenas as bibliotecas, documentação, exemplos e imagem de emulador para uma versão e plataforma específica.

Esta ferramenta é também responsável pelos *updates* que a Google possa eventualmente lançar. O seu aspeto é semelhante ao apresentado na Figura 16, sendo possível escolher com mais pormenores os elementos que se pretende transferir e atualizar de cada versão.

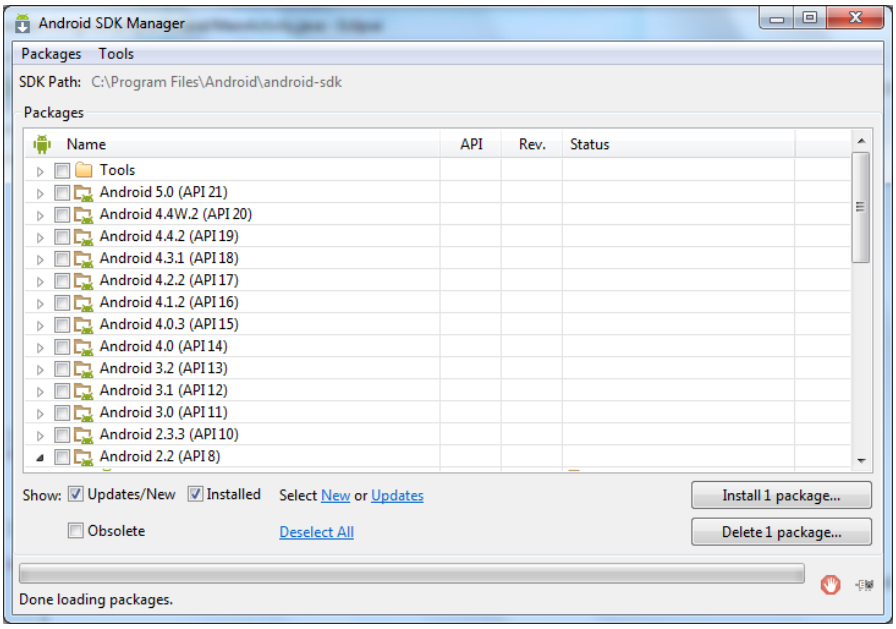



*Figura 16 - User Interface do Android SDK Manager*

Disponibilizando um grande conjunto de ferramentas, existe um conjunto de 16 ferramentas fundamentais que acompanham todas as instalações da SDK, as quais estão descritas na Tabela 14. Visto que as IDE's compatíveis já fazem uso destas mesmas ferramentas, raramente será necessário o programador preocupar-se em as utilizar manualmente.

| **Programa** | **Função** |
|---|---|
| Android | Permite gerir AVD's, projetos e componentes da SDK |
| ddms | Debugger de aplicações |
| dmtracedump | Cria diagramas de chamadas com base nos ficheiros de log |
| Draw 9-patch | Permite criar imagens NinePatch com um editor gráfico |
| emulator | Um emulador de Android, baseado em QEMU |
| hierarchyviewer | Permite fazer debug e ajudar a otimizar a parte visual |
| hprof-conv | Converte ficheiros HPROF em formatos standard para fácil analise |
| layoutopt | Analisa o layout de aplicações |
| mksdcard | Cria discos SD virtuais (usado pelo emulador) |
| Monkey | Ferramenta para teste de stress de eventos |
| monkeyrunner | API para comandar o Monkey externamente |
| ProGuard | Importante ferramenta de ofuscação e compactação de aplicações |
| Systrace | Permite analisar as tarefas e os recursos usados pelas mesmas |
| sqlite3 | Permite o acesso a ficheiros de base de dados de Android |
| traceview | Visualizador gráfico de logs |
| zipalign | Otimiza os .apk's através de alinhamento de dados |

*Tabela 14 - Lista de ferramentas típicas da SDK do Android*

### 5.3.3. RestClient

Durante o desenvolvimento de aplicações que comuniquem por processos ou mesmo por máquinas distintas, é de extrema importância ter ferramentas que permitam simular um dos extremos do elo de comunicação. Neste caso concreto usou-se esta ferramenta para simular um cliente REST, facilitando a deteção, isolamento e identificação de erros no lado do servidor.

A facilidade da escolha de métodos HTTP, preenchimento de corpo no envio de pedidos POST e inspecção da resposta, aliada ao facto de ser compatível com várias plataformas, fazem desta uma ferramenta muito boa para testar serviços REST. Uma representação do ecrã inicial desta ferramenta é visível na Figura 17.

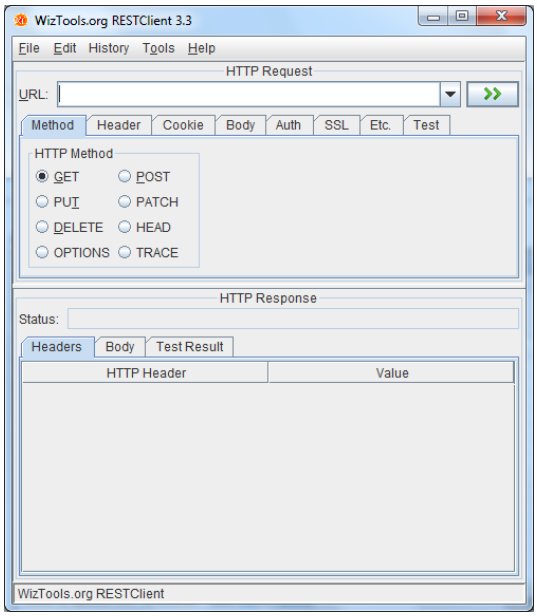



*Figura 17 - User Interface do RESTClient*

### 5.3.4. Adobe Photoshop

Um dos editores de imagem *raster* desenvolvido pela Adobe Systems, tendo sido lançado publicamente pela primeira vez em 1988. É atualmente um dos *softwares* de edição e tratamento de imagens mais utilizado em todo o mundo.

Apesar do tratamento de imagens ser a sua área de trabalho principal, as suas ferramentas de *brush*, *eraser, magic wand* e *crop*, aliadas a uma capacidade de exportação em múltiplos formatos otimizados para web, fazem deste um bom editor para prototipar, preparar e exportar elementos gráficos para posterior incorporação na parte visual de aplicações Android.

### 5.3.5. JustInMind Prototyper

É um *software* que permite criar *mockups* interativos de aplicações para Web, Android, iOS e Windows Phone. Foi uma ferramenta bastante importante para prototipar, definir o aspeto visual de toda a aplicação e debater outras possíveis soluções. Com isto obteve-se um

feedback antecipado, o desenvolvimento foi mais linear e evitaram-se tantas correções nos elementos gráficos.

Embora não tenha sido usado, esta ferramenta permite criação de *mockups* interativos remotamente, dando a hipótese de um cliente fazer a sua avaliação fora de reuniões, algo que será certamente uma mais-valia para empresas de desenvolvimento.

### 5.3.6. Brackets

Trata-se de um editor escrito em HTML, CSS e JavaScript focado especialmente no desenvolvimento Web. Foi criado inicialmente pela Adobe Systems e atualmente mantido como projeto *open-source*.

Dispor de suporte a *plugins* e sendo construído naquelas linguagens, permite a este editor ter uma grande oferta no que toca aos *plugins* desenvolvidos pela comunidade, desde elementos gráficos até *debuggers* JavaScript, mostrando-se uma boa ferramenta para desenvolvimento Web dentro do ecossistema open-source.

As suas capacidades de *auto-complete* e ligação ao Google Chrome para *debug* e *preview* em tempo real foram os pontos que tornaram esta a ferramenta de eleição para construir os elementos Web neste projeto, englobando todos os elementos apresentados em WebViews na aplicação Android e toda a área administrativa.

## 5.4. Tecnologias e Linguagens Usadas

Foi aplicada uma grande diversidade de tecnologias, padrões e outras ferramentas intimamente ligadas ao código. Embora não estritamente limitadas a estas, as principais estão descritas no decorrer deste subcapítulo.

### 5.4.1. Java

É uma linguagem de programação de uso geral, suporta execução concorrente, é orientada a classes e a objetos. Teve a sua primeira versão pública lançada em 1995 e desde aí tornou-se uma das linguagens de programação mais usada em todo o mundo.

Um dos aspetos que realça o seu sucesso é a sua filosofia WORA[31], que pretende que na prática seja possível correr a mesma aplicação em plataformas e sistemas diferentes sem necessidade de recompilar o código, algo conseguido através da sua compilação para *bytecode* e execução numa máquina virtual Java em vez de correr diretamente sobre o sistema operativo. Mesmo com um nível de virtualização, esta linguagem tem um desempenho muito bom, equiparável a várias linguagens de alto nível.

A sua difusão em computadores desktop, servidores e dispositivos móveis contribui para a existência de muitas bibliotecas, ferramentas, *snippets* de código e uma grande ajuda de comunidades em vários fóruns de discussão.

[31] http://javaindetail.com/2014/10/18/write-once-run-anywhere-wora/ [Últ. acesso a 12 de Dezembro de 2014]

### 5.4.2. XML

Trata-se de um padrão criado com o propósito de facilitar a troca de informação através da internet entre diferentes sistemas. É baseado em linguagem de marcações semelhantes ao HTML.

XML é também a base de muitos formatos conhecidos, tais como XHTML, DOCX e SVG, é facilmente adaptável a novos formatos, algo conseguido pelo facto de ser um formato facilmente percetível pelo ser humano e fácil de processar por máquinas.

Apesar de ter sido criado com vista a utilização em internet, este formato tornou-se bastante popular e é costume ser usado para outros fins, tais como definir o aspeto visual de dispositivos móveis.

Um exemplo deste formato pode ser visito na Figura 18 (Exemplo adaptado do artigo na Wikipedia[32]).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<receita nome="pão" preparo="5 minutos" cozedura="1 hora">
  <titulo>Pão simples</titulo>
  <ingredientes>
    <ingrediente quantidade="3">Farinha</ingrediente>
    <ingrediente quantidade="7">Fermento</ingrediente>
    <ingrediente quantidade="1.5" estado="morna">Água</ingrediente>
    <ingrediente quantidade="1">Sal</ingrediente>
  </ingredientes>
  <instrucoes>
    <passo>Misture tudo e dissolva bem.</passo>
    <passo>Cobrir com um pano e deixar em local morno.</passo>
    <passo>Misture novamente e leve ao forno.</passo>
  </instrucoes>
</receita>
```

*Figura 18 - Exemplo de dados XML*

### 5.4.3. JSON

Tratando-se de uma alternativa ao formato XML, o JSON tem precisamente o mesmo propósito de facilitar a troca de informação através da internet, a um custo inferior de recursos necessários para processamento e criação, igualmente fácil de compreender por seres humanos e menor *overhead* de informação.

Foi desenvolvida originalmente por Douglas Crockford, baseado na sintaxe de JavaScript, no entanto é suportado na maioria das linguagens atuais. A sua forma de organização de dados baseia-se em objetos do tipo chave-valor, sendo possível associar a uma chave um valor do tipo Number, String, Boolean, Array, Object e Null.

Em contraste com o exemplo dado para XML na Figura 18, o mesmo exemplo pode ser visto em formato JSON na Figura 19.

[32] http://pt.wikipedia.org/wiki/XML

```
{
  "nome": "pão",
  "prepara": "5 minutos",
  "cozedura" : "1 hora",
  "titulo" : "Pão simples",
  "ingredientes" : [
    {" quantidade": 3, "ingrediente": "Farinha"},
    {" quantidade": 7, "ingrediente": "Fermento"},
    {" quantidade": 1.5, "ingrediente": "Água", "estado": "morna"},
    {" quantidade": 1, "ingrediente": "Sal"}
  ],
  "instrucoes" : [
    "Misture tudo e dissolva bem.",
    "Cobrir com um pano e deixar em local morno.",
    "Misture novamente e leve ao forno."
  ]
}
```

*Figura 19 - Exemplo de dados JSON*

### 5.4.4. REST

Introduzido e definido como tese de doutoramento de Roy Fielding em 2000 [51], trata-se de um conjunto de regras para definir arquiteturas de acesso a dados remotos. É bastante usado em *Web Services* públicos, pois é de fácil implementação e não está dependente da plataforma.

Para um *Web Service* seguir o modelo REST é necessário que cumpra as seguintes regras:

- Seguir o modelo cliente-servidor, ficando os clientes isentos de guardar dados
- Não manter um estado entre ações, permitindo uma melhor atomicidade dos pedidos
- Ser possível de reter em cache dados que se mantenham inalterados
- Conter uma arquitetura em camadas, não sendo possível o cliente saber a que máquina se encontra ligado, facilitando o balanceamento de carga
- Possivelmente permitir a adição, remoção ou alterar funcionalidades sem que o serviço fique indisponível
- Manter uma interface uniformizada e bem desacoplada

Tirando partido de mecanismos já definidos no HTTP, o REST possibilita as 4 operações CRUD (Create, Read, Update e Delete) sobre dados remotos, as quais são definidas como um *standard* presente nos vários Webservices. Neste caso são implementadas usando os métodos HTTP da Tabela 15.

| Método | CRUD | Função |
|---|---|---|
| GET | READ | Lista uma coleção de recursos ou obtém dados de um membro da coleção |
| POST | CREATE | Adiciona um novo membro a uma coleção |
| PUT | UPDATE | Substitui ou atualiza uma coleção ou um membro específico de uma coleção |
| DELETE | DELETE | Apagar uma coleção ou um membro específico de uma coleção |

*Tabela 15 - Métodos HTTP associados aos métodos CRUD*

### 5.4.5. Programação orientada a eventos (Event-driven programming)

É um paradigma de programação, onde em vez de seguir um fluxo de eventos padrão, segue um fluxo de acordo com acontecimentos externos a que se chama de eventos.

Tem vindo a ser usada especialmente nas camadas de *software* responsáveis pela interface gráfica e em dispositivos embebidos para receber eventos de hardware. Mais recentemente também tem vindo a ser implementado em *software* de alto desempenho, já que com a sua natureza assíncrona, permite um crescimento de processamento em paralelo, uma melhor distribuição de tempo de processador pelos diferentes núcleos e se for bem aplicado, permite igualmente aliviar os tempos de espera de acesso a hardware e outros recursos normalmente de acesso mais demorado.

Apesar da grande aderência por parte de muitas frameworks, este encontra-se sujeito a várias críticas. Tal como é o caso das críticas feitas por Miro Samek [52] em que refere que este paradigma pode levar a um aumento do número de bugs pelos seguintes motivos:

- Aumento da complexidade das condições lógicas
- Cada ponto de decisão requer uma avaliação de expressões complexas
- A mudança entre diferentes modos requer a modificação de muitas variáveis, o que pode levar facilmente a inconsistências

Não obstante, este paradigma tem bastantes qualidades e na atualidade está implementado em muitas frameworks de código aberto e proprietário. Com a predominância de processadores capazes de cada vez mais tarefas em simultâneo e o aumento da importância da parte gráfica das aplicações, tudo aponta para que se mantenha em uso no desenvolvimento de *software* por muito tempo.

### 5.4.6. Padrão Reativo (Reactive pattern)

Tirando partido da programação orientada a eventos, este padrão descreve uma forma de propagação do fluxo do programa em cadeia, onde um evento pode levar ao lançamento de outros eventos em cadeia. Estes eventos costumam ter mensagens associadas, sendo uma das suas principais funções o transporte de dados.

Um exemplo deste padrão é visível em folhas de cálculo, onde alterar um valor de uma célula, pode levar à necessidade de recalcular o resultado de outras células, consequentemente outras células dependentes das mesmas serem recalculadas e assim por diante até todas as células terem sido atualizadas. Outro exemplo de bastante

importância é nas linguagens de descrição de *hardware* (HDL[33]), tais como Verilog[34], onde a definição de um componente se propaga ao restante circuito.

É notória uma semelhança entre este padrão reativo e o padrão observador [53], só sendo distinguível devido a este último manter as chamadas e dados em *stack*. Embora seja possível implementar o padrão reativo usando o padrão observador, o observador sofrerá de problemas de complexidade exponencial e crescimento de pilha de dados de forma abruta.

No caso das frameworks de alto desempenho e alta disponibilidade, este padrão é de extrema importância, pois permite atender um grande número de pedidos, sem que os limites causados pelos tempos de resposta de *hardware* constituam um entrave ao desempenho do sistema.

### 5.4.7. Vert.x

Vert.x é uma framework com suporte a várias linguagens de programação (poliglota), orientada a eventos, construída segundo o padrão reativo e assente na máquina virtual Java (JVM). Foi uma das frameworks estudadas na fase de investigação deste projeto e consequentemente foi a escolhida para usar como base para conceção e implementação deste projeto.

Nesta framework um conjunto de código executável normalmente compreendido num único ficheiro tem a designação de Verticle. Este pode ser escrito em Java, JavaScript, Ruby, Groovy, Python, Scala e Clojure [54].

Um módulo é um conjunto de um ou mais Verticles, construídos para um fim relacionado, que pode ser reutilizado para diferentes aplicações. É semelhante ao conceito de bibliotecas de *software*.

Uma instância de Vert.x é onde são executados os Verticles, podendo estar diversos Verticles em execução simultânea, no entanto cada instância Vert.x está associada a uma única instância de máquina virtual Java. Podem haver no entanto diversas instâncias a correr na mesma máquina, ou em máquinas diferentes na mesma rede, e estas serem configuradas para trabalhar em *cluster* e comunicarem entre si.

Verticles comunicam entre si através do EventBus, um canal de comunicação que suporta entrega de mensagens em *unicast* ou *multicast*, podendo comunicar diretamente com um cliente através de um *browser*, abstraindo por completo a localização física e as linguagens usadas.

Como forma de evitar acessos concorrentes e evitar assim possíveis deadlocks, esta framework foi desenhada segundo um modelo de concorrência simples, onde é usada uma única *thread* por instância. Um mecanismo, conhecido por ciclo de eventos, garante que os Verticles sejam executados à medida que os eventos são recebidos. Tendo isto em conta , é vital que se evite código bloqueante num Verticle (p. ex. `Thread.sleep()`,`Object.wait()`,`CountDownLatch.await()` ou loops de grande

[33] http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Hardware_description_language.html [Último acesso a 12 de Dezembro de 2014]

[34] http://www.verilog.com/ [Último acesso a 12 de Dezembro de 2014]

complexidade), caso contrário o ciclo de eventos ficará bloqueado e a instância do Verticle poderá ficar inoperacional.

Nos casos em que não é possível evitar código bloqueante, tais como acessos a base de dados por JDBC ou operações de processamento intensivo, existe a hipótese de se usar um Verticle Worker para o efeito. Este são semelhantes aos Verticles comuns, no entanto em vez de executar no mesmo ciclo de eventos da instância principal, são executados por uma *pool* de *threads* reservadas precisamente para o efeito. Com isto é possível que código bloqueante não afete a performance geral do sistema.

### 5.4.8. Gradle

Similar ao Apache Ant[35] ou Apache Maven[36], o Gradle é uma ferramenta de automação de construção de projetos de *software*, que utiliza uma linguagem baseada em Groovy[37] no lugar de uma configuração em formato XML. Foi desenhado especialmente para projetos modulares extremamente grandes, permitindo *builds* incrementais apenas dos módulos realmente necessários, em vez de seguir uma linha de dependências estrita.

### 5.4.9. SVN

Formalmente conhecida por Apache Subversion[38], é uma ferramenta Open Source bastante usada para controlo de versões e controlo de revisões, apontado como sucessor do antigo CVS.

Foi lançada em 2000 pela CollabNet e em 2009 aceite para o projeto Apache Incubator, tendo ficado à responsabilidade da comunidade para manutenção e desenvolvimento de funcionalidades. Apesar de já se tratar de uma ferramenta de certo modo antiga e de estar a cair em desuso em relação à nova ferramenta GIT[39], continua a ser uma ferramenta muito boa, simples de usar e suportada por um vasto leque de IDE's e utilitários associados, cumprindo bem a sua função.

## 5.5. Requisitos

No decorrer do projeto, foram levantados vários requisitos, alguns pressupostos desde o início, outros adicionados ou alterados durante o decorrer do mesmo. Como forma de sintaxe, este subcapítulo contém a lista não detalhada das novas funcionalidades introduzidas em cada release.

[35] http://ant.apache.org/ [Último acesso a 12 de Dezembro de 2014]
[36] http://maven.apache.org/ [Último acesso a 12 de Dezembro de 2014]
[37] http://beta.groovy-lang.org/ [Último acesso a 12 de Dezembro de 2014]
[38] https://subversion.apache.org/ [Último acesso a 12 de Dezembro de 2014]
[39] http://git-scm.com/ [Último acesso a 12 de Dezembro de 2014]

### 5.5.1. Release 1

Dado ser a primeira release, a sua função era dar uma ideia de como a aplicação pretendia comportar-se e incentivar a novas sugestões.

Pretendia-se que tivesse as seguintes funcionalidades:

- Interface básica com lista de produtos registados
- Inserção de produtos via código de barras (sem resolução de nome) e via introdução por categoria, peso e quantidade
- Leitura de código de barras através de Scandit
- Menus e ecrãs modelo de visualização de estatísticas, ajuda e outros ecrãs relevantes

### 5.5.2. Release 2

Esta release pretendia estender a release anterior, adicionando um real suporte a gestão de dados.

Definiu-se assim as seguintes funcionalidades:

- Inserção de produtos via código de barras (com resolução de nomes) e via introdução por categoria, peso e quantidade
- Primeira implementação funcional do servidor, com capacidade de registo de dispositivos e obtenção de dados da campanha e valores que se pretendem alcançar (designados “valores objetivo” ao longo deste projeto)
- Envio de registo para o servidor
- Criação de ecrã dinâmico com categoria, subcategorias de produtos e pesos por defeito, de acordo com uma fonte de dados externa (ficheiro JSON)

### 5.5.3. Release 3

Nesta release foi feita uma completa remodelação gráfica da aplicação Android, baseada nas guidelines oficiais [55], no livro [56] e em mockups previamente avaliados e aprovados (Ver anexo A1).

Com isto enumera-se as seguintes funcionalidades:

- Interface de aplicação cliente de acordo com as normas do Android 4
- Suporte à inserção agrupada de produtos da mesma categoria e peso para que os registos possam ser inseridos de forma automática e minimizar assim a tarefa de inserção de código de barras de produtos de forma manual
- Conversão dos diferentes ecrãs para fragmentos, para uma maior fluidez
- Estatísticas globais a serem obtidas e apresentadas de acordo com as informações transmitidas pelo servidor
- Lista similar a catálogo, apenas categorias e subcategorias de produtos, unicamente para fins de consulta
- Informações acerca da comunicação e últimas atualizações de dados

- Ecrã de apoio
- Opção de sair e parar qualquer thread de atualização em background
- Criação de ecrã dinâmico com categoria, subcategorias de produtos e pesos default, de acordo com uma fonte de dados externa (ficheiro local ou dados enviados pelo servidor)

### 5.5.4. Release 4

Esta release deu mais relevância à interface de administração, de forma a poder ser usada por alguém externo ao desenvolvimento. Concluiu-se também a lista de funcionalidades discutidas e apresentadas nos mockups (Ver anexo A0).

Enumera-se as seguintes funcionalidades:

- Interface de administração acessível via Web
- Possibilidade de gestão de campanhas, dispositivos e definição de valores pretendidos por campanha
- Exportação de dados referentes a produtos, estatísticas divididas em intervalos de tempo e registos em bruto para processamento exterior
- Interface na aplicação cliente para visualização de estatísticas pessoais
- Adição de um módulo de resolução de localização para situações em que não haja comunicação de dados

### 5.5.5. Release 5

Uma vez as funcionalidades principais já tendo sido implementadas, esta release focou-se em trazer melhorias a problemas encontrados durante o desenvolvimento, mais focadas na diversidade de dispositivos e no uso em localizações geográficas remotas, em que a comunicação de dados pode ser inexistente.

Concretizaram-se as seguintes funcionalidades:

- Melhor adaptação a dispositivos de diferentes dimensões
- Melhorias ao protocolo de comunicação, para minimizar os erros em situações de ligação de dados intermitente
- Adição de localização geográfica nos registos exportados para possibilitar estudos espaciais e estudos estatísticos sobre os mesmos

## 5.6. Desenho da Arquitetura

Tal como referido anteriormente, a fase de desenvolvimento deste projeto deu origem a 5 releases de software. Juntamente com estas foram produzidos vários artefactos, nomeadamente diagramas e tabelas que facilitaram a organização e conceção do código.

Ao longo deste subcapítulo vai-se descrever alguns destes artefactos, as suas mudanças e motivos que levaram a isso.

### 5.6.1. Release 1

Tratando-se da primeira release, a intensão principal seria mostrar uma ideia inicial, quer de interface quer de funcionalidades, para sucessivas melhorias.

Nos dispositivos testados até à altura (câmara menor que 2.1MP), a biblioteca de leitura de código de barras Scandit apresentou menores tempos e menor taxas de erros no reconhecimento e leitura de códigos de barra, motivo pelo qual foi usada nesta release.

Bases de dados locais ou remotas ainda não estavam presentes nesta etapa. Da mesma forma, todas as categorias e subcategorias criadas eram apenas provisórias e unicamente para efeitos de demonstração, não estando obrigatoriamente iguais à lista posteriormente fornecida pela Caritas.

**Diagrama de atividades**

Como ponto de partida, esta aplicação mostrava imediatamente um ecrã de recolha, com 2 botões que conduziam a um ecrã de introdução via código de barras e a um ecrã de introdução manual.

Escolhendo introdução manual, seria apresentado um ecrã para escolher a categoria do produto a adicionar (p. ex. Charcutaria ou Líquidos), em seguida a subcategoria (p. ex. fiambre ou leite), a unidade aproximada (p. ex. 200gr ou 1.5L) e por fim um ecrã para escolher a quantidade a adicionar. Os dados seriam enviados para a atividade principal.

Escolhendo introdução via código de barras, seria apresentado um ecrã de captura e qualquer código lido seria enviado para a atividade principal, com a quantidade padrão de 1.

Outra ação seria utilizando o menu na atividade principal para aceder às estatísticas, que no caso desta release apenas apresentava uma imagem ilustrativa, algo que seria melhor definido no futuro.

Este comportamento descrito é visível na Figura 20.

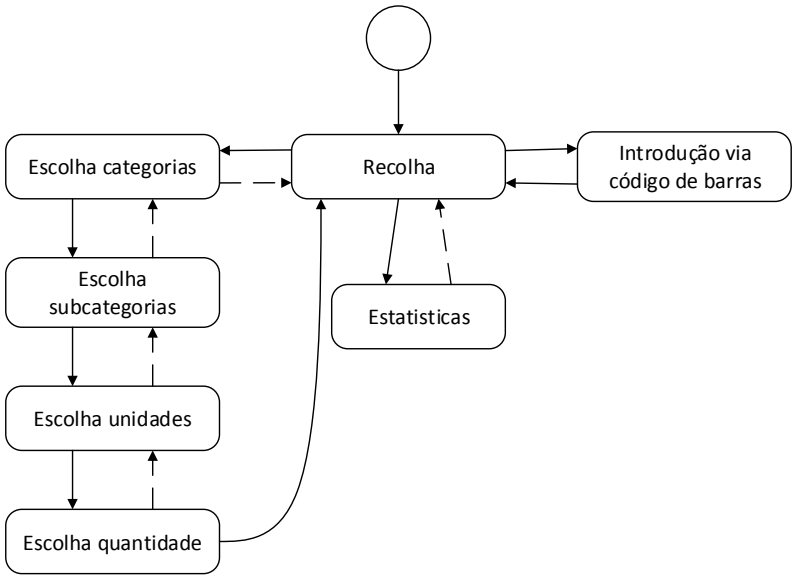



*Figura 20 - Diagrama de atividades da release 1*

### Diagrama de classes

Embora esta fosse a primeira release, já se teve algum cuidado nas decisões no que diz respeito à arquitetura do sistema. Como já se previa a existência de diversas atividades, tinha-se a noção que era necessário uma forma fácil para receber dados das mesmas e apenas seria necessário uma instância do mecanismo que recebia e geria os dados, optou-se pelo padrão de *design* de software "Singleton". Tal como o nome sugere, em todo a aplicação só existe uma instancia única, ficando esta responsável de instanciar também os objetos de acesso à base de dados e lista de campanhas, recebendo dados e pedidos da atividade de recolha para mostrar itens registados e das atividades de introdução manual e via código de barras, algo que está esquematizado no diagrama de classes da Figura 21.

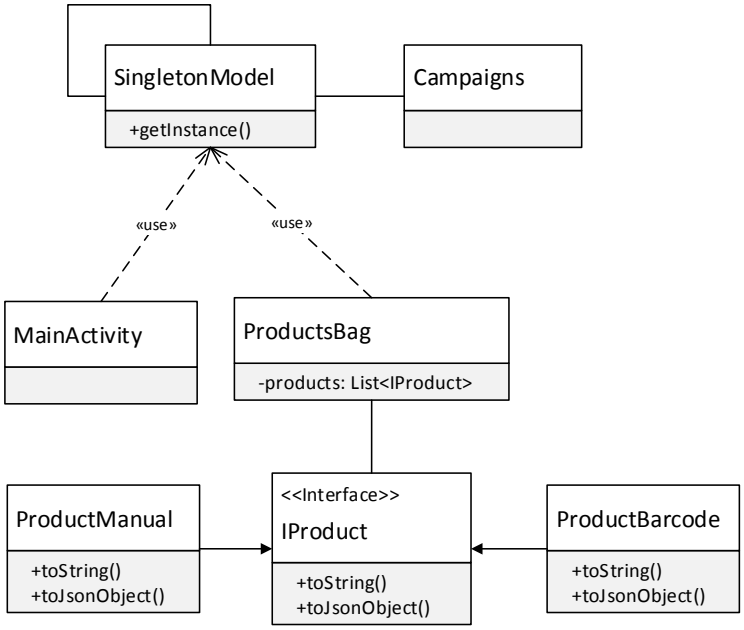



*Figura 21 - Diagrama de classes da release 1*

### 5.6.2. Release 2

Após a primeira release, começou-se o desenvolvimento da parte servidor e a aplicação foi trabalhada de forma a guardar dados eficientemente, quer nos dispositivos móveis, quer no servidor. Com isto obteve-se a release 2, com um conjunto de alterações que serão descritas.

**Diagrama de classes**

Aproveitando o modelo já desenhado para a release 1, adicionou-se uma classe para gerir os dados provenientes ou destinados ao servidor. Esta faz uso de outra classe igualmente criada para efetuar pedidos GET e POST ao servidor e gerir a transmissão de dados ao nível da ligação. Também se adicionou uma classe para gerir e comunicar com a base de dados local, dando maior confiança quanto à integridade dos dados e permitindo a sua persistência entre arranques da aplicação. Estes acréscimos podem ser vistos na Figura 22.

Com esta pequena alteração já foi possível enviar uma lista de produtos para um servidor remoto.

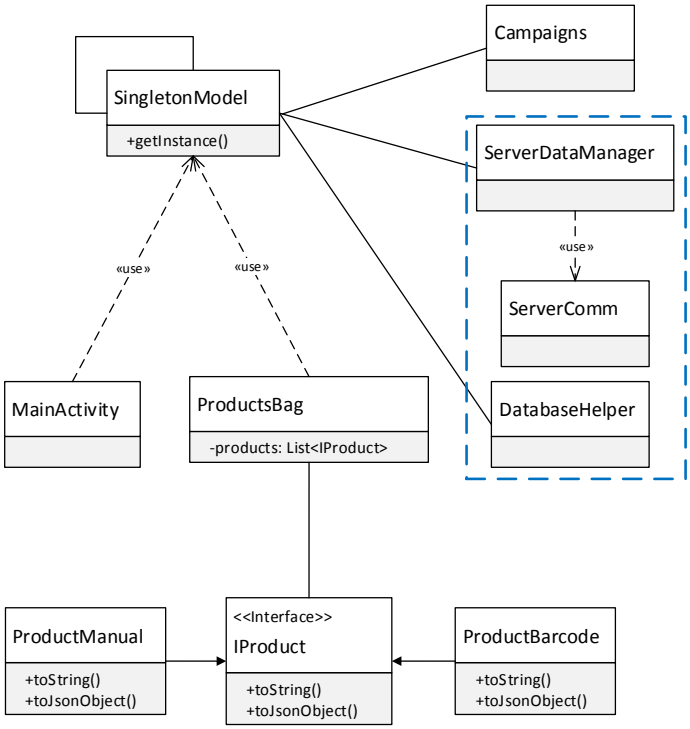



*Figura 22 - Diagrama de classes da release 2*

### Base de dados SQLite (Dispositivos móveis)

Com a inclusão de comunicação remota e de uma base de dados local, houve a necessidade de criar uma estrutura de base de dados.

A base de dados incluída na aplicação móvel foi implementada utilizando SQLite, já presente na própria framework Android. Embora não disponha de uma grande variedade de tipos de dados nem permitam relacionamento de entidades, esta desde o início mostrou um desempenho excelente em pesquisas e inserções de dados, permitindo resultados imediatos do ponto de vista de perceção humana, mesmo com milhares de registos.

Apesar de haver funcionalidades que só viriam a ser implementadas em releases futuras, optou-se por criar um conjunto de tabelas (Figura 23), juntamente com as respetivas função de inserir, editar, procurar e apagar dados, nomeadamente:

- *Campaings* – Lista de campanhas registadas no servidor e colocadas como visíveis
- *Products* – Dados de produtos (e o respetivo EAN) para lookup de produtos local
- *Names* – Representação textual das categorias, para facilitar a criação do gráfico de estatísticas
- *Objectives* – Valores que se pretendem atingir de cada categoria ou subcategoria
- *Statistics* – Valores já atingidos de cada categoria ou subcategoria, de acordo com o valor reportado pelo servidor
- *Types* – Representação textual dos códigos usados para designar categorias e subcategorias de bens
- *Registries* – Presentação de uma sessão de registos (vários registos de produtos, de tipos diferentes, enviados em simultâneo)
- *Registries*_man – Registos de produtos feitos através da introdução manual
- *Registries*_auto – Registos de produtos feitos através de introdução automática, por meio da leitura de código de barras


*Figura 23 - Estrutura da base de dados SQLite da release 2*

### Base de dados MongoDB (Servidor)

A criação de um servidor REST para receção de registos dos dispositivos móveis levou à evidente necessidade de dispor de uma base de dados para guardar dados recebidos e disponibilizar informação necessária.

Criou-se um conjunto de collections base para ajudar a testar a real comunicação aplicação-servidor, as quais estão descritas na Tabela 16 e detalhadas nas tabelas seguintes.

| **Collection** | **Função** |
|---|---|
| campaigns | Campanhas introduzidas no sistema |
| devices | Dispositivos registados no sistema |
| locations | Registo de localizações enviadas periodicamente por cada dispositivo |
| products | EAN, tipo e peso dos produtos reconhecidos |
| registries | Registos de doações |
| types | Tipos de produtos e a sua designação |
| deviceCampaigns | Associação entre dispositivos e campanhas |

*Tabela 16 - Collections da base de dados MongoDB na Release 2*

| **campaigns** | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idCampaign | String | Identificador único da campanha |
| name | String | Nome da campanha a apresentar |
| password | String | Password de inscrição |
| warehouse | String | Armazém de destino dos produtos |
| dateStart | Int64 | Data de início da campanha em UNIX Timestamp (milissegundos) |
| dateEnd | Int64 | Data de fim da campanha em UNIX Timestamp (milissegundos) |

*Tabela 17 - Campos presentes na collection campaigns*

| **devices** | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idDevice | String | Identificador único de dispositivo |
| isBlocked | Boolean | Nome da campanha a apresentar |
| ismobile | Boolean | Password de inscrição |

*Tabela 18 - Campos presentes na collection devices*

| products | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| ean | String | Identificador do código EAN do produto |
| type | String | Indicador to tipo de produto |
| weight | Int32 | Peso ou volume médio do produto, de acordo com as unidades presentes em unit |
| unit | String | Unidade em que é medido o produto |
| name | String | Nome dado ao produto, a aparecer nos dispositivos |

*Tabela 19 - Campos presentes na collection products*

| registries | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idCampaign | String | Identificador único da campanha |
| idDevice | String | Identificador único de dispositivo |
| date | Int64 | Data em UNIX Timestamp (milissegundos) do momento em que o registo foi finalizado no dispositivo |
| products | Array | Array de dados do tipo definido na Tabela 21 |

*Tabela 20 - Campos presentes na collection registries*

| registries->products | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| ean* | String | Identificador do código EAN do produto |
| quantity | Int32 | Quantidade de produtos específicos registados |
| type | String | Indicador to tipo de produto |
| weight | Int32 | Peso ou volume médio do produto, de acordo com as unidades presentes em unit |
| unit | String | Unidade em que é medido o produto |

*Tabela 21 - Modelo dos campos presentes em cada estrutura do array products*

*Campo não obrigatório

| types | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| symbol | String | Símbolo identificador alfanumérico associado à categoria |
| name | String | Nome associado à categoria |
| subtypes | Array | Array de dados do tipo definido na Tabela 23 |

*Tabela 22 - Modelo dos campos presentes na collection types*

| types->subtypes | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| id | Int32 | Identificador de subtipo de categoria |
| name | String | Nome associado ao subtipo de categoria |

*Tabela 23 - Modelo dos campos presentes em cada estrutura do array subtypes*

| deviceCampaigns | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idDevice | String | Identificador único de dispositivo |
| idCampaign | String | Identificador único da campanha |
| date | Int64 | Data em que o dispositivo se registou na campanha em UNIX Timestamp (milissegundos) |

*Tabela 24 - Modelo dos campos presentes na collection deviceCampaigns*

### Interface REST

As funcionalidades disponibilizadas pela interface REST ficaram restritas às funcionalidades já devidamente implementadas, quer pelo código do servidor, quer pela estrutura da base de dados.

Em suma, as chamadas aos métodos poderiam ser feitas de acordo com os esquemas de URL presentes na Tabela 25.

| URL | Método | Descrição |
|---|---|---|
| http://server/folder/device | GET | Obtémo estado do dispositivo (p. ex. Registado, bloqueado) |
| http://server/folder/device | POST | Regista dispositivo no sistema |
| http://server/folder/location | POST | Adiciona um conjunto de localizações onde o dispositivo esteve a recolher |
| http://server/folder/campaign | GET | Obtém a lista de campanhas visíveis |
| http://server/folder/campaign | POST | Regista o dispositivo a uma campanha |
| http://server/folder/product | GET | Obtém um conjunto de produtos |
| http://server/folder/registry | POST | Adiciona um conjunto de registos para processamento |
| http://server/folder/type | GET | Obtém a lista de tipos de bens |

*Tabela 25 - Lista de resources disponíveis na interface REST na Release 2*

### Envio de dados

Uma vez que a aplicação cliente corre em dispositivos móveis, os quais podem estar em localizações geográficas remotas, muitas vezes sem comunicação de dados, é completamente impensável fazer uma aplicação que requeira um canal constante de comunicação de dados. Mesmo em zonas com dados móveis, estes podem estar sujeitos a custos adicionais, o que obviamente se pretende evitar.

Por outro lado os dados vão sendo adicionados à aplicação, e mais cedo ou mais tarde, precisam de ser enviados para o servidor. Embora esta ação possa ser feita através de uma indicação manual, não é correto colocar a responsabilidade desta ação no utilizador final, que muito provavelmente estará a usar a aplicação como um utilitário da ação de campanha e desconhece por completo estes pormenores.

Como ponto de equilíbrio, optou-se pela abordagem presente na Figura 24, sendo esta válida tanto para dados móveis como para comunicação *wireless*.

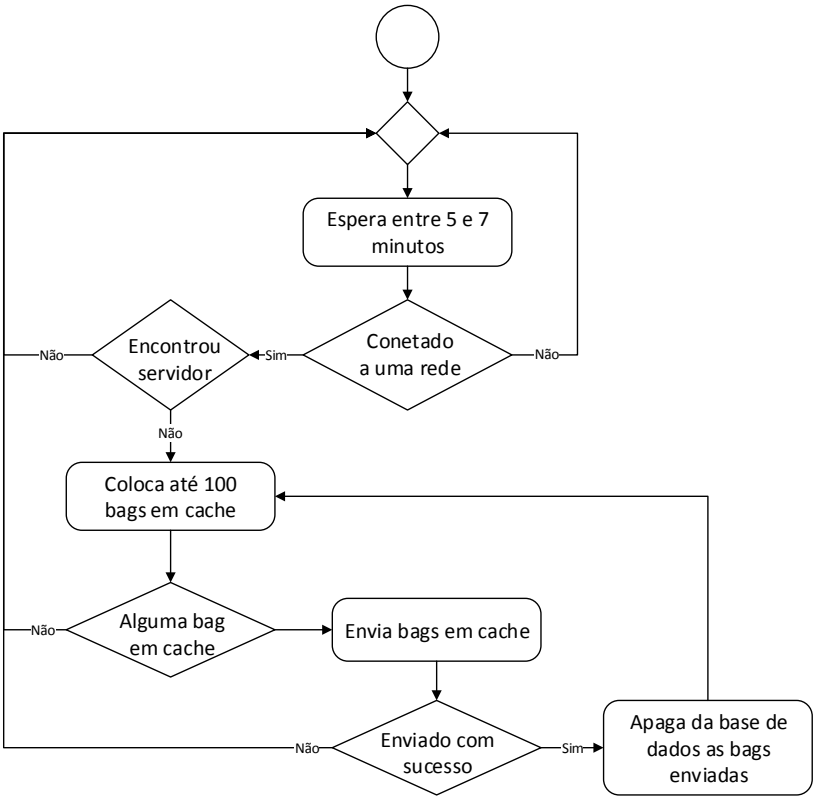



*Figura 24 - Sequência de passos para envio de dados na release 2*

### Armazenamento de registos

O armazenamento dos registos no servidor é um processo dividido em 3 etapas:

1. Receção dos registos pelo servidor REST e sucessivo despacho para o Verticle de Registos
2. Receção pelo Verticle de Registos para conversão dos registos em JSON para objetos dos tipo Registry e despacho para o Worker (Figura 25)
3. Receção pelo Worker, resolução do tipo de produto baseado no EAN e armazenamento na collection registries (Figura 25)

Embora o primeiro passo seja comum a todos os Verticles, o segundo e terceiro foram uma solução para fazer a resolução de um número considerável, de dados de forma que não provocasse o abrandamento do ciclo de eventos no caso de se receber um grande número de registos. Este problema certamente não seria tão obvio até aqui, mas seria evidente que com a evolução do sistema, seria uma boa prática.

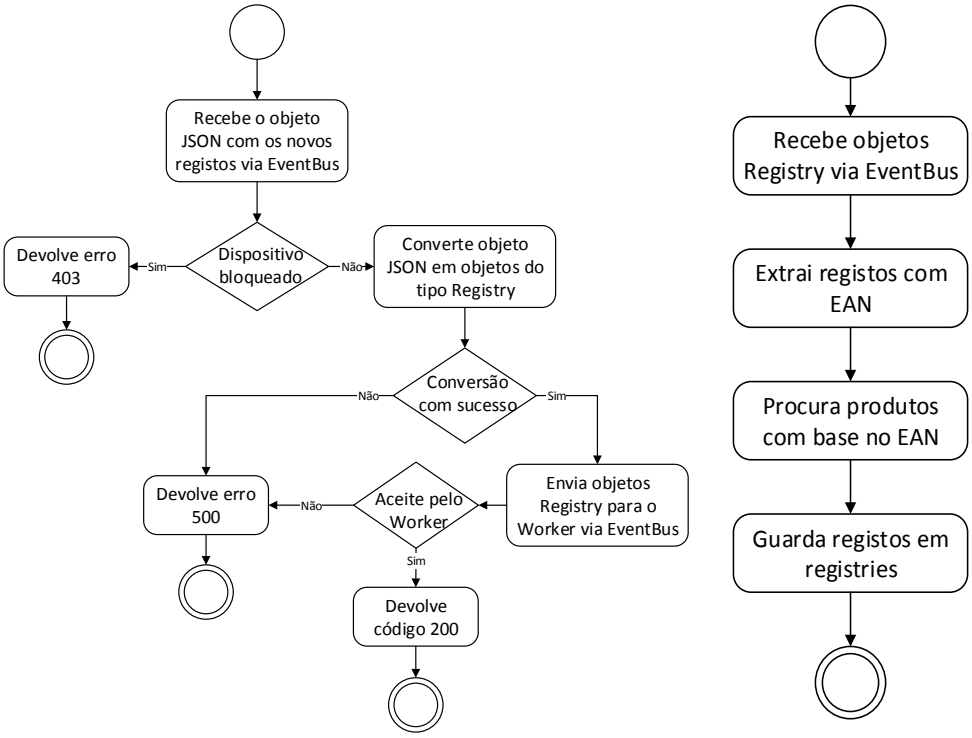



*Figura 25 - Diagrama de receção e armazenamento de registos*

### Estrutura do servidor

Mesmo se tratando da primeira release do servidor, este já dispôs dos elementos que desempenhavam as funções mais importantes e que o acompanharam até à última release.

A solução escolhida foi criar um Verticle que implementa um servidor REST, usando a biblioteca de WebServer do próprio Vert.x. Estes pedidos são enviados pelo EventBus em unicast para um Verticle Handler especializado, conforme o serviço que se tenha requisitado. Estes Verticles podem comunicar entre si para desempenhar funções extra (p. ex. verificar se o dispositivo que está a pedir estatísticas de uma campanha se encontra

bloqueado) ou simplesmente enviar ou requerer dados ao módulo de comunicação com a base de dados MongoDB.

Um caso especial acontece com o Verticle Handler que recebe os registos efetuados, em que devido a tarefas mais complexas (algumas de complexidade $O(n^3)$) e com bastante pesquisa de dados, tais como a resolução de categoria e peso dos produtos recebidos, incremento por categoria e no futuro a adicionação de novos produtos através dos dispositivos móveis, foi fundamental fazer alterações para que estas operações não abrandassem o ciclo de eventos da thread principal. Por este motivo, o Verticle que recebe os registos faz uma conversão dos registos convertidos para objetos, envia o pedido para um worker e assim que for aceite, informa o dispositivo móvel que foi recebido com sucesso. Este worker por sua vez irá processar todos os registos recebidos numa thread em separado, mantendo o desempenho e capacidade de resposta do ciclo de eventos da thread principal.

Um esquema desta estrutura pode ser vista na Figura 26.

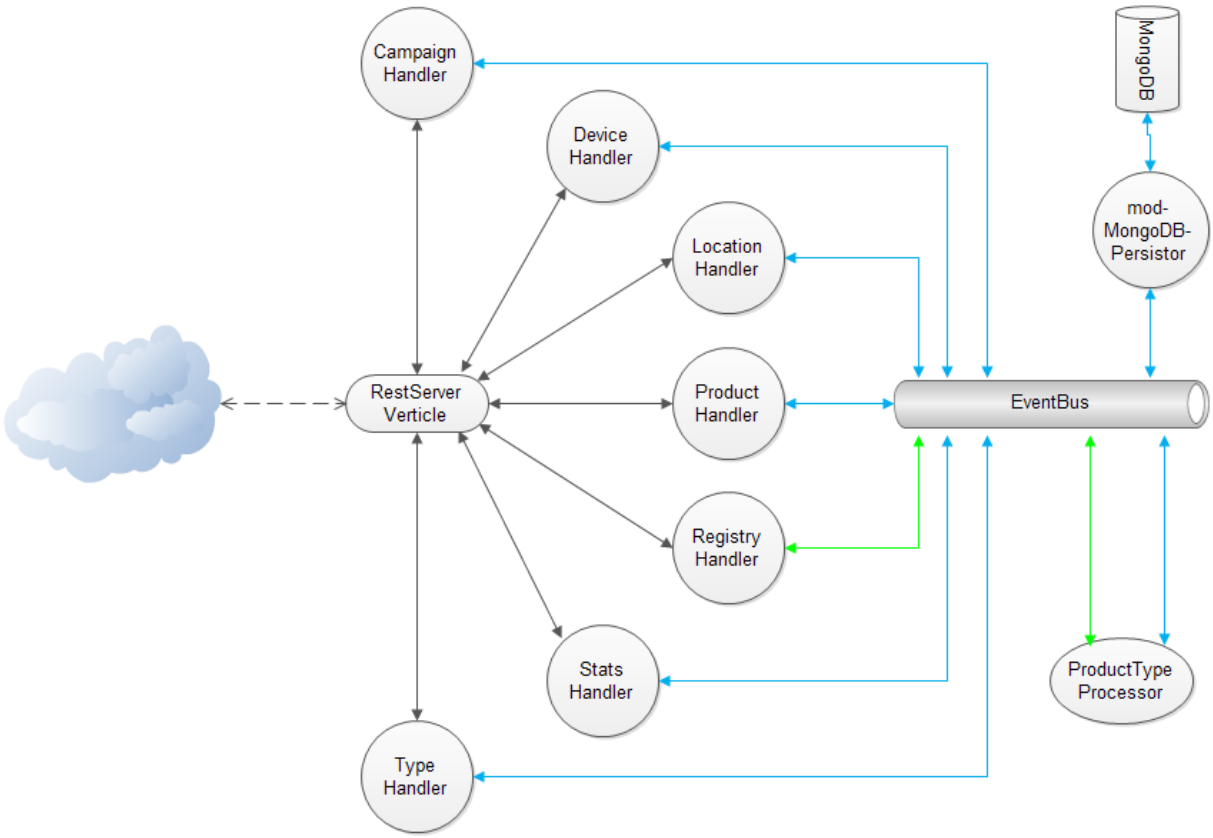



*Figura 26 - Estrutura do servidor na release 2*

### 5.6.3. Release 3

Esta release veio causar um grande impacto no desenvolvimento de toda a aplicação, com uma User Interface do lado cliente completamente remodelada, cálculo das estatísticas em tempo real, maior leque de serviços REST e acima de tudo, a capacidade de adicionar produtos do lado cliente através da introdução agrupada de produtos com características semelhantes.

**Diagrama de atividades**

Com esta release, o modelo de atividades foi completamente modificado, abandonando-se o uso de atividades para cada ecrã e o uso de um menu tradicional para locomoção entre estas, para um modelo com um numero reduzido de atividades, um grande numero de fragmentos reutilizáveis e a incorporação de um menu lateral, ao estilo do que é comum encontrar em aplicações para Android 3.0+.

Na prática a nova aplicação dispões apenas de 3 atividades:

- A atividade principal onde se começa por escolher a campanha a usar e por fim têm-se acesso ao ecrã de recolha e ao menu lateral para abrir outras secções do programa
- A atividade de leitura de código de barras
- A atividade de introdução manual de produtos, em que a própria recorre a fragmentos sucessivos

Foi também adicionado uma opção de sair para fechar a aplicação e obrigar qualquer thread em background a parar e libertar recursos de rede.

Um diagrama disto pode ser visto na Figura 27.

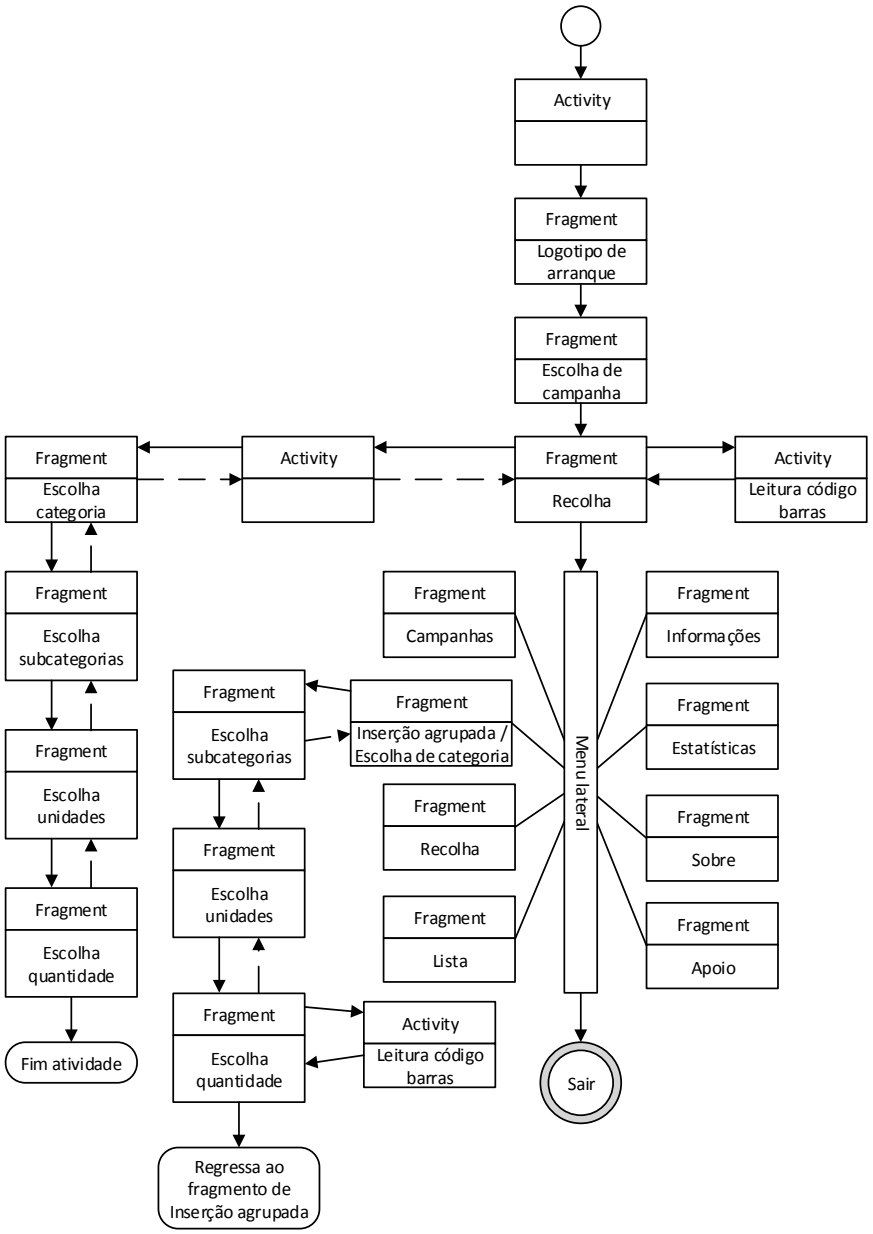



*Figura 27 - Diagrama de atividades da release 3*

### Diagrama de classes

O uso de fragmentos em vez de atividades requereu a construção de um mecanismo de comunicação bidirecional atividade-fragmento, através da criação das interfaces IMessages (Fragmento->Atividade) e IFragmentListener (Atividade->Fragmento). Fragmentos que requeiram barra de procura, alteração do comportamento padrão de *hardware keys* e receção de avisos atualização de dados, deverão obrigatoriamente implementar IFragmentListener.

Da mesma forma, a introdução de inserções agrupadas veio alterar a forma como os produtos são inseridos. Todos os produtos registados são introduzidos numa classe ProductsBag, algo que simboliza um conjunto de sacos doados. Nesta ProductsBag podem ser introduzidos produtos em caso de inserção normal ou será criado um MassProductContainer, em caso de inserção agrupada. Este além de guardar os produtos de código de barras, também guarda a indicação da categoria, subcategoria, unidades e faixa de peso do produto.

Estas alterações podem ser vistas na Figura 28 e Figura 29.

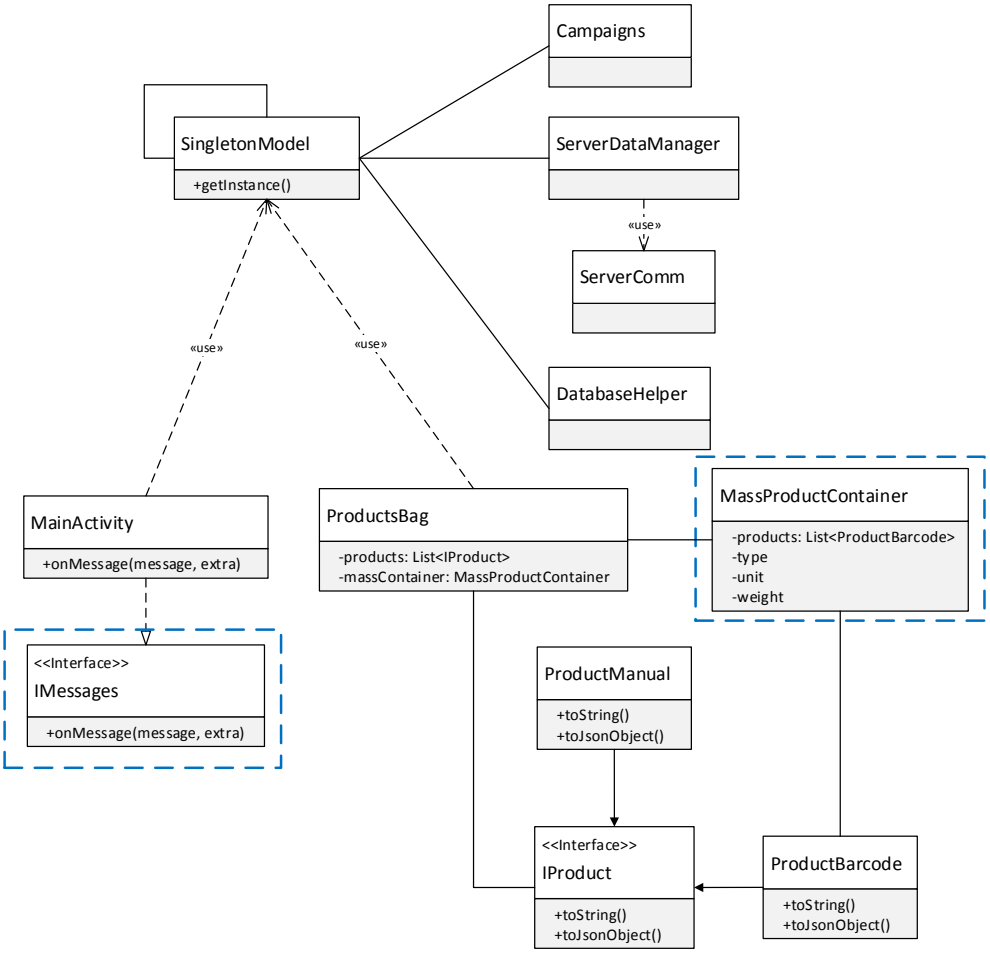



*Figura 28 - Diagrama de classes da release 3 (principal)*

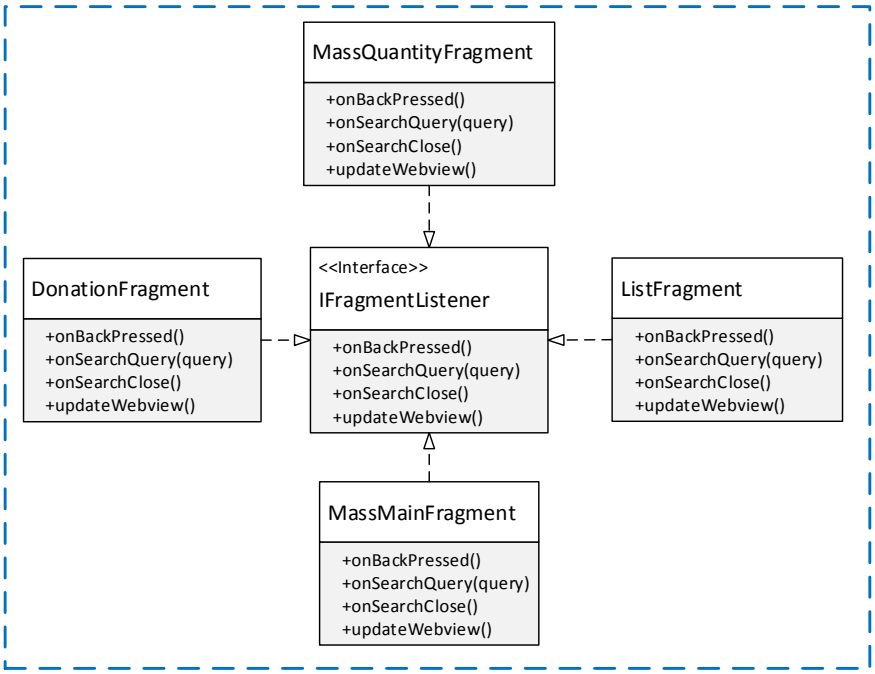



*Figura 29 - Diagrama de classes da release 3 (Fragmentos)*

### Base de dados SQLite (Dispositivos móveis)

Uma novidade desta release foi a captura de localização via GPS à medida que se iam registando produtos, para posterior submissão ao servidor e permitir uma análise estatística mais trabalhada.

Outra adição importante foi a introdução de uma tabela de definições genéricas, capaz de guardar dados semelhantes a um sistema chave-valor. Foi pensado especialmente para guardar a data de ultima atualização da lista de produtos, para permitir assim que a aplicação móvel informe o servidor e se decida se deve enviar a lista de novos produtos entretanto introduzidos no sistema.

Para a inserção múltipla foi possível usar as tabelas registries_auto, registries_man e registries, usando registries_man com quantity 0 para identificar a categoria e intervalo de peso e registries_auto para guardar os EAN's dos produtos registados. Prática esta que evidentemente veio a ser modificada na release seguinte.

Em resumo, as tabelas adicionadas foram:

- *Locations* – Lista de localizações GPS e o momento da sua receção
- *Settings* – Definições de uso geral (atualmente apenas contem a data da ultima atualização de produtos, podendo ter novas utilidades no futuro)

Com estas alterações, a nova base de dados ficou com o aspeto da Figura 30.

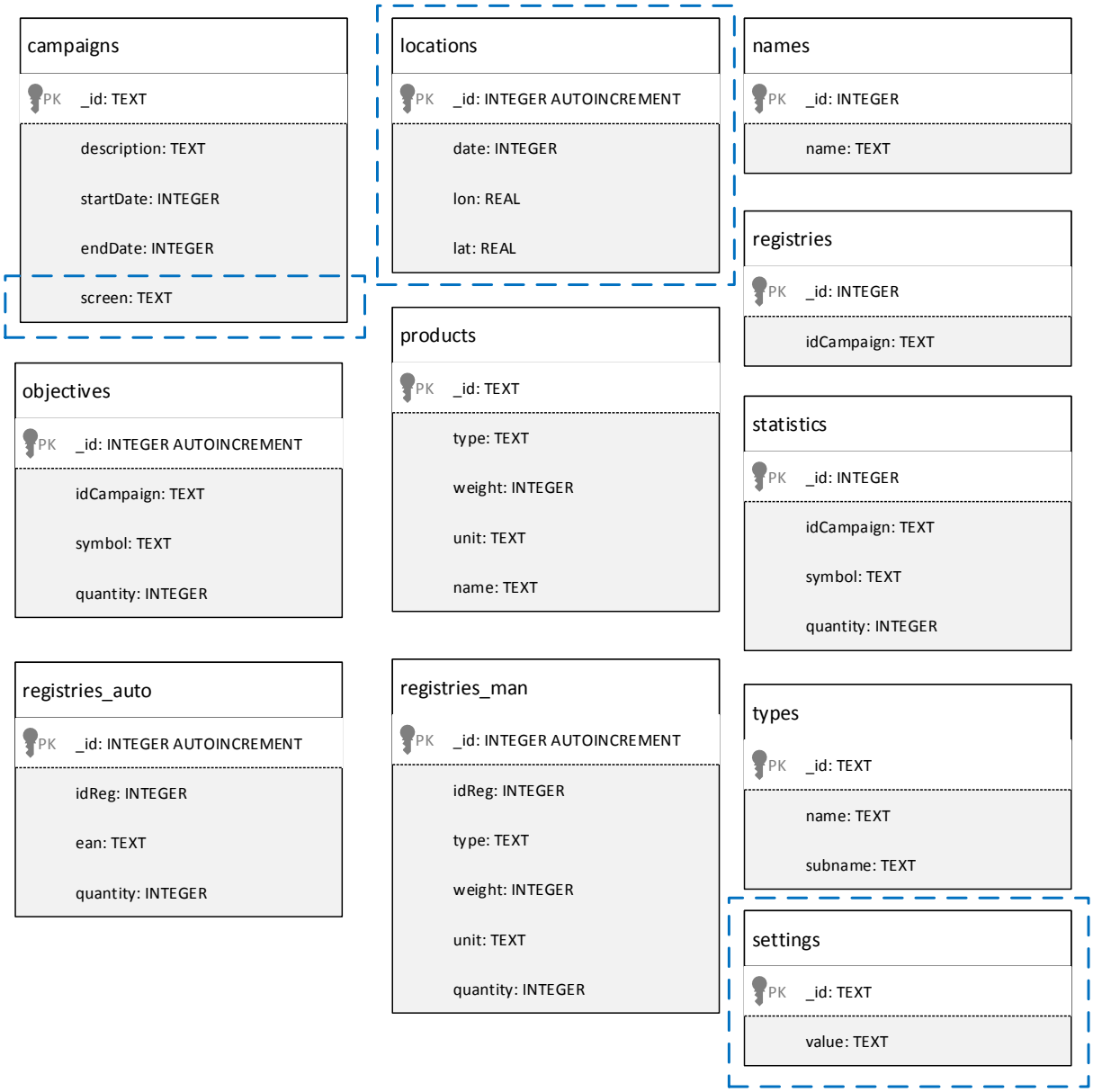



*Figura 30 - Estrutura da base de dados SQLite da release 3*

### Base de dados MongoDB (Servidor)

Com a nova capacidade da aplicação para dispositivos móveis de registar a localização em que se estavam a fazer os registos, veio a necessidade de registar essas localizações.

Juntamente com as localizações, veio também a inclusão de cálculo estatístico sobre os dados recebidos, tendo-se assim a necessidade de guardar esses valores calculados. Diretamente ligado às estatísticas, seria necessário definir os valores objetivos, neste caso o valor que se pretende atingir para cada categoria de bens.

Para possibilitar isso, criaram-se 3 novas collections na base de dados, locations, objectives e statistics, para guardarem as localizações recebidas, os valores que se pretende atingir e as estatísticas já calculadas até ao momento, respetivamente. Estas alterações podem ser vistas na Tabela 26 e em mais detalhe nas Tabela 27-Tabela 32.

| Coleção | Função |
|---|---|
| campaigns | Campanhas introduzidas no sistema |
| devices | Dispositivos registados no sistema |
| locations | Registo de localizações enviadas periodicamente por cada dispositivo |
| products | EAN, tipo e peso dos produtos reconhecidos |
| registries | Registos de doações |
| types | Tipos de produtos e a sua designação |
| deviceCampaigns | Associação entre dispositivos e campanhas |
| objectives | Lista de valores objetivos de cada bem, associado a cada campanha |
| statistics | Lista de valores atingidos de cada bem, associado a cada campanha |

*Tabela 26 - Collections da base de dados MongoDB na Release 3*

| locations | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idDevice | String | Identificador único de dispositivo |
| locations | Array | Array de dados do tipo definido na Tabela 27 |

*Tabela 27 - Modelo dos campos presentes na collection locations*

| locations->locations | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| date | Int64 | Timestamp da receção da captação da localização por parte do dispositivo movel |
| lon | Double | Longitude reportada pelo GPS |
| lat | Double | Latitude reportada pelo GPS |

*Tabela 28 - Modelo dos campos presentes em cada estrutura do array locations*

| objectives | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idCampaign | String | Identificador único da campanha |
| values | Array | Array de dados do tipo definido na Tabela 28 |

*Tabela 29 - Modelo dos campos presentes na collection objetives*

| objetives->values | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| id | String | Identificador de categoria |
| value | Int64 | Valor que se pretende alcançar da categoria em questão |

*Tabela 30 - Modelo dos campos presentes em cada estrutura do array values*

| statistics | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idCampaign | String | Identificador único da campanha |
| values | Array | Array de dados do tipo definido na Tabela 31 |

*Tabela 31 - Modelo dos campos presentes na collection statistics*

| statistics ->values | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| id | String | Identificador de categoria |
| value | Int64 | Valor alcançado até ao momento da categoria em questão |

*Tabela 32 - Modelo dos campos presentes em cada estrutura do array values*

### Interface REST

A adição de registo de localizações e o cálculo estatístico, levou à adição de 3 novos recursos REST, para envio de localizações e receção de estatísticas e objetivos de campanha. Ficando assim disponíveis os métodos presentes na Tabela 33.

| URL | Método | Descrição |
|---|---|---|
| http://server/folder/device | GET | Obtém o estado do dispositivo (p. ex. Registado, bloqueado) |
| http://server/folder/device | POST | Regista dispositivo no sistema |
| http://server/folder/location | POST | Adiciona um conjunto de localizações onde o dispositivo esteve a recolher |
| http://server/folder/campaign | GET | Obtém a lista de campanhas visíveis |
| http://server/folder/campaign | POST | Regista o dispositivo a uma campanha |
| http://server/folder/product | GET | Obtém um conjunto de produtos |
| http://server/folder/registry | POST | Adiciona um conjunto de registos para processamento |
| http://server/folder/stats | GET | Obtém estatísticas atuais da campanha |
| http://server/folder/objectives | GET | Obtém valores objetivo da campanha |
| http://server/folder/type | GET | Obtém a lista de tipos de bens |

*Tabela 33 - Lista de resources disponíveis na interface REST na Release 3*

### Armazenamento de registos

Com a introdução da inserção agrupada e a adição de cálculo estatístico, o Worker que processa os registos foi o Verticle que sofreu mais alterações em todo o servidor.

A distinção entre inserção agrupada ou normal é feita logo no início do processamento, os produtos que necessitarem serão resolvidos com base no EAN, os registos adicionados à collection registries e os valores estatísticos recalculados e incrementados, tudo da forma mais paralela possível, por forma a tirar partido das capacidades do MongoDB de acesso distribuído.

O fluxograma deste processo é visível na Figura 31.

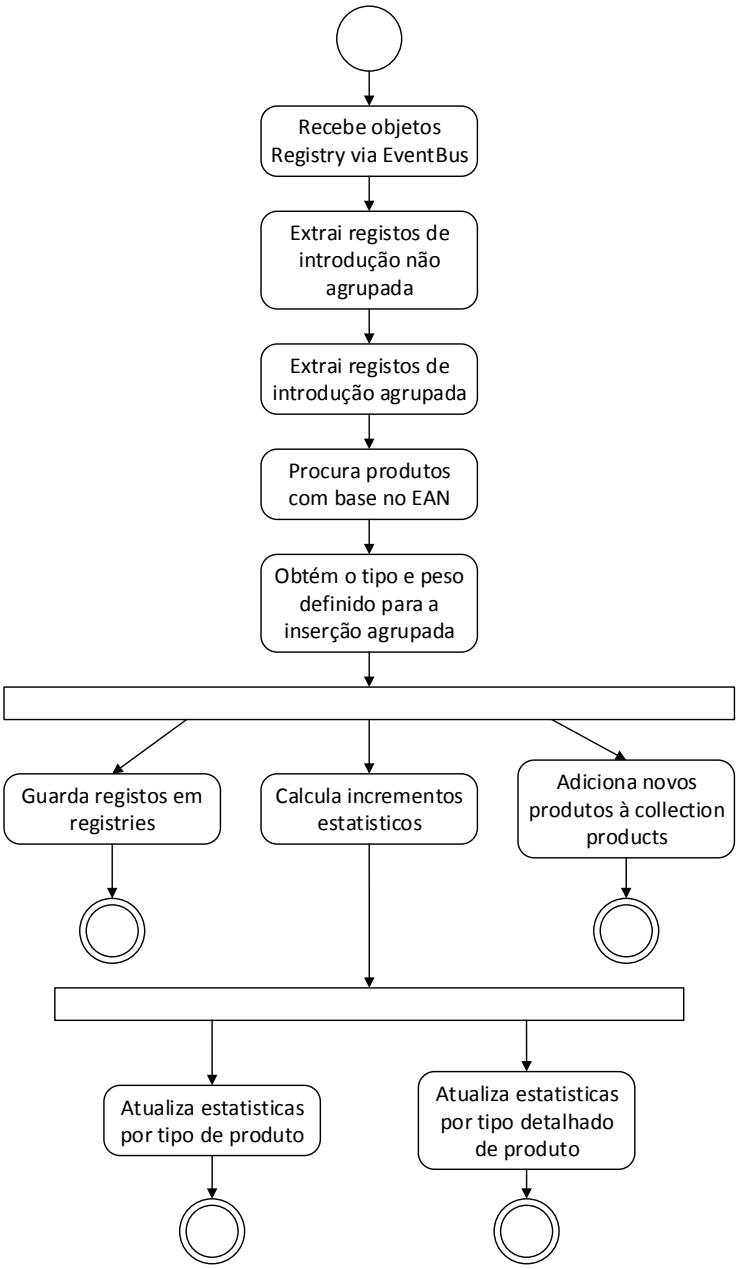



*Figura 31 - Diagrama de armazenamento de registos na release 3*

### Estrutura do servidor

Com as alterações já mencionadas, foram também criados novos Verticles para tratar os pedidos referentes a guardar localizações e obter estatísticas e objetivos. Ficando assim a estrutura do servidor como se pode ver na Figura 32.

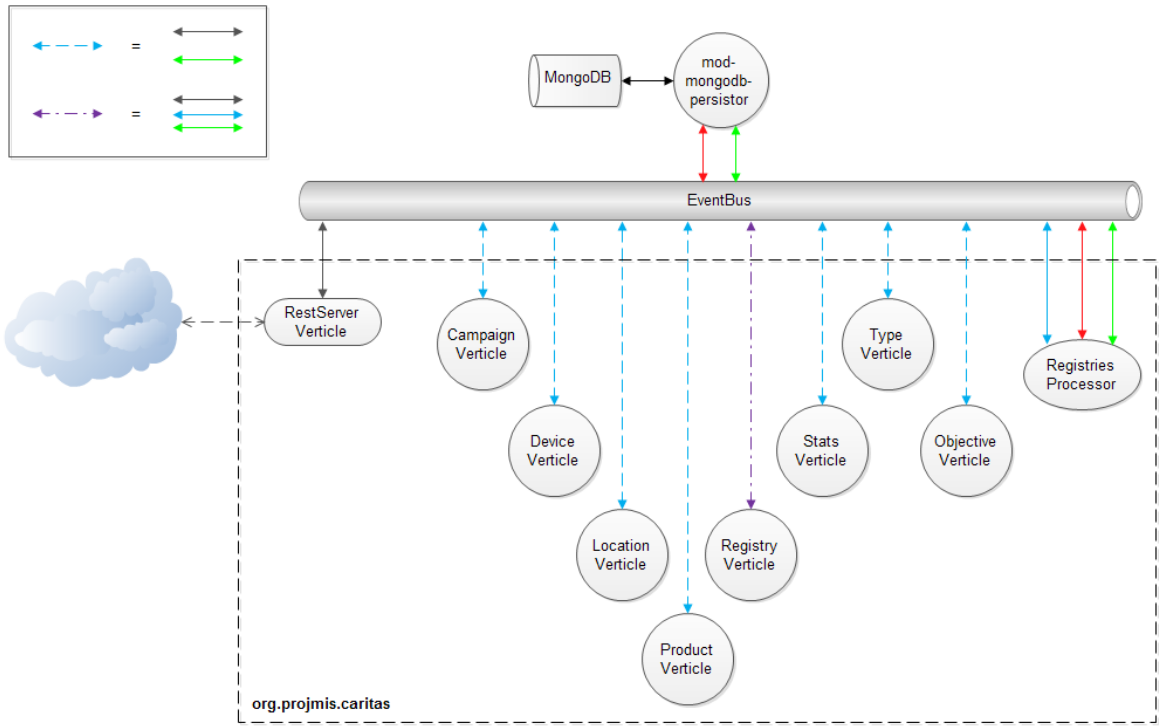



*Figura 32 - Estrutura do servidor na release 3*

## 5.6.4. Release 4

Já com as funcionalidades da release 3 implementadas, foram feitas alterações de forma a melhorar a estrutura de toda a aplicação, dando maior fiabilidade e desempenho.

Criaram-se as estatísticas pessoais como forma de incentivo aos participantes nas campanhas e adicionou-se uma interface de administração Web, a trabalhar em paralelo com o serviço REST.

Outra melhoria importante e que vai ao encontro de possibilidades da framework Vert.x, até aqui não exploradas, foi a criação de um módulo que partindo da identificação da célula de rede móvel, obtenha as coordenadas geográficas, recorrendo a tabelas de *lookup* guardadas na base de dados ou através de serviços externos. Pare efeitos de referência, neste relatório este módulo receberá a designação “CellTowerID Resolver”.

### Diagrama de atividades

Com a adição das estatísticas pessoais, foi criado um novo fragmento, com a respetiva entrada no menu lateral. Sem mais alterações, o diagrama de atividades ficou com o aspeto da Figura 33.

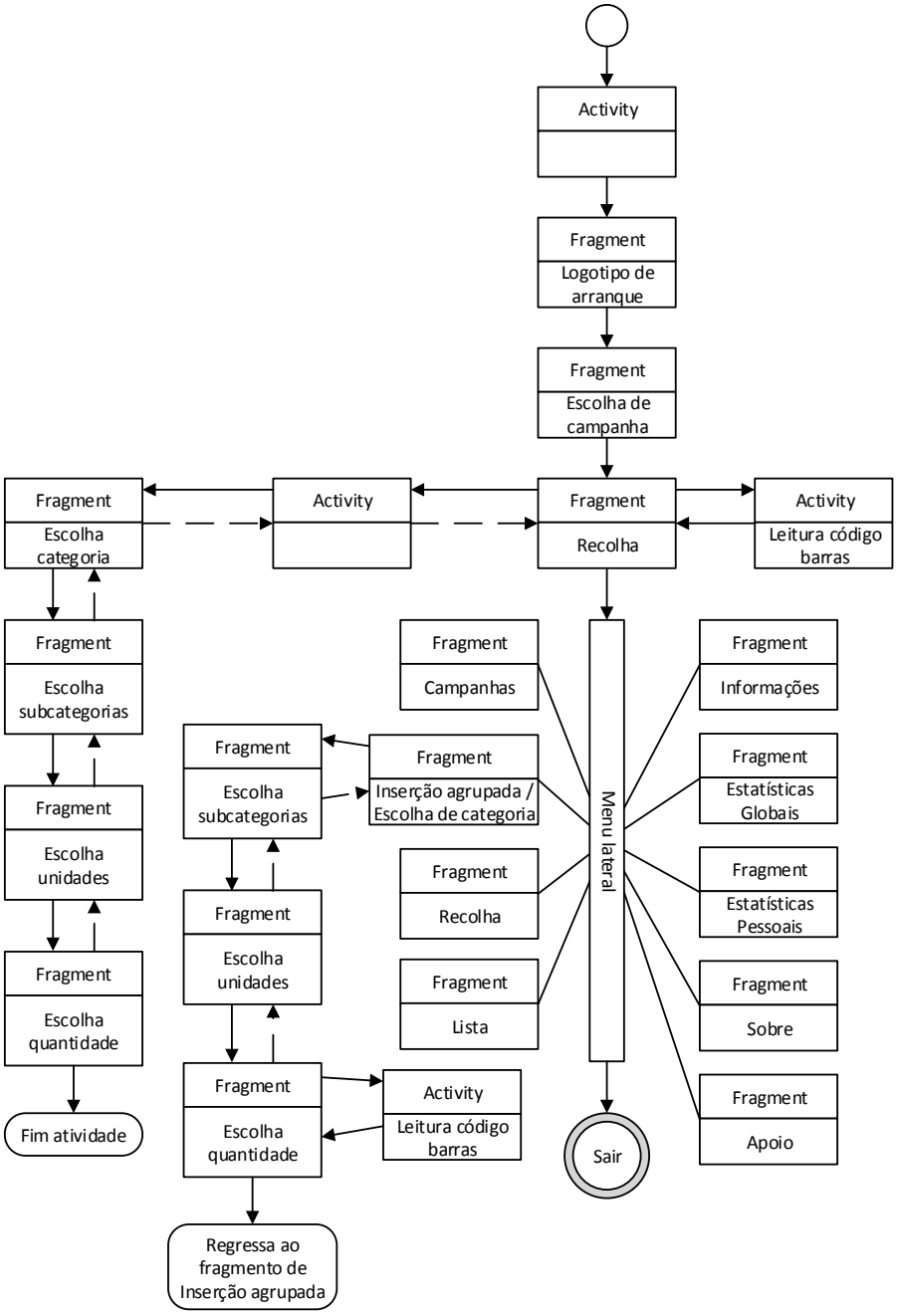



*Figura 33 - Diagrama de atividades da release 4*

**Base de dados SQLite (Dispositivos móveis)**

As releases anteriores tinham alguns problemas que a introdução agrupada veio evidenciar. Como forma de os combater, optou-se pelas seguintes soluções para combater cada problema:

- O alinhamento de localizações e registos enviados assincronamente requer bastante processamento – Todos as localizações serão gravadas no registo a que competem, usando localizações em cache e identificador de rede móvel para evitar a necessidade de ativar o GPS
- Adição de estatísticas pessoais – Adicionou-se uma nova tabela para registar os valores das estatísticas pessoais
- O método para distinguir inserções em massa de inserções normais era confuso e propicio a falhas – Adição de campos a indicar se se trata de uma introdução em massa, categoria e peso na tabela de registos
- Registos não finalizados devem ser persistidos caso a aplicação seja fechada – Adição de duas novas tabelas para conter os registos normais e registos agrupados, só sendo estes transferidos para a tabela de registos após finalização dos mesmos

Com estas alterações, foram adicionadas as seguintes tabelas na base de dados:

- *Onmass_registries* – Registos agrupados feitos, mas ainda não finalizados
- *Onscreen_registries* – Registos normais (não agrupados) feitos, mas ainda não finalizados
- *Personal_stats* – Estatísticas pessoais, tal como fornecidas pelo servidor

Além disto, a tabela *Locations* fundiu-se com a tabela *Registries*, cabendo unicamente ao dispositivo móvel informar da sua localização, preferencialmente por coordenadas GPS, ou por identificador de rede GSM caso as coordenadas GPS não estejam disponíveis.

Assim sendo, a estrutura ficou tal como se pode ver na Figura 34.

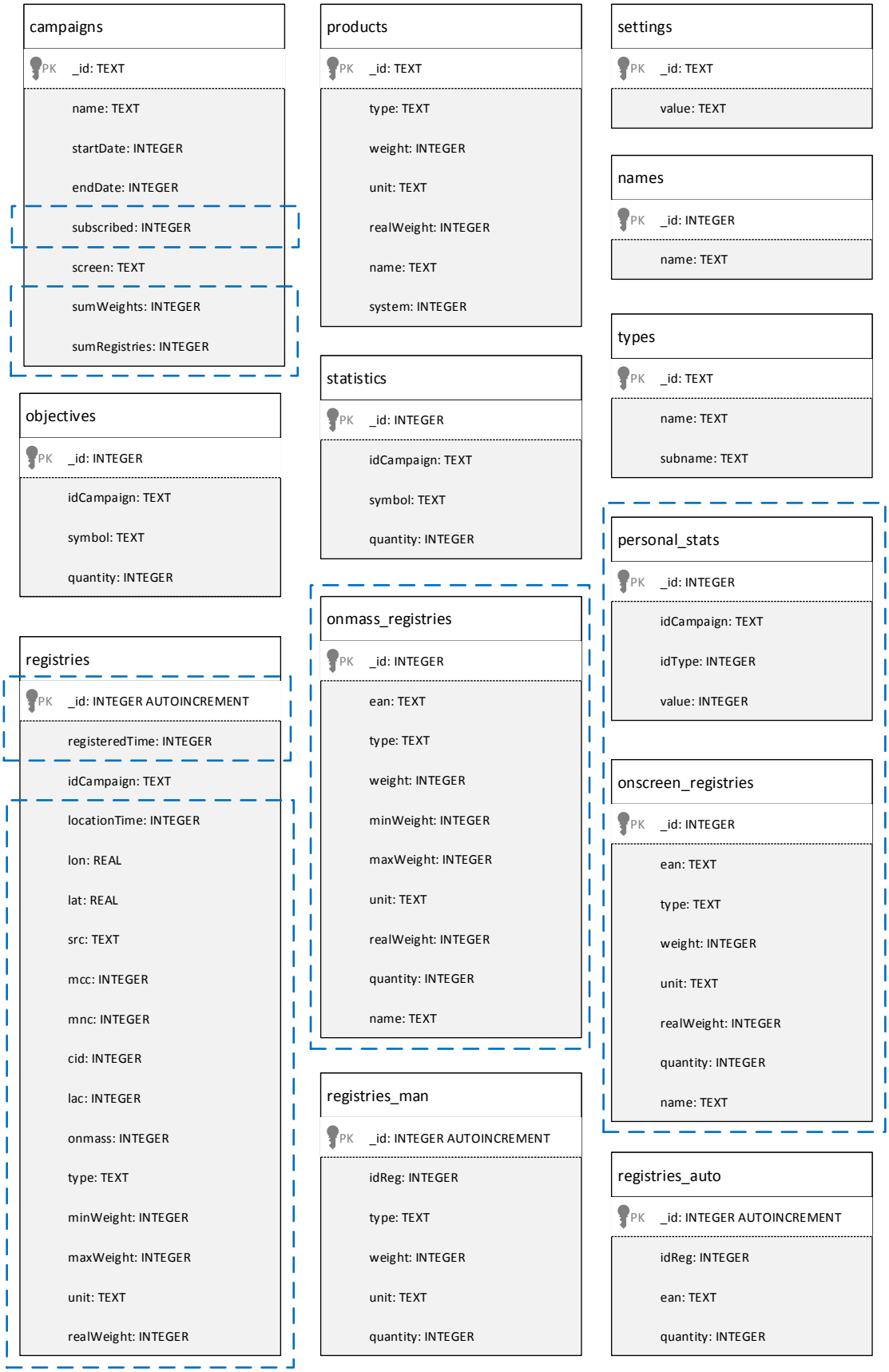



*Figura 34 - Estrutura da base de dados SQLite da release 4*

### Base de dados MongoDB (Servidor)

Após uma atenta leitura sobre design e otimização de bases de dados não relacionais [34] [57] [58], consulta de outros projetos e reflexão sobre os dados requeridos em cada consulta, chegou-se à conclusão que o modelo de base de dados do servidor não se encontrava devidamente otimizado, o que levou à consequente alteração do mesmo.

As estatísticas e objetivos passaram a fazer parte da própria campanha, minorando bastante o número de acessos à base de dados. No entanto as estatísticas pessoais ficaram numa collection separada, porque além de dependerem do dispositivo e campanha em simultâneo, a sua informação é necessária com menor frequência.

A inserção agrupada veio trazer a hipótese do sistema ir reconhecendo novos produtos ao longo do tempo, o que apesar de ser uma excelente funcionalidade, abre portas a muitas falhas de origem humana. Existe assim a hipótese do mesmo produto ser catalogado por engano numa categoria diferente ou um código EAN ser reutilizado num novo produto. Para isto foi criado a collection productConflicts, onde serão guardados os conflitos dos novos produtos recebidos, para posterior desempate pela interface de administração.

Com estas alterações, a base de dados ficou resumida às collections descritas na Tabela 34, com os pormenores de cada descritos nas Tabela 35-Tabela 45.

| **Coleção** | **Função** |
|---|---|
| campaigns | Campanhas introduzidas no sistema |
| devices | Dispositivos registados no sistema |
| personalStats | Estatísticas referentes a cada utilizador por campanha |
| productConflicts | Lista de códigos EAN em conflito |
| products | EAN, tipo e peso dos produtos reconhecidos |
| registries | Registos de doações |
| registriesSingle | Registos de doações não agrupados, para fácil exportação |
| types | Tipos de produtos e a sua designação |
| unknownProducts | Lista de EAN’s não reconhecidos |

*Tabela 34 - Collections da base de dados MongoDB na Release 4*

| **campaigns** | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idCampaign | String | Identificador único da campanha |
| name | String | Nome da campanha a apresentar |
| password | String | Password de inscrição |
| warehouse | String | Armazém de destino dos produtos |
| dateStart | Int64 | Data de início da campanha em UNIX Timestamp (milissegundos) |
| dateEnd | Int64 | Data de fim da campanha em UNIX Timestamp (milissegundos) |
| visible | Boolean | Indica se a campanha deve ser visível nos dispositivos |
| numDevices | Int32 | Número de dispositivos inscritos |
| objectives | Document | Estrutura JSON com os valores Int64 de peso pretendidos de cada categoria ou subcategoria |
| stats | Document | Estrutura JSON com os valores Int64 de peso já alcançados de cada categoria ou subcategoria |
| detailedStats | Document | Estrutura JSON com intervalos de tempo e respetivas *stats* associadas às várias categorias e subcategorias |
| sumWeights | Int64 | Somatório do peso das doações registadas (em gramas) |
| sumRegistries | Int32 | Somatório do número de registos submetidos |
| devices | Array | Lista de idDevice's dos vários dispositivos registados na campanha |

*Tabela 35 - Modelo dos campos presentes na collection campaigns*

| **devices** | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idDevice | String | Identificador único de dispositivo |
| isBlocked | Boolean | Nome da campanha a apresentar |
| ismobile | Boolean | Password de inscrição |
| lastLocation | Document | Estrutura de dados referentes à ultima localização enviada pelo dispositivo, definida na Tabela 37 |
| locations | Array | Array de dados referentes às localizações enviadas pelo dispositivo, definidas na Tabela 37 |
| campaigns | Array | Array de Strings representativas das idCampaign's das campanhas em que o dispositivo está registado |

*Tabela 36 - Modelo dos campos presentes na collection devices*

| lastLocation e locations->locations | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| date | Int64 | Timestamp da receção da captação da localização por parte do dispositivo móvel |
| lon* | Double | Longitude reportada pelo GPS |
| lat* | Double | Latitude reportada pelo GPS |
| src | String | Origem das coordenadas geográficas (p. ex. gps, celltowerid, null) |
| mcc | Int32 | MCC da rede a que o dispositivo se encontrava ligado |
| mnc | Int32 | MNC da rede a que o dispositivo se encontrava ligado |
| lac | Int32 | LAC da rede a que o dispositivo se encontrava ligado |
| cid | Int32 | CID da rede a que o dispositivo se encontrava ligado |

*Tabela 37 - Modelo dos campos presentes no campo lastLocation e em cada estrutura do array locations*

| personalStats | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idCampaign | String | Identificador único da campanha |
| idDevice | String | Identificador único de dispositivo |
| sumWeights | Int64 | Somatório do peso das doações registadas (em gramas) pelo dispositivo |
| sumRegistries | Int32 | Somatório do número de registos submetidos pelo dispositivo |

*Tabela 38 - Modelo dos campos presentes na collection personalStats*

| productConflicts | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| ean | String | Identificador do código ean em conflito |
| resolved | Boolean | Define se já foi decidido |
| entries | Array | Array de dados do tipo definido na Tabela 40 |
| choosed | Int32 | Indicador do elemento de entries que foi eleito como correto |

*Tabela 39 - Modelo dos campos presentes na collection productConflicts*

| productConflicts->entries | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| type | String | Indicador do tipo de produto |
| weight | Int32 | Peso ou volume médio do produto, de acordo com as unidades presentes em unit |
| unit | String | Unidade em que é medido o produto |
| realWeight | Int32 | Peso médio do produto em gramas |
| modified | Int64 | Data em UNIX Timestamp (milissegundos) em que o registo foi modificado ou adicionado |
| name | String | Nome dado ao produto, a aparecer nos dispositivos |
| owner* | String | idDevice do dispositivo de quem adicionou o novo produto ao sistema |

*Tabela 40 - Modelo dos campos presentes em cada estrutura do array entries*

| **products** | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| ean | String | Identificador do código ean do produto |
| type | String | Indicador to tipo de produto |
| weight | Int32 | Peso ou volume médio do produto, de acordo com as unidades presentes em unit |
| unit | String | Unidade em que é medido o produto |
| minWeight* | Int32 | Valor mínimo de weight em que o produto se pode enquadrar |
| maxWeight* | Int32 | Valor máximo de weight em que o produto se pode enquadrar |
| realWeight | Int32 | Peso médio do produto em gramas |
| modified | Int64 | Data em UNIX Timestamp (milissegundos) em que o registo foi modificado ou adicionado |
| name | String | Nome dado ao produto, a aparecer nos dispositivos |
| owner* | String | idDevice do dispositivo de quem adicionou o novo produto ao sistema |

*Tabela 41 - Modelo dos campos presentes na collection products*

| **registries** | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| idCampaign | String | Identificador único da campanha |
| idDevice | String | Identificador único de dispositivo |
| date | Int64 | Data em UNIX Timestamp (milissegundos) do momento em que o registo foi finalizado no dispositivo |
| dateReceived | Int64 | Data em UNIX Timestamp (milissegundos) do momento em que o registo foi recebido no servidor |
| location | Document | Estrutura de dados referentes à localização em que o registo foi criado, definida na Tabela 37 |
| massInput | Boolean | Indica se o registo trata-se de uma inserção agrupada |
| type* | String | Indicador to tipo de produto |
| weight* | Int32 | Peso ou volume médio do produto, de acordo com as unidades presentes em unit |
| unit* | String | Unidade em que é medido o produto |
| minWeight* | Int32 | Valor mínimo de weight em que o produto se pode enquadrar |
| maxWeight* | Int32 | Valor máximo de weight em que o produto se pode enquadrar |
| realWeight* | Int32 | Peso médio do produto em gramas |
| products | Array | Array de produtos contidos no registo, com os dados definidos na Tabela 43 |

*Tabela 42 - Modelo dos campos presentes na collection registries*

| registries ->products | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| ean* | String | Identificador do código EAN do produto |
| type | String | Indicador do tipo de produto |
| weight | Int32 | Peso ou volume médio do produto, de acordo com as unidades presentes em unit |
| unit | String | Unidade em que é medido o produto |
| realWeight | Int32 | Peso médio do produto em gramas |

*Tabela 43 - Modelo dos campos presentes em cada estrutura do array products*

| registriesSingle | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| ean* | String | Identificador do código ean do produto |
| quantity | Int32 | Quantidade de produtos iguais doados |
| type | String | Indicador to tipo de produto |
| weight | Int32 | Peso ou volume médio do produto, de acordo com as unidades presentes em unit |
| unit | String | Unidade em que é medido o produto |
| realWeight | Int32 | Peso médio do produto em gramas |
| date | Int64 | Data em UNIX Timestamp (milissegundos) do momento em que o registo foi finalizado no dispositivo |
| idDevice | String | Identificador único de dispositivo |
| idCampaign | String | Identificador único da campanha |
| location | Document | Estrutura de dados referentes à localização em que o registo foi criado, definida na Tabela 37 |

*Tabela 44 - Modelo dos campos presentes na collection registriesSingle*

| unknownProducts | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| ean | String | Identificador do código EAN do produto |
| quantity | Int32 | Número de produtos com o mesmo EAN no registo |
| idDevice | String | Identificador único de dispositivo |
| idCampaign | String | Identificador único da campanha |
| location | Document | Estrutura de dados referentes à localização em que o registo foi criado, definida na Tabela 37 |

*Tabela 45 - Modelo dos campos presentes na collection unknownProducts*

*Campo não obrigatório

### Envio de dados

Foram feitas melhorias no processo de envio de dados para o servidor, tentando atingir um maior equilíbrio entre atualização de dados e envio de registos.

As principais medidas foram:

- Pausa entre acessos dividido em duas categorias de tempo variável, 10-13 minutos de espera caso a última atualização tenha ocorrido com sucesso ou 5-7 minutos de espera caso contrário
- Novos produtos registados pela introdução agrupada serão imediatamente submetidos para posteriormente facilitar e agilizar a resolução de produtos no servidor
- A leitura de estatísticas globais e pessoais são feitas antes e depois do envio de registos, assim em caso de falha de conetividade durante envios longos, pelo menos sempre será possível obter as estatísticas o mais atualizadas possível

Tendo isto em conta, os eventos para submissão de registos para o servidor podem resumir-se ao fluxograma da Figura 35.

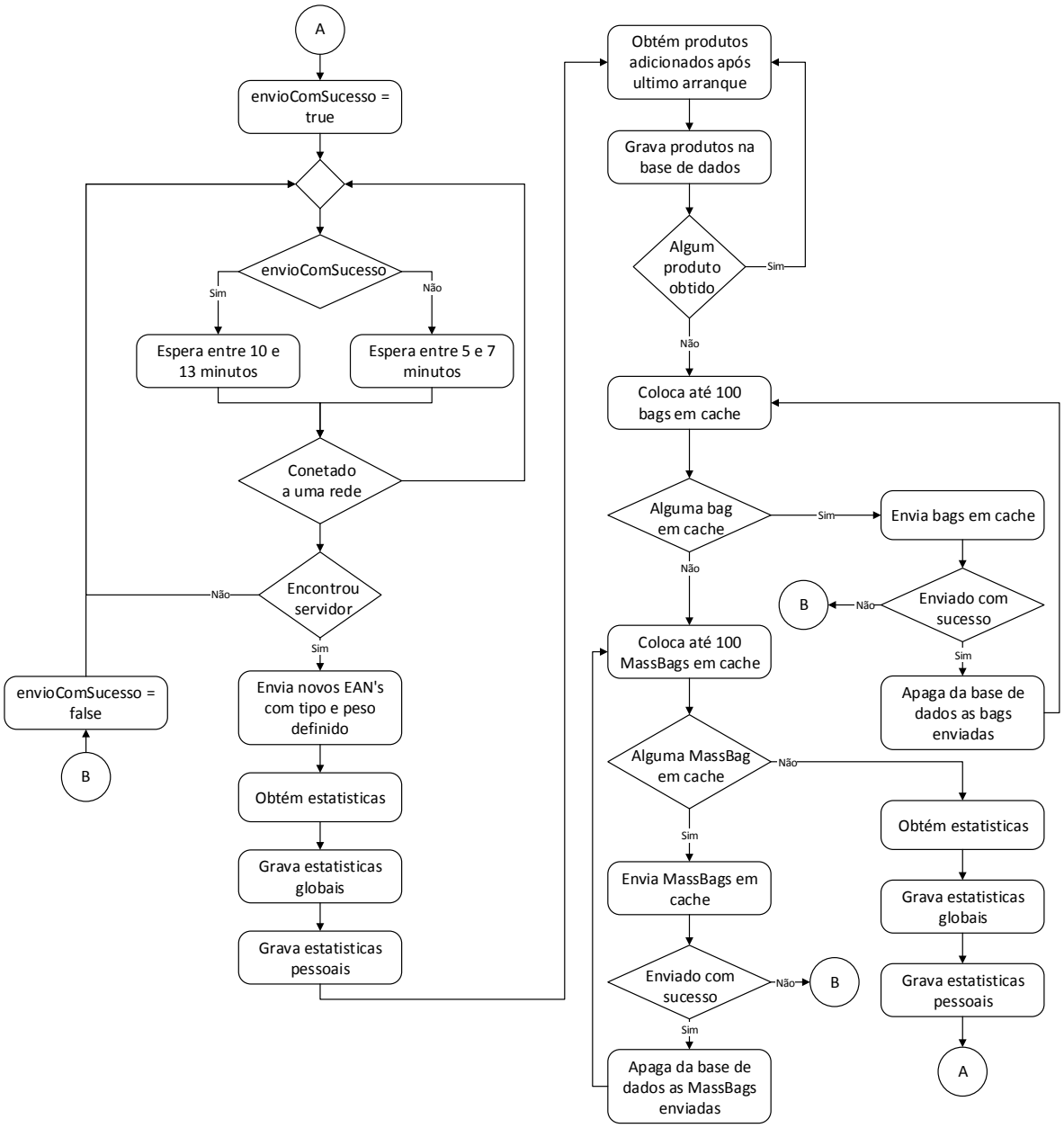



*Figura 35 - Sequência de passos para envio de dados na release 4*

**Armazenamento de registo**

Com as alterações introduzidas na estrutura da base de dados e as alteração ao Worker associado, foi possível obter um maior paralelismo de ações, o que se reflete imediatamente num aumento de performance de todo o sistema. Aumento este que poderia ser ampliado se fossem usados vários servidores de base de dados.

A resolução de localizações através do módulo criado é também logo uma das primeiras ações aquando da receção de registos, sendo esta feita de forma a não reduzir a velocidade do ciclo de eventos.

Com estes pequenos passos, a performance do servidor foi grandemente aumentada, ficando o novo fluxograma de armazenamento de registos como se pode ver na Figura 36.

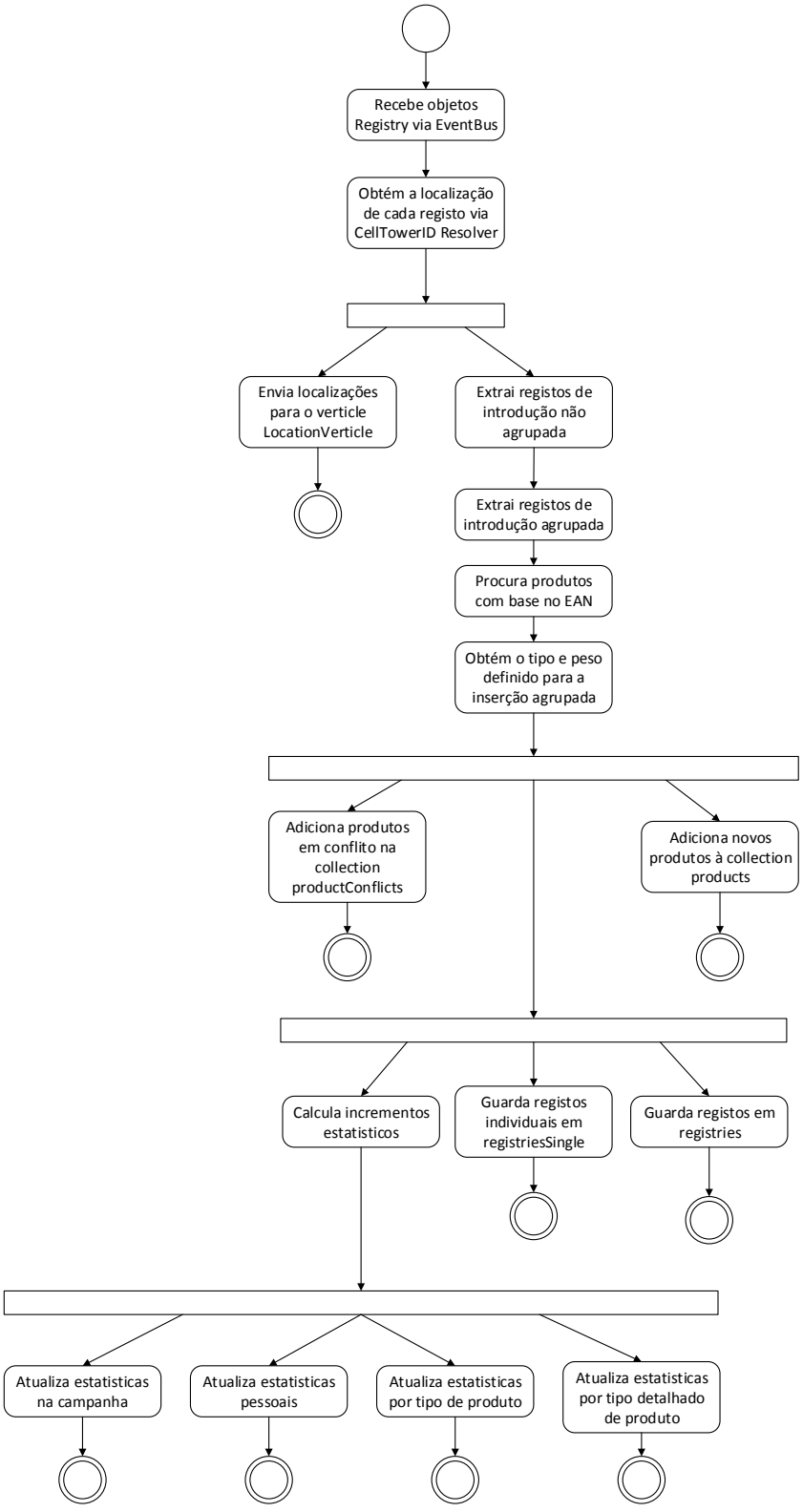



*Figura 36 - Diagrama de armazenamento de registos na release 4*

### Resolução de posições

O módulo de resolução de posições foi um dos aspetos de realce desta release, permitindo obter as coordenadas geográficas de um registo unicamente através do identificador da célula de rede móvel.

As suas fontes de dados são uma tabela preenchida com os dados do OpenCellID [59], dados recolhidos num trabalho da cadeira de Computação Ubíqua do Mestrado em Informática e Sistemas e uma cache de localizações obtidas com base no servidor Mmap da Google [60], o mesmo usado pelo próprio sistema Android para o mesmo efeito, pela ordem apresentados.

O modelo de funcionamento está apresentado no fluxograma da Figura 37.

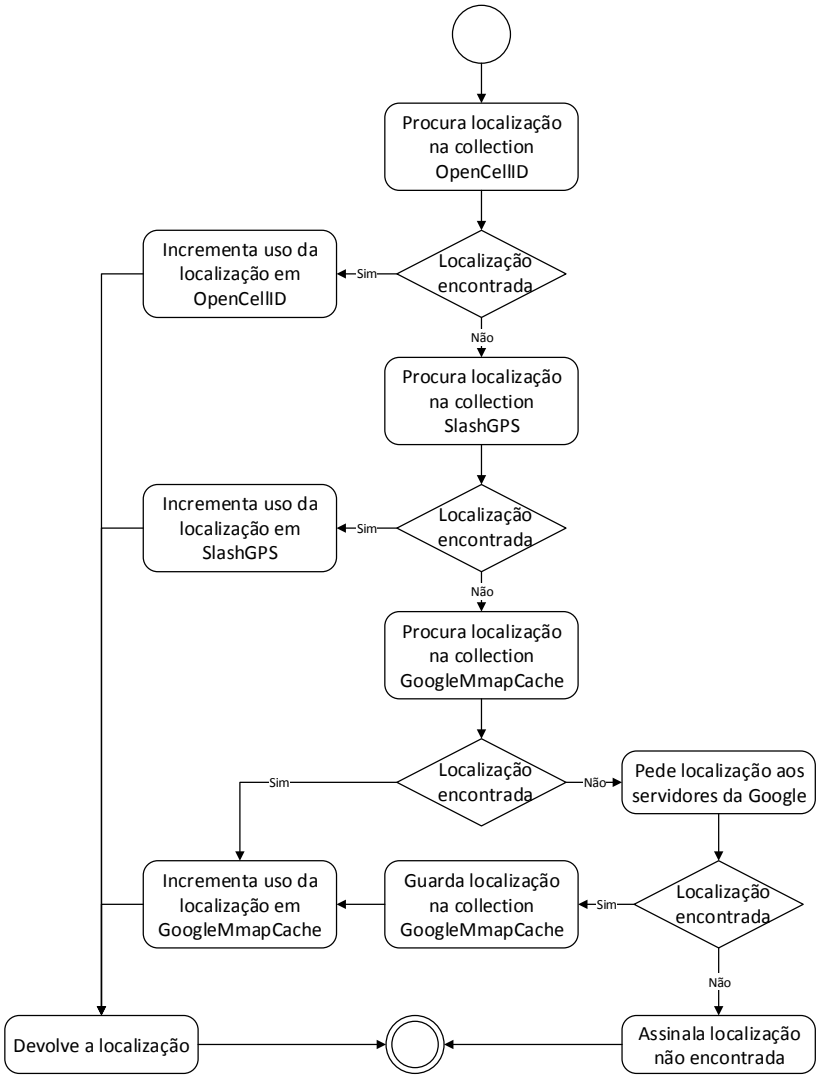



*Figura 37 - Diagrama de procedimentos para resolução de localizações*

### Esquema da base de dados MongoDB (CellTowerID Resolver)

O módulo criado para resolução de localizações está igualmente dependente dos dados presentes numa base de dados. Embora fosse possível estarem todos os dados alojado na mesma base de dados, optou-se por isolar as tabelas referentes a este módulo numa base de dados distinta. Isto abrirá portas à possibilidade de reaproveitar este módulo noutros projetos, expandir a sua utilização por partes de *software* distintas que venham a ser implementadas e facilitar o teste do módulo como unidade independente.

Com isto a base de dados do módulo CellTowerID Resolver pode resumir-se às collections descritas na Tabela 46, e em mais pormenor na Tabela 47, que é válida para as diversas collections.

| **Coleção** | **Função** |
|---|---|
| slashgps | Dados de localizações do projeto SlashGPS |
| opencellid | Dados de localizações do projeto OpenCellID |
| mmapcache | Dados de localizações obtidas dos servidores do Google |

*Tabela 46 - Collections da base de dados MongoDB na Release 4 associado ao módulo de resolução de localizações*

| **slashgps, opencellid e mmapcache** | | |
|---|---|---|
| **Nome** | **Tipo** | **Função** |
| _id | ObjectId | Identificador único gerado pela base de dados |
| mcc | Int32 | Identificador de código de pais da comunicação móvel |
| mnc | Int32 | Identificador de código de rede da comunicação móvel |
| lac | Int32 | Identificador da “área de localização” do controlador da base da estação da rede de comunicação móvel |
| cid | Int32 | Identificador da estação do transceiver ou setor de transceiver em caso de antenas direcionáveis |
| lon | Double | Longitude estimada da antena de rede móvel |
| lat | Double | Latitude estimada da antena de rede móvel |
| nUsed | Int32 | Número de vezes que o registo permitiu a resolução de uma localização |

*Tabela 47 - Modelo dos campos presentes nas collections slashgps, opencellid e mmapcache*

### Interface de administração

A interface de administração é acessível pelo mesmo endereço em que o servidor REST se encontra à escuta. Dado que o servidor REST é uma reimplementação de um Web Server, muitas das funcionalidades normalmente presentes noutras frameworks de Web Servers, também se encontram aqui.

Definiu-se assim as URL's presentes na Tabela 48 e Tabela 49.

| **Método GET** | |
|---|---|
| **URL** | **Descrição** |
| /admin | Redireciona para /admin/index.html |
| /admin/ | Redireciona para /admin/index.html |
| /admin/index.html | Página inicial de administração |
| */favicon.ico | Devolve o conteúdo do ficheiro images/favicon.ico |
| /admin/login.html | Página de login |
| /admin/logout.html | Invalida o cookie de sessão e redireciona para /admin/login.html |
| /admin/images/(image) | Devolve o conteúdo da imagem images/(image) |
| /admin/campaigns.html | Página de gestão de campanhas |
| /admin/devices.html | Página de gestão de dispositivos |
| /admin/conflicts.html | Página de gestão de conflitos |
| /admin/stats.html | Página de seleção de estatísticas a visualizar |
| /admin/objectives.html | Página de seleção de objetivos a visualizar |
| /admin/export.html | Páginas de exportação de dados |

*Tabela 48 - Lista de resources disponíveis via método GET para a interface de administração*

| **Método POST** | |
|---|---|
| **URL** | **Descrição** |
| /admin/login.html | Envio de dados de autenticação |
| /admin/campaigns.html | Alterar visibilidade de campanhas |
| /admin/newcampaign.html | Criação de campanha |
| /admin/delcampaign.html | Eliminação de campanha |
| /admin/devices.html | Bloqueio de dispositivos |
| /admin/conflicts.html | Decisão para resolução de conflitos |
| /admin/objectives.html | Definição de valor objetivo |

*Tabela 49 - Lista de resources disponíveis via método POST para a interface de administração*

### Estrutura do servidor

Em comparação com a release anterior, esta tem uma estrutura em tudo semelhante, sendo apenas de salientar a adição do Verticle que trata das tarefas administrativas e da criação do código das páginas para o mesmo efeito e também a incorporação do módulo de resolução de localizações.

O diagrama da estrutura do servidor para esta release pode ser visto na Figura 38.

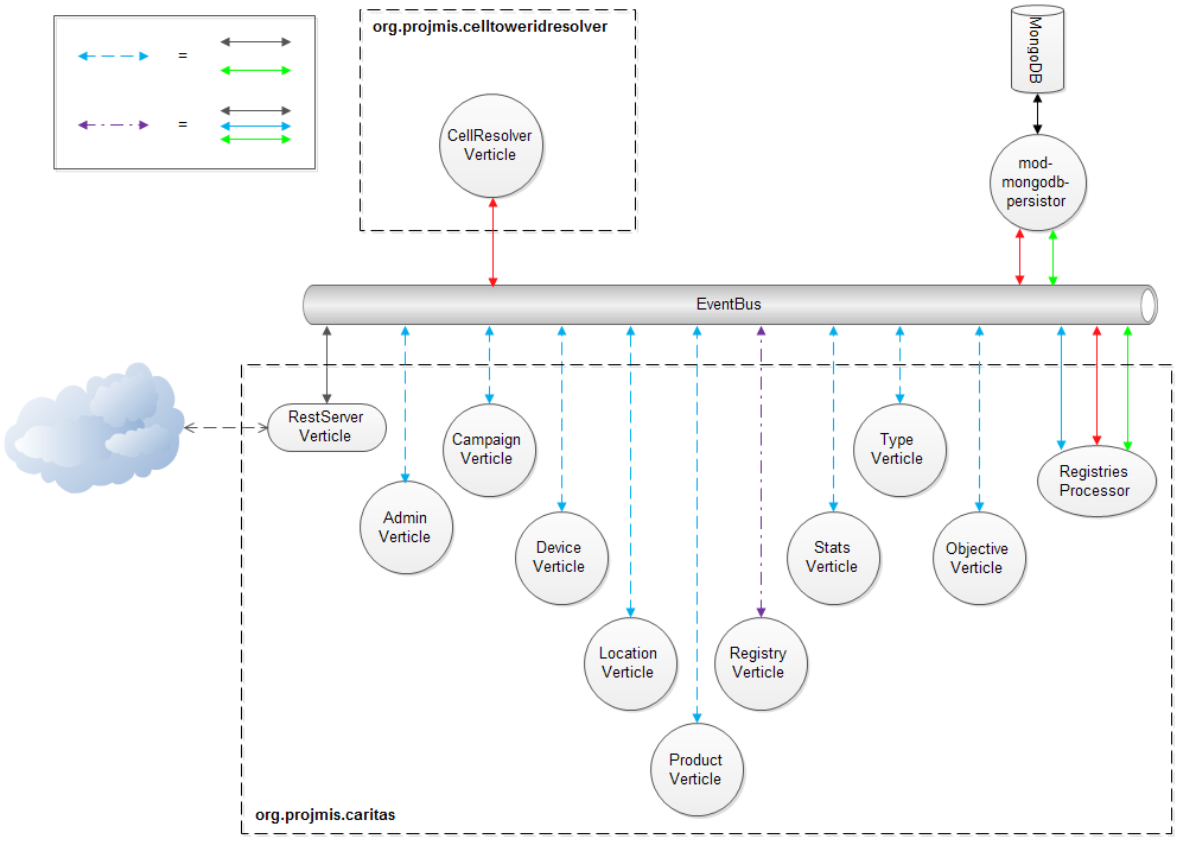



*Figura 38 - Estrutura do servidor na release 4*

### 5.6.5. Release 5

Esta última release teve como ponto central trazer otimizações que consigam resolver problemas encontrados e a correção de bugs detetados.

A maior otimização teve a ver precisamente com a comunicação com o servidor durante o arranque da aplicação cliente, onde havia a hipótese de os dados ficarem inconsistentes no dispositivo móvel dado ser necessário várias chamadas REST para obter todos os dados necessários, caso a ligação de dados não fosse estável. A solução para isto passou por reestruturar a interface REST.

#### Interface REST

Pelos motivos descritos, as *resources* da interface REST foram diminuídas e todos os dados necessários para o arranque da aplicação móvel foram centrados em "startup". Embora as normas referentes à construção de WebServices REST não recomendem a implementação de métodos GET's que possam modificar dados, neste caso em particular achou-se que tornaria mais fácil compreensão e diminuiria pelo menos mais um acesso ao servidor.

A compactação de vários *resources* num único destinado ao arranque da aplicação conseguiu eliminar todas as falhas de acesso aos mesmos, não tendo sido encontrada mais nenhuma forma que levasse a um arranque com dados inconsistentes.

A interface REST ficou por isso reduzida ao que se pode ver na Tabela 50.

| URL | Método | Descrição |
|---|---|---|
| http://server/folder/startup | GET | Obtém todos os dados referentes ao estado do dispositivo, campanhas, objetivos, estatísticas e tipos, e também regista o dispositivo no sistema se necessário |
| http://server/folder/ping | POST | Informa se o servidor está pronto a processar pedidos |
| http://server/folder/campaign | POST | Regista o dispositivo numa nova campanha |
| http://server/folder/product | GET | Obtém um conjunto de produtos |
| http://server/folder/registry | POST | Adiciona um conjunto de registos para processamento |
| http://server/folder/stats | GET | Obtém estatísticas atuais |

*Tabela 50 - Lista de resources disponíveis na interface REST na Release 5*

## 5.7. Testes aplicados

Visto que este projeto poderá vir a ser implementado fora do contexto académico, e para facilitar igualmente no processo de deteção, identificação e resolução de bugs, algumas medidas foram tomadas no contexto de testes de software.

Mesmo sendo um projeto desenvolvido por uma só pessoa, é importante seguir certas regras, de forma a prevenir futuros problemas, muitas vezes indicadores de más metodologias de teste durante o desenvolvimento. Esta necessidade revelou-se ainda mais importante com a introdução de novos requisitos nas várias releases, sem o qual todo o desenvolvimento teria demorado certamente mais tempo.

A aplicação servidor foi a que se revelou mais fácil de aplicar os vários testes, quer pela simplicidade de acesso e boa descrição de serviços REST, quer pelos próprios mecanismos de teste do Vert.x, que automaticamente executam testes no momento de compilação. No entanto a validação destes testes nem sempre foi simples, pois apesar de alguns métodos mais simples apenas retornarem dados que podem ser confrontados com uma tabela de lookup, outros métodos levam a alterações dos dados na base de dados, nem sempre de forma consistente no intervalo de tempo desejado, o que levou a que alguns testes tivessem que ser validados de forma manual.

A aplicação cliente foi bem mais difícil de testar, muitas vezes recorrendo apenas a inspeções de código e analise de logs, especialmente para as camadas funcionais. A camada gráfica da aplicação cliente teve que ser testada de forma puramente manual, fazendo passagem entre os ecrãs, tentando alterar a ordem dos fluxos de dados e simular eventos que se crê que pudessem levar à ocorrência de falhas.

Foram também feitos testes funcionais e de usabilidade por responsáveis da Caritas ao longo de todo o desenvolvimento, desta forma não houve apenas um foco no código, mas na aplicação e a sua interação como um todo.

No final, foi possível detetar e eliminar todos bugs encontrados, o que não garantindo a sua inexistência, já dá garantias de uma aplicação refinada e pronta a ser utilizada em campo.

## 5.8. Resultado final

Este subcapítulo pretende dar a conhecer o aspeto gráfico da aplicação final, o qual foi um ponto muito importante, uma vez que esta aplicação será usada por pessoas comuns, sem grande conhecimento tecnológico. Um estudo das *guidelines* e muita prototipagem, até conseguir testar e compreender devidamente várias funcionalidades e conceitos introduzidos a partir do Android 3.0, foram ferramentas imprescindíveis na conceção de toda a parte visual desta aplicação.

Com isto e após vários *mockups*, obteve-se a estrutura para a interface gráfica descrita na Figura 39. Cada um dos respetivos ecrãs serão descritos com mais pormenor no seguimento deste subcapítulo.

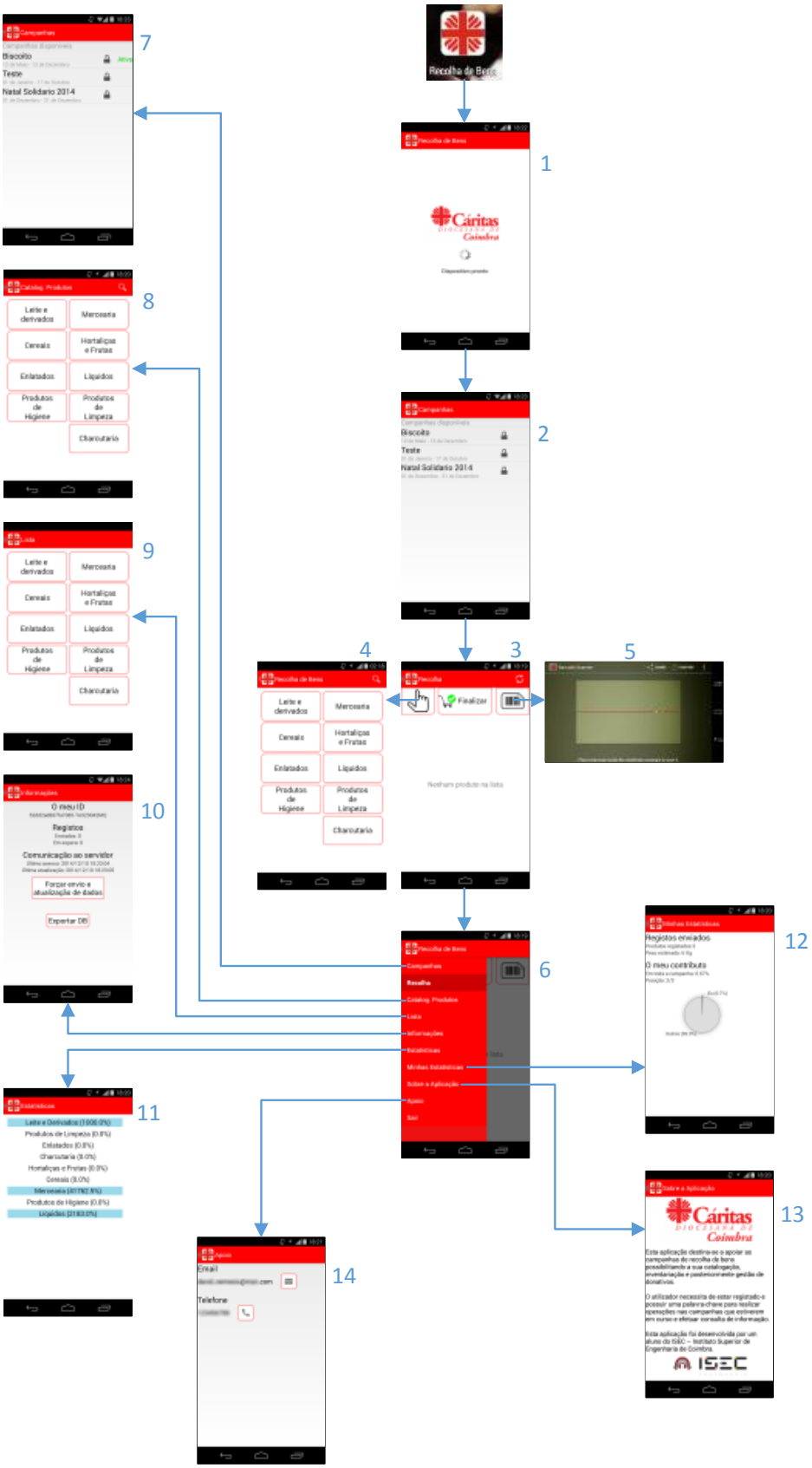



*Figura 39 - Estrutura da aplicação cliente*

### 5.8.1. Arranque da aplicação (1)

Este ecrã é visível sempre que se inicia a aplicação, partindo do princípio que esta não se encontra em background.

Contém apenas o logotipo da Cáritas Diocesana de Coimbra, um ícone animado para mostrar que a aplicação continua a responder e uma mensagem a descrever a ação atual, tal como se pode ver na Figura 40.


*Figura 40 - Ecrã de arranque da aplicação*

É nesta fase que a aplicação verifica a disponibilidade da rede, submete qualquer registo que esteja em lista de espera e atualiza todos os dados referentes a campanhas, estatísticas e produtos.

Assim que todos os dados forem atualizados ou caso ocorra *timeout* da ligação, a aplicação passa para o ecrã seguinte com um efeito de *fade-out*.

Caso a tecla *back* seja pressionada duas vezes, a aplicação encerra.

### 5.8.2. Escolha de campanha no arranque (2)

Nesta fase da aplicação o utilizador terá que escolher a campanha para a qual está a fazer a recolha. Visualmente é semelhante ao que se pode ver na Figura 41.


*Figura 41 - Ecrã de escolha de campanha no arranque da aplicação*

Para melhor proteção de acessos, estas campanhas têm uma *password* associada, sendo necessário que a mesma seja fornecida, ficando a partir daí o dispositivo inscrito. As campanhas inscritas podem ser facilmente identificadas através de um ícone de cadeado ao lado dos respetivos nomes, onde um cadeado aberto significa que a campanha está destrancada para o dispositivo em questão. Cadeados fechados significam que o dispositivo ainda não está inscrito nas campanhas e qualquer tentativa de inscrição resultará numa *dialogbox* a pedir a respetiva *password*.

Após seleção da campanha a aplicação passa para o ecrã seguinte.

Caso a tecla *back* seja pressionada duas vezes, a aplicação encerra.

### 5.8.3. Recolha ou Ecrã Principal (3)

Este é o primeiro ecrã a aparecer caso a aplicação esteja devidamente iniciada e pronta a ser utilizada. É aqui que podem ser registados produtos através de código de barras ou através de introdução manual numa lista construída especificamente para esse fim.

Pode tomar o aspeto do primeiro ecrã da Figura 42 caso não esteja nenhum produto registado, ou algo semelhante ao segundo ecrã da mesma Figura, após inserir pelo menos um registo (neste caso particular 1 embalagem de 1.5L de leite).

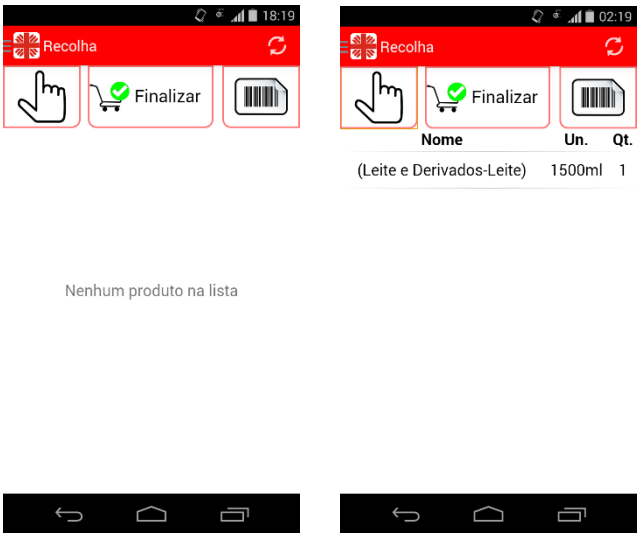



*Figura 42 - Ecrã de recolha (sem produtos e com um produto registado)*

O botão de finalizar fará os registos serem arquivados e preparados para envio assim que tal for possível.

Caso a tecla *back* seja pressionada duas vezes, a aplicação será colocada em background e todos os registos feitos até ao momento permanecerão guardados.

### 5.8.4. Introdução manual (4)

A Figura 43 mostra o ecrã disponibilizado sempre que se escolhe introdução manual no ecrã de Recolha. Este permitirá escolher uma determinada categoria de produtos a adicionar.

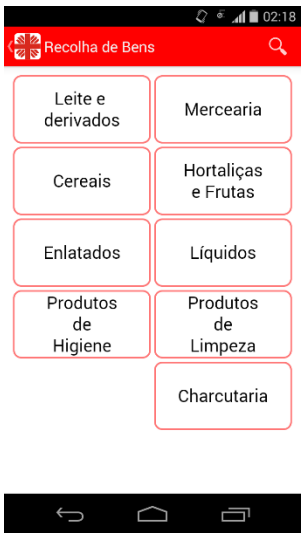



*Figura 43 - Primeiro ecrã de introdução manual de produto (escolha de categoria)*

Ao escolher uma categoria, será apresentado um ecrã para escolher subcategorias, unidades e quantidade, tal como se pode ver no primeiro, segundo e terceiro ecrã da Figura 44, respetivamente.


*Figura 44 - estantes ecrãs para introdução manual de produto (escolha de subcategoria, unidades e quantidade)*

Ao pressionar o botão Adicionar, a aplicação retorna ao ecrã de recolha com o produto adicionado.

Pressionar o botão *back* fará a mudança para os ecrãs anteriores até voltar ao ecrã de recolha.

### 5.8.5. Introdução automática por código de barras (5)

Selecionar esta opção no modo de recolha fará a ativação do modo de captura semelhante ao que se pode ver na Figura 45. O utilizador deverá colocar o código de barras em frente à câmara, a uma distância razoável, para que este esteja focado e facilmente visível.

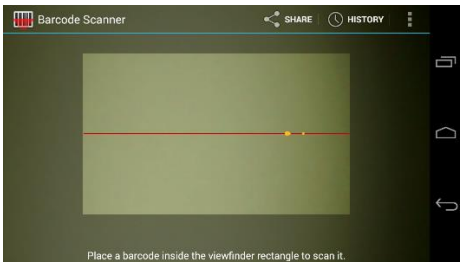



*Figura 45 - Ecrã da aplicação em modo de leitura de código de barras*

Após cada leitura será lançado novamente o leitor de código de barras, até que o utilizador pressione intencionalmente o botão *back*.

### 5.8.6. Menu de navegação (6)

Sempre que o ícone de menu lateral esteja visível no canto superior esquerdo ao lado do ícone da aplicação, é possível abrir o menu de navegação pressionando o ícone da mesma ou fazendo slide da esquerda para a direita.

O menu tomará o aspeto presente na Figura 46 e permitirá a navegação pelos diversos ecrãs da aplicação.


*Figura 46 - Menu de navegação lateral*

### 5.8.7. Escolha de campanha (7)

Tal como se pode ver na Figura 47, este ecrã é em tudo semelhante ao ecrã (2). As únicas diferenças relevantes são que o utilizador não é forçado a selecionar uma campanha para continuar mudar de ecrã e tem indicação de qual a campanha em uso.

Caso a tecla *back* seja pressionada duas vezes, a aplicação será colocada em background.


*Figura 47 - Ecrã de escolha de campanha após inicialização*

### 5.8.8. Catalogação de produtos ou Inserção agrupada (8)

Este modo de introdução de produtos é em tudo semelhante ao modo de introdução manual do ecrã de Recolha. A sua principal diferença reside no último ecrã de escolha de quantidade, onde em vez de dispor apenas de um *NumberPicker* para escolher a quantidade, tem também um botão para ativar a leitura de códigos de barras.

O utilizador com isto irá ler vários códigos de produtos da mesma categoria, subcategoria e pesos semelhantes, só sendo necessário identificar a quantidade de cada um fazendo *tap* no registo que pretende editar.

A nível visual, o ecrã assume o formato do primeiro ecrã da Figura 48 caso não contenha registos, ou do segundo ecrã após se adicionar pelo menos um registo.


*Figura 48 - Ecrã de inserção agrupada (sem produtos e com um produto registado)*

Ao pressionar Finalizar, os registos serão arquivados para envio, à semelhança do que acontece com o ecrã de Recolha.

### 5.8.9. Lista (9)

Contrariamente aos ecrãs até agora mostrados, este tem não nenhuma utilidade funcional para a aplicação. É semelhante à introdução manual ou à catalogação agrupada, só dispondo dos ecrãs de categorias e subcategorias, unicamente para fins de visualização.

### 5.8.10. Informações (10)

Este ecrã destina-se a visualizar o estado das atualizações e submissões de dados, permitindo ter garantias que o envio de dados está a ser feito com sucesso. A Figura 49 retrata um aspeto possível para este ecrã.


*Figura 49 - Ecrã de informações*

São apresentados o ID do dispositivo para gestão interna, o número de registos enviados com sucesso e os que ainda estão a aguardar o envio. Também é possível ver quando foi feito o último contacto com o servidor e a ultima altura em que foram trocados dados.

Existe também um botão para forçar imediatamente o envio e atualização de dados (em vez de esperar o tempo estipulado no algoritmo) e um botão para exportar a base de dados atual, unicamente para fins de *debug*.

### 5.8.11. Estatísticas (11)

Tratando-se de um dos elementos mais importantes para este projeto, é com este ecrã (visível na Figura 50) que o utilizador tem uma noção do andamento das recolhas em tempo real. Optou-se por uma interface extremamente simples para facilitar a visibilidade, onde tanto é possível obter o feedback através de barras como de percentagem.

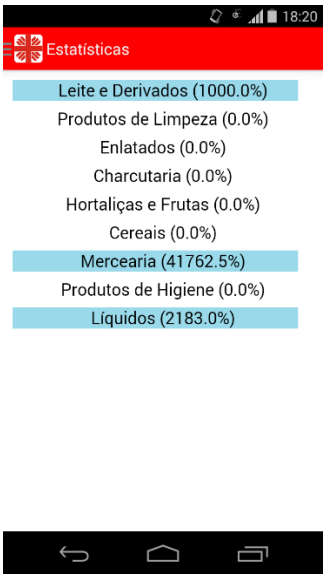



*Figura 50 - Ecrã com gráfico das estatísticas*

### 5.8.12. Estatísticas pessoais (12)

Resultado de uma filosofia de incentivo à competitividade e melhoria contínua dos participantes, este ecrã dá um feedback do desempenho do utilizador em comparação com os demais e o quão importante foi o seu contributo na submissão de registos.

Tal como se pode ver na Figura 51, está presente uma estatística de peso e número de produtos registados, contributo para a campanha e a respetiva posição num *ranking*.


*Figura 51 - Ecrã de estatísticas pessoais*

### 5.8.13. Sobre a aplicação (13) e Apoio (14)

Estes são 2 ecrãs igualmente sem uma utilidade funcional sobre os dados da aplicação, no entanto são de presença obrigatória para a mesma.

O ecrã de apresentação visível no primeiro ecrã da Figura 52 descreve o âmbito e parceria feita para a conceção deste projeto. Da mesma forma o ecrã de apoio da Figura 53 disponibiliza um numero e e-mail de apoio fornecidos pela própria Cáritas Diocesana de Coimbra para prestar esclarecimento aos utilizadores.

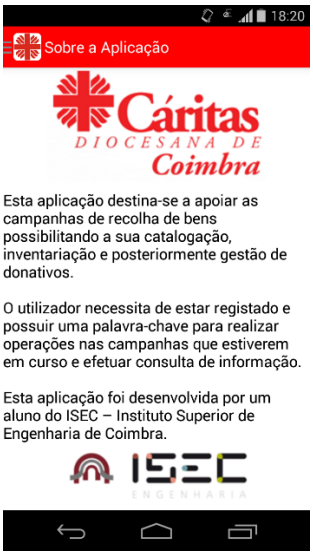



*Figura 52 - Ecrã de apresentação*


*Figura 53 - Ecrã de apoio*

# 6. Conclusões e Trabalho Futuro

Tal como se mostrou evidente em diversas partes, a elaboração deste projeto permitiu adquirir novos conhecimentos em diversas áreas, não só do ponto de vista de desenvolvimento de aplicações, mas também deu uma ideia do estado atual da tecnologia, deu a conhecer novos serviços de acesso a dados dos mais variados tipos e ajudou a conhecer melhor as instituições de solidariedade que atuam em território nacional.

Com um trabalho tão extenso, evidentemente há muito a dizer sobre diversos pontos, sendo estes discutidos nos subcapítulos seguintes.

## 6.1. Conclusões da Fase de Investigação

Uma das partes primárias deste projeto foi sem dúvida a fase de investigação, onde se pesquisaram soluções de leitura de códigos EAN, protocolos de comunicação focados em comunicações móveis, frameworks de alto desempenho e alta disponibilidade e bases de dados aplicáveis a estas mesmas tecnologias.

Efetuaram-se também diversos testes e usaram-se diversas frameworks e ferramentas, que embora maioritariamente não tenham sido usadas, permitiu adquirir novos conhecimentos e facilitar as decisões para projetos que eventualmente se venham a concretizar.

Desta fase resultou também uma comparação tecnológica que engloba um número de tecnologias superior ao que foi possível encontrar até ao momento da sua conceção, que embora não tenha sido fator único, foi uma das chaves importantes na decisão das tecnologias a usar no desenvolvimento deste projeto.

## 6.2. Conclusões do Desenvolvimento *Server-side*

Embora já possuindo uma boa experiência em programação Java, o desenvolvimento de toda a parte server-side foi sem dúvida algo completamente novo.

O modelo reativo e orientado a eventos do Vert.x dá lugar a uma forma particularmente diferente de conceber uma aplicação, onde o objetivo é processar um pedido o mais rápido possível e enviar eventos que despoletem mais ações ou simplesmente enviem uma resposta de volta. O EventBus foi também algo extremamente interessante, permitindo comunicar com diversas partes do software a partir de um meio de comunicação único.

Com esta nova filosofia de desenvolvimento de aplicações, há também o lado negativo. Uma nova filosofia que vai precisamente para o oposto do que se considera os modelos corretos, resultando em erros que requereram bastantes horas para se perceber, e muitas mais para replanear e corrigir: Instruções do género Thread.sleep() ou object.wait() resultam na completa paragem do ciclo de eventos, colocando o servidor parcialmente ou mesmo completamente inoperacional.

Embora o Vert.x tenha uma boa documentação e um bom conjunto de exemplos, faltam-lhe exemplos de operações mais elementares, tais como a criação de eventos de forma manual. A solução só foi possível consultando e percebendo algum do código fonte da framework Vert.x, o que é evidentemente uma tarefa morosa.

Outros problemas estiveram relacionados com os módulos usados:

- O módulo de acesso à base de dados, mais recente, disponibilizado através de Maven contém um bug em que não aceita um ID diferente do tipo String nos registos do MongoDB, os quais são guardados em formato ObjectID por padrão (tal como recomendado). Não foi possível encontrar muita ajuda na Internet, a única solução foi portanto usar a versão imediatamente anterior
- Os módulos de acesso a base de dados MySQL e PostgreSQL deram bastantes problemas, tinham muitas funcionalidades em falta e não estavam a receber participação suficiente do *developer* original nem da comunidade, o que impediu o uso de base de dados relacionais
- Embora o MongoDB suporte a ordenação de dados segundo um campo específico, esse ordenamento não tinha qualquer efeito através do módulo de acesso para Vert.x. Veio-se a descobrir que este bug já estava assinalado e marcado para resolução. Como o projeto não podia ficar pendente por esse problema, e dado que os dados que necessitavam de ordenação eram relativamente pequenos, optou-se pela construção de classes comparáveis para possibilitar o ordenamento através das próprias ferramentas do Java para o efeito
- Apesar do Vert.x e de boa parte dos seus módulos serem desenvolvidos em Java e esta ser a linguagem que executa com melhor desempenho dentro da framework Vert.x, bastante documentação e exemplos só estão disponíveis em JavaScript ou mais focadas nesta linguagem. Converter algum deste código para Java não foi fácil e requereu a consulta de código fonte para entender a origem de muitos problemas, a maioria causados por nomes diferentes em *handlers* de acesso, em relação ao que era previsível pela documentação

Mesmo com todos estes problemas, o desenvolvimento em Vert.x foi uma experiência bastante interessante e garantidamente será uma boa referência se no futuro houver necessidade de programar em outras frameworks que sigam igualmente o modelo reativo.

## 6.3. Conclusões do Desenvolvimento *Client-side*

O desenvolvimento da parte cliente em Android serviu não só para relembrar alguns conhecimentos já adquiridos em Arquiteturas Móveis da Licenciatura em Engenharia Informática e Computação Ubíqua, mas também para adquirir conhecimentos completamente novos de como fazer aplicações segundo os padrões atuais.

Pela primeira vez usou-se JSON para acesso a um serviço REST, o que facilitou imenso a troca de informação e permitiu testar muitas particularidades quanto á forma como os dados são encapsulados e reconvertidos.

A consulta de alguns exemplos disponíveis no *site* do próprio Android SDK [55] e a leitura de um livro dedicado precisamente à *User Interface* de Android [56], permitiu aprender algumas boas práticas a respeito de:

- *Design* da aplicação em geral e de acordo com regras uniformizadas
- Manter o esquema de navegação
- Facultar pesquisa embutida na própria *user interface*
- Comunicação Fragmento-Atividade
- Utilização de WebViews aliadas a JavaScript para criação de páginas dinâmicas como interface da aplicação
- Gestão de diferentes resoluções e aspetos de ecrãs
- Criação de temas gráficos
- Utilização de base de dados SQLite embutida nos próprios dispositivos

No entanto ainda existem algumas particularidades gráficas complexas, associadas às versões mais recentes do sistema Android, que embora não tenham afetado o trabalho, serão pertinentes para futura pesquisa, especialmente para uma futura para facultar suporte ao Android 5.0.

Outro problema foi a incapacidade de suporte a versões abaixo da versão Android 3.0 do Android, devido unicamente às bibliotecas externas já não serem compatíveis com as mesmas. Dado a constante migração para versões mais recentes do sistema operativo, aos poucos o suporte a versões mais antigas vai sendo abandonado.

Outro problema ainda mais evidente é a incompatibilidade com outros sistemas operativos não Android. Existem duas frameworks bastante promissoras neste campo, Cordova [40]e PhoneGap [41]. Estas permitem a construção de aplicações recorrendo a HTML, css, JavaScript e a chamadas a procedimentos facultados pela própria framework, permitindo assim abstrair a plataforma da aplicação. Esta solução só não foi adotada porque só se teve conhecimento tarde demais e o quase total desconhecimento de programação em JavaScript seriam um risco demasiado grande para se refazer a aplicação cliente do início.

[40] http://cordova.apache.org/ [Último acesso a 12 de Dezembro de 2014]
[41] http://phonegap.com/ [Último acesso a 12 de Dezembro de 2014]

## 6.4. Análise de Outros Pontos

A Cáritas Diocesana de Coimbra foi um grande parceiro deste projeto, que se ofereceu desde o início para funcionar como instituição piloto. Embora houvesse muita boa vontade, a conclusão de releases realmente funcionais e algumas avaliações coincidiram com o período de férias de vários funcionários, o que levou a alguns abrandamentos do desenvolvimento. Ainda como prova do interesse, no decorrer de uma palestra dedicada a inovação e empreendedorismo [61], a Caritas fez questão de apresentar este projeto, o qual foi muito bem recebido pelo público presente.

Antes de recorrer à Cáritas, o mesmo projeto tinha sido proposto ao Banco Alimentar. Este embora mostrando-se agradado, não mostrou interesse em aderir ao uso da aplicação. É compreensível, pois já dispunham de métodos manuais de funcionamento bem definidos, equipamento para fins semelhantes e a recolha de bens já é o seu foco principal há bastante tempo.

Outra coisa que não correu como desejado foi o acesso ao servidor, com interrupções constantes da rede do ISEC, que muitas vezes coincidiam com os testes que estavam em decurso, atrasando também a aprovação por parte da Cáritas Diocesana de Coimbra.

## 6.5. Trabalho Futuro

Embora este trabalho associado à disciplina de Projeto Industrial esteja concluído, não significa que todo o trabalho desenvolvido vá simplesmente parar.

O trabalho de investigação foi de extrema importância para conhecer novas frameworks e poder experimentar a sua aplicação na prática, no entanto muitas destas encontram-se em grandes mudanças. O espaço temporal entre o fim da fase de investigação e o fim da fase de implementação foi suficiente para o aparecimento de novas versões. Seria por isso interessante repetir os testes e verificar se os resultados ainda se mantêm, ou se por outro lado houve melhorias significativas que compensem uma escolha diferente para um projeto futuro semelhante.

Referente à aplicação desenvolvida, a Cáritas Diocesana de Coimbra tem vindo a mostrar interesse em alargar o âmbito do projeto e adicionar novas funcionalidades, nomeadamente no registo de doações monetárias. Esta será portanto uma funcionalidade desenvolvida já fora de âmbito de projeto a relativamente curto prazo.

Aquando da finalização da release 5, observaram-se 3 oportunidades de publicar e disponibilizar este projeto para o público no geral:

- Concurso de desenvolvimento para a plataforma eVida
- Publicação no *website* Cidadania 2.0
- Submissão de um artigo para a conferência ISAmI 2015 (International Symposium Ambient Intelligence)

Embora já fora da data de escrita deste relatório, estes pontos serão discutidos em anexo assim que tal for possível.

Também no seguimento deste projeto, estão planeados 2 projetos de licenciatura. Um para a construção da interface de administração, usando linguagens que simplifiquem o desenvolvimento Web e façam a passassem de dados da base de dados MongoDB para uma base de dados relacional, facilitando assim a análise e relacionamento de dados. Outro para implementar a funcionalidade de registo de doações monetárias e aprofundar o aspeto de análise de dados.

Outra possivel proposta para futuro seria investigar a possibilidade do desenvolvimento de uma aplicação cliente usando a framework Cordova ou PhoneGap, permitindo assim, com uma única aplicação, abranger dispositivos de outras arquiteturas e de diferentes sistemas operativos (p.ex. iPhone, iTab e Windows Phone).

# 7. Referências Bibliográficas

[1] R. K. Ganti, F. Ye e H. Lei, “Mobile crowdsensing: current state and future challenges,” *IEEE Communications Magazine,* nº 1, 2011.

[2] A. Vaccari, L. Liang, A. Biderman, C. Ratti, F. Pereira, J. Oliveirinha e A. Gerber, “A holistic framework for the study of urban traces and the profiling of urban processes and,” 2009.

[3] F. Girardin, A. Vaccari, A. Gerber, A. Biderman e C. Ratti, “Quantifying urban attractiveness from the distribution and density of digital footprints,” 2009.

[4] F. Rodrigues, A. Alves, E. Polisciuc, S. Jiang, J. Ferreira e F. C. Pereira, “Estimating disaggregated employment size from Points-of-Interest and census data: From mining the web to model implementation and visualization,” *International Journal on Advanced Intelligent Systems,* vol. 7, 2013.

[5] Cáritas Diocesana de Coimbra, “Identidade,” 2011. [Online]. Available: http://www.caritas.pt/site/coimbra/index.php?option=com_content&view=article&id=3734&Itemid=94. [Acedido em 11 Dezembro 2014].

[6] Google, “History,” Google, 2014. [Online]. Available: http://www.android.com/history/. [Acedido em 12 Dezembro 2014].

[7] N. Gok e N. Khanna, Cover image for Building Hybrid Android Apps with Java and JavaScript, O'Reilly Media, Inc., 2013.

[8] Xamarin, “Location Services,” 2014. [Online]. Available: http://developer.xamarin.com/guides/android/platform_features/maps_and_location/location/. [Acedido em 12 Dezembro 2014].

[9] O. Attia, P. Shrivastava, R. Zastepine e A. C. Outmezguine, “System and method for decoding and analyzing barcodes using a mobile device”. US Patente US7287696 B2, 30 October 2007.

[10] G. R. Narayan e M. V. James, “Barcode Recognition from Video by Combining Image Processing and Xilinx,” *Procedia Engineering,* nº 38, 2012.

[11] H. K. Fröschle, U. Gonzales-Barron, K. McDonnell e S. Ward, “Investigation of the potential use of e-tracking and tracing of poultry using linear and 2D barcodes,” *Computers and Electronics in Agriculture,* nº 66, 2009.

[12] Y.-C. Lin, W.-F. Cheung e F.-C. Siao, “Developing mobile 2D barcode/RFID-based maintenance management system,” *Automation in Construction,* nº 37, 2013.

[13] B. d. C. Rocha e J. M. G. Saias, “Implementação de um sistema de inventariação com arquitectura distribuída e visão computacional,” Universidade de Évora, Évora, Portugal, 2013.

[14] M. B. Juric, I. Rozman e M. Hericko, “Performance comparison of CORBA and RMI,” *Information and Software Technology,* nº 42, 2000.

[15] M. B. Juric, I. Rozman, B. Brumen, M. Colnaric e M. Hericko, “Comparison of performance of Web services, WS-Security, RMI, and RMI–SSL,” nº 79, 2006.

[16] L. Dol, “JSON Vs. XML for Data Interchange,” 5 Março 2011. [Online]. Available: http://softwaremonkey.org/article/computer/Json-Vs-Xml. [Acedido em 14 Dezembro 2014].

[17] R. S. I. Matsushita e D. D. Nguessan, “Framework para Integração de Serviços Móveis Baseados em Rede Social,” *Fasci-Tech,* vol. 1, nº 5, 2011.

[18] M. Fasel, “Performance Comparison Between Node.js and Java EE For Reading JSON Data from CouchDB,” shine technologies, 2013. [Online]. Available: http://blog.shinetech.com/2013/10/22/performance-comparison-between-node-js-and-java-ee/. [Acedido em 14 Dezembro 2014].

[19] W. Seongmin, “http://www.cubrid.org/blog/dev-platform/inside-vertx-comparison-with-nodejs/,” CUBRID, 8 4 2013. [Online]. Available: http://www.cubrid.org/blog/dev-platform/inside-vertx-comparison-with-nodejs/. [Acedido em 11 Dezembro 2014].

[20] T. Parviainen, Real-time Web Application Development using Vert.x 2.0, Packt Publishing, 2013.

[21] C. P. Bineytioglu, J. A. Boden, R. Schulz, M. Vökler e R. Wawrzyniak, “Evaluation of Asynchronous Server,” *KOS.content,* nº 1, 2013.

[22] K. K.-Y. Lee, W.-C. Tang e K.-S. Choi, “Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage,” *Computer Methods and Programs in Biomedicine,* nº 110, 2013.

[23] J. van der Veen, B. van der Waaij e R. Meijer, “Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual,” *CLOUD 2012: 2012 IEEE 5th International Conference on Cloud Computing,* 2012.

[24] P. Atzeni, F. Bugiotti e L. Rossi, “Uniform access to NoSQL systems,” nº 43, 2014.

[25] V. Abramova e J. Bernardino, “NoSQL Databases: MongoDB vs Cassandra,” *Proceedings of the International C* Conference on Computer Science and Software Engineering,* 2013.

[26] K. Kovacs, “Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase vs Couchbase vs OrientDB vs Aerospike vs Neo4j vs Hypertable vs ElasticSearch vs Accumulo vs VoltDB vs Scalaris comparison,” [Online]. Available: http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis. [Acedido em 11 Dezembro 2014].

[27] A. C. Carniel, A. A. Sá, M. X. Ribeiro, R. Bueno, C. D. d. A. Ciferri e R. R. Ciferri, “Análise Experimental de Bases de Dados Relacionais e NoSQL,” *Simpósio Brasileiro de Bancos de Dados,* 2012.

[28] R. M. Toth, “Abordagem NoSQL – uma real alternativa”.

[29] R. Cattell, “Scalable SQL and NoSQL Data Stores,” *SIGMOD Record,* vol. 39, nº 4, 2010.

[30] B. G. Tudorica e C. Bucur, “A comparison between several NoSQL databases with comments and notes,” nº 10, 2011.

[31] A. B. M. Moniruzzaman e S. A. Hossain, “NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison,” *International Journal of Database Theory and Application,* vol. 6, nº 4, 2013.

[32] J. S. v. d. Veen, B. v. d. Waaij e R. J. Meijer, “Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual,” nº 18, 2012.

[33] E. Redmond e J. R. Wilson, Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement, Pragmatic Bookshelf, 2012.

[34] D. McCreary e A. Kelly, Making Sense of NoSQL: A guide for managers and the rest of us, Manning Publications, 2013.

[35] N. Graham, “Retail Reference Architecture: Real-Time, Geo-Distributed inventory,” MongoDB, 2014. [Online]. Available: http://pt.slideshare.net/mongodb/retail-reference-architecture-part-2-realtime-geo-distributed-inventory. [Acedido em 14 Dezembro 2014].

[36] Y. Finkelstein, “MongoDB at eBay,” MongoDB, 4 Maio 2012. [Online]. Available: http://www.mongodb.com/presentations/mongodb-ebay. [Acedido em 14 Dezembro 2014].

[37] T. Bley, “Mass inserts, updates: SQLite vs MySQL (update: delayed inserts),” 12 Setembro 2012. [Online]. Available: http://we-love-php.blogspot.pt/2012/08/mass-inserts-updates-sqlite-vs-mysql.html. [Acedido em 11 Dezembro 2014].

[38] O. S. Tezer, “SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems,” DigitalOcean, 21 Fevereiro 2014. [Online]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-d%20atabase-management-systems. [Acedido em 14 Dezembro 2014].

[39] GlobalGiving Foundation, “GlobalGiving.co.uk: donate to projects around the world supporting disaster relief, education, health, women and children, and more,” 2013. [Online]. Available: http://www.globalgiving.co.uk/. [Acedido em 11 Dezembro 2014].

[40] Everyclick Ltd, “Raise money for charity with Everyclick,” 2014. [Online]. Available: http://www.everyclick.com/. [Acedido em 11 Dezembro 2014].

[41] GoFundMe, “Charity Donations & Charity Fundraising Websites,” 2014. [Online]. Available: http://www.gofundme.com/charity-donations. [Acedido em 11 Dezembro 2014].

[42] JustGive, “JustGive.org - JustGive - The Destination for Online Charitable Giving,” 2014.

[Online]. Available: https://www.justgive.org/. [Acedido em 11 Dezembro 2014].

[43] St. Mary's Food Bank Alliance, "The world's first food bank," [Online]. Available: http://www.firstfoodbank.org/. [Acedido em 11 Dezembro 2014].

[44] Community Action Partnership of Kern, "Food Bank," [Online]. Available: http://www.capk.org/index.cfm/fuseaction/Pages.Page/id/538. [Acedido em 11 Dezembro 2014].

[45] Dell Inc., "Dell Employee Volunteer," 2014. [Online]. Available: https://play.google.com/store/apps/details?id=com.dell.yourcause. [Acedido em 11 Dezembro 2014].

[46] RazRon, "Charitable Donations Log Lite," 2013. [Online]. Available: https://play.google.com/store/apps/details?id=raj.rohit.android.cdlLite. [Acedido em 11 Dezembro 2014].

[47] Cerulean Tech Creations, "Donations," [Online]. Available: http://www.ceruleantechcreations.com/donations.php. [Acedido em 11 Dezembro 2014].

[48] Roadrunner Food Bank, [Online]. Available: http://www.rrfb.org/. [Acedido em 11 Dezembro 2014].

[49] Wood Buffalo Food Bank, [Online]. Available: http://woodbuffalofoodbank.com.s178952.gridserver.com/. [Acedido em 11 Dezembro 2014].

[50] Redemption Media Ltd, [Online]. Available: http://www.foodbankapp.co.uk/. [Acedido em 11 Dezembro 2014].

[51] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. [Acedido em 11 Dezembro 2014].

[52] M. Samek, "Who Moved My State?," 1 Abril 2003. [Online]. Available: http://www.drdobbs.com/who-moved-my-state/184401643. [Acedido em 11 Dezembro 2014].

[53] E. Freeman, B. Bates, K. Sierra e E. Robson, Head First Design Patterns, O'Reilly Media, 2004.

[54] T. Fox e N. Maurer, "Vert.x Documentation," 2014. [Online]. Available: http://vertx.io/docs.html. [Acedido em 11 Dezembro 2014].

[55] Google Inc, "Design | Android Developers," 2014. [Online]. Available: https://developer.android.com/design/index.html. [Acedido em 11 Dezembro 2014].

[56] D. Pacholczyk, Mobile UI Design Patterns 2014, UXPin, 2014.

[57] S. Hoberman, Data Modeling for MongoDB, Technics Publications, 2014.

[58] K. Banker, MongoDB in Action, Manning Publications, 2011.

[59] OpenCellID, “OpenCellID - OpenCellID,” 2014. [Online]. Available: http://opencellid.org/. [Acedido em 11 Dezembro 2014].

[60] Google Inc, “The Google Maps Geolocation API - Google Maps API for Work,” 2014. [Online]. Available: https://developers.google.com/maps/documentation/business/geolocation/. [Acedido em 11 Dezembro 2014].

[61] CITEK, Interviewee, *Your innovation day: Other ways for innovation and entrepreneurship.* [Entrevista]. 19 Setembro 2014.

[62] H. Rheingold, Smart Mobs: The Next Social Revolution. New York: Basic Books, 2002.

[63] J. B. e. al, ““Participatory sensing”, Workshop on World- Sensor-Web, co-located with ACM SenSys,” 2006. [Online]. Available: http://www.sensorplanet.org/wsw2006.

[64] “Collective Mobility Patterns from Smart Card Records: A Case Study in Shenzhen,” St. Louis, MO, USA, 2009.

[65] J. Rouillard, “Contextual QR Codes,” Athens, Greece, 2008.

[66] C. Pautasso, O. Zimmermann e F. Leymann, “Restful web services vs. "big" web services: making the right architectural decision,” 2008.

# Anexos

## A1. Mockups da aplicação cliente

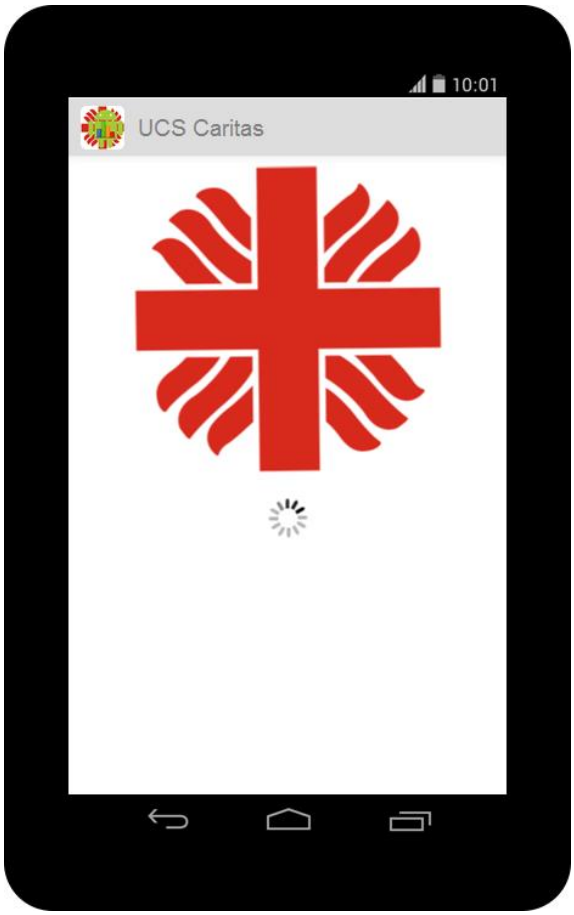



*Figura 54 - Ecrã de arranque da aplicação*

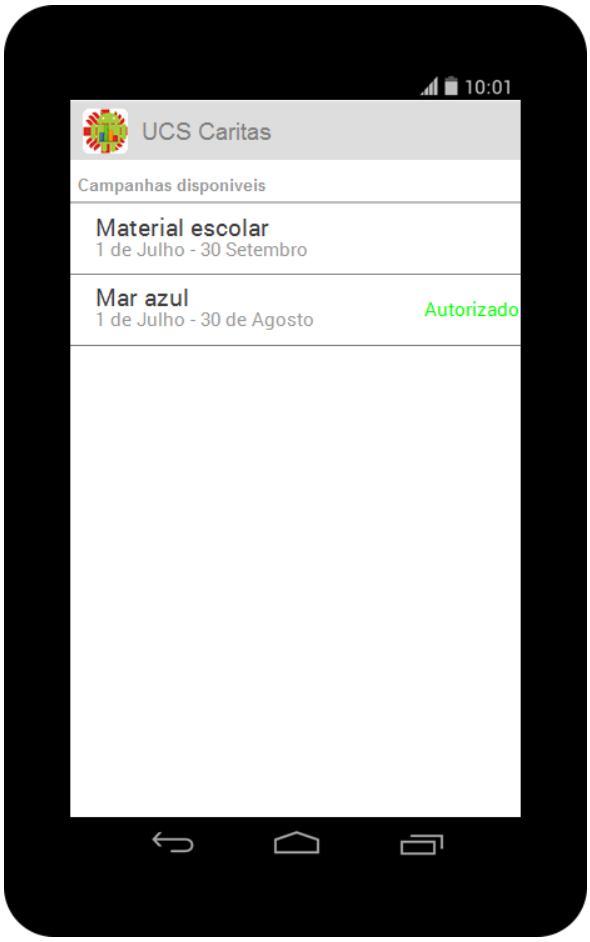



*Figura 55 - Ecrã de seleção de campanha*

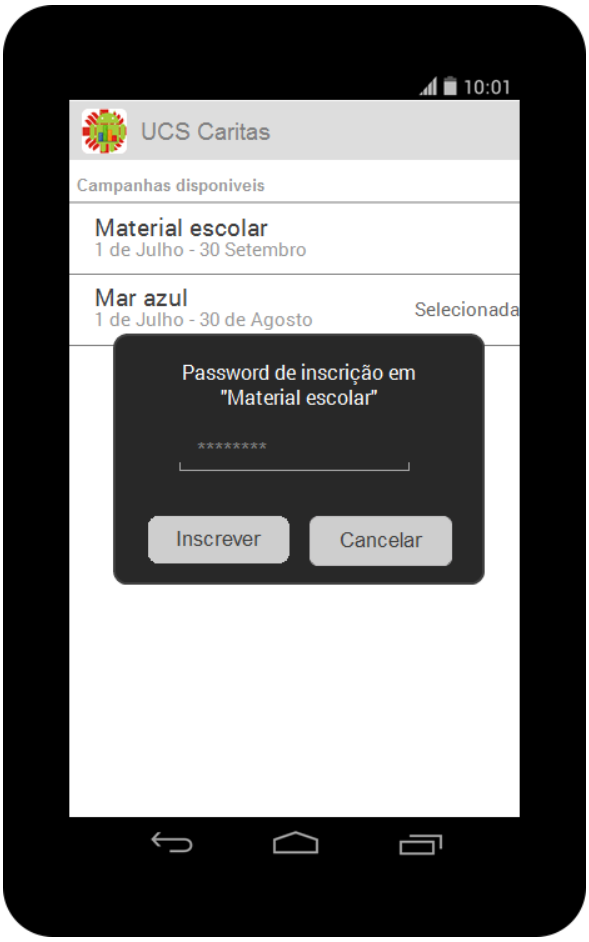



*Figura 56 - Introdução de password na inscrição a uma nova campanha*

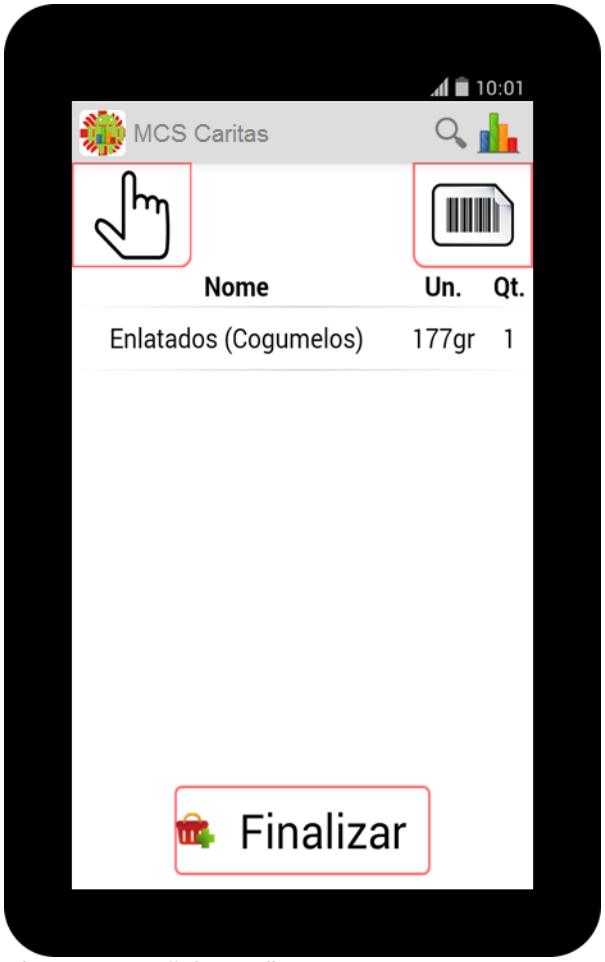



*Figura 57 - Ecrã de recolha*


*Figura 58 - Alternativa de cor no ecrã de recolha*


*Figura 59 - Alternativa de posicionamento no ecrã de recolha*

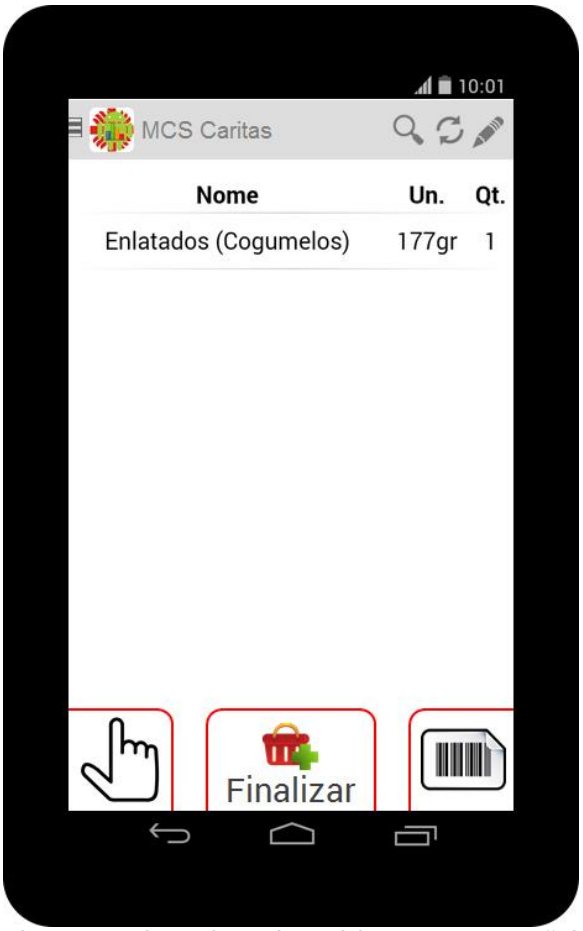



*Figura 60 - Alternativa 2 de posicionamento no ecrã de recolha*

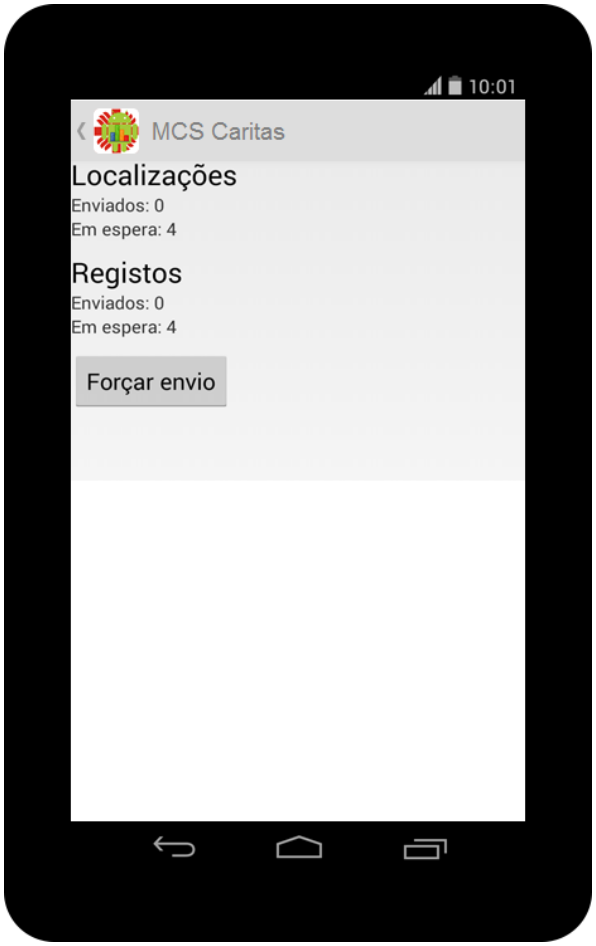



*Figura 61 - Ecrã de Informações*

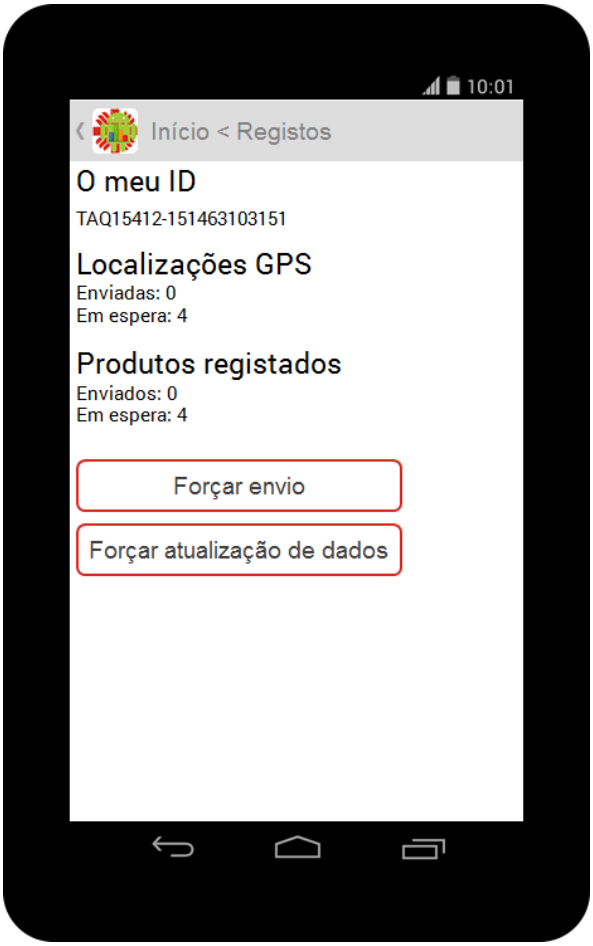



*Figura 62 - Ecrã de Informações alternativo*

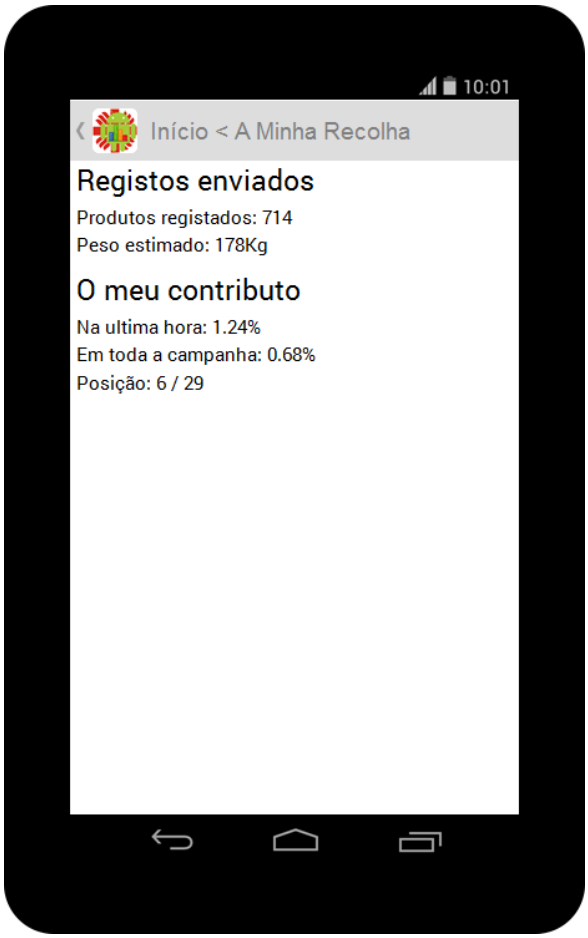



*Figura 63 - Ecrã de estatísticas pessoais*


*Figura 64 - Ecrã de estatísticas pessoais alternativo*


*Figura 65 - Menu de navegação lateral*


*Figura 66 - Menu de navegação lateral com cor alternativa*


*Figura 67 - Ecrã de introdução manual*


*Figura 68 - Ecrã de introdução manual com uma categoria selecionada*

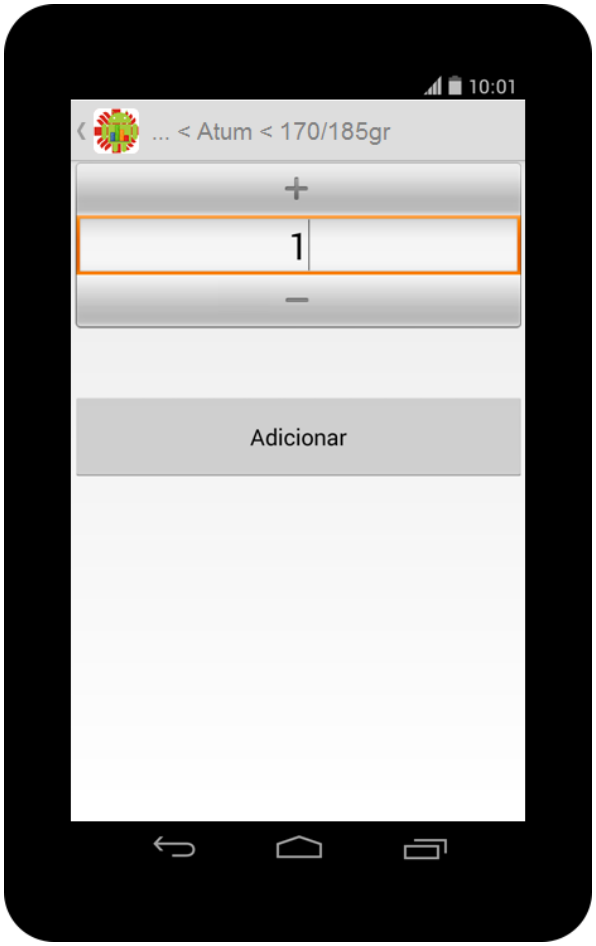



*Figura 69 - Ecrã de escolha de quantidade na introdução manual*

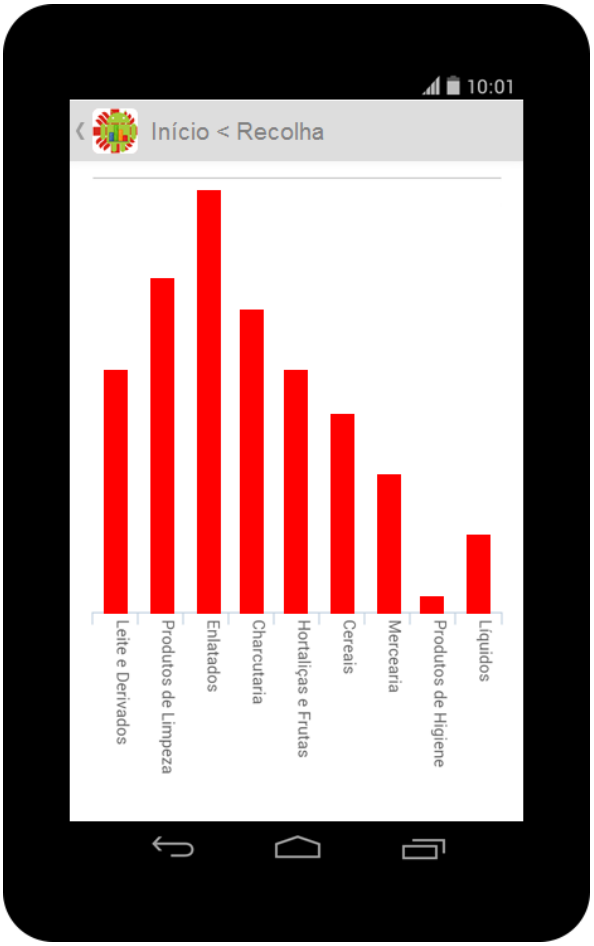



*Figura 70 - Ecrã de estatísticas*


*Figura 71 - Ecrã de introdução agrupada vazio*


*Figura 72 - Ecrã de introdução agrupada com produtos*

# A2. Ficheiros ou dados JSON

## A2.1. Exemplo de registo não agrupado enviado


```
{
  "idCampaign":"b0ac7925-ff54-4366-a2bd-09a548caeec4",
  "registries":[
    {
      "date":1418914423098,
      "products":[
        {
          "ean":"20203429",
          "quantity":1
        },
        {
          "ean":"5601312043348",
          "quantity":1
        },
        {
          "unit":"un",
          "quantity":1,
          "weight":1,
          "type":"b-6",
          "realWeight":1
        },
        {
          "unit":"un",
          "quantity":1,
          "weight":1,
          "type":"d-9",
          "realWeight":1
        }
      ],
      "location":{
        "date":0,
        "lat":40.1936647,
        "lon":-8.3966104,
        "src":"last"
      }
    }
  ]
}
```

## A2.2. Exemplo de registo agrupado enviado


```
{
  "idCampaign":"a702b35f-10d9-45da-8d67-75e88c3fc63f",
  "idDevice":"53dd2ad887fa7085-TA929042MQ",
  "date":1416984002377,
  "dateReceived":1416984006736,
  "location":{
    "date":0,
    "lon":-8.412388883605482,
    "lat":40.24879411029594,
    "src":"last",
    "mcc":268,
    "mnc":1,
    "lac":18,
    "cid":17782
  },
  "massInput":true,
  "type":"b-41",
  "weight":300,
  "unit":"gr",
  "minWeight":300,
  "maxWeight":300,
  "realWeight":300,
  "products":[
    {
      "ean":"5601069004517",
      "quantity":100
    },
    {
      "ean":"20203429",
      "quantity":1
    },
    {
      "ean":"5601312043348",
      "quantity":1
    }
  ]
}
```

## A2.3. Exemplo de documentos das collections da base de dados

**campaign.json**


```
{
  "_id" : "98b81029-72ef-4270-93cc-4fab0985d19a",
  "idCampaign" : "a702b35f-10d9-45da-8d67-75e88c3fc63f",
  "name" : "Biscoito",
  "password" : "123",
  "warehouse" : "Outro",
  "dateStart" : NumberLong("1399849200000"),
  "dateEnd" : NumberLong("1418515199999"),
  "visible" : true,
  "numDevices" : 3,
  "objectives" : {
    "a" : 10000,
    "b" : 10000,
    "c" : 10000,
    "d" : 10000,
    "e" : 10000,
    "f" : 10000,
    "g" : 10000,
    "h" : 10000,
    "i" : 10000
  },
  "stats" : {
    "a" : 10000,
    "b" : 417625,
    "c" : 0,
    "d" : 0,
    "e" : 0,
    "f" : 21830,
    "g" : 0,
    "h" : 0,
    "i" : 0,
    "b-3" : 3000,
    "f-3" : 21830,
    "b-14" : 125,
    "b-11" : 4500,
    "a-2" : 10000,
    "b-1" : 410000
  },
  "detailedStats" : {
    "1416603600000-1416607199999" : {
      "b" : 3000,
      "b-3" : 3000
    },
    "1416664800000-1416668399999" : {
      "f" : 6000,
```

```
      "f-3" : 6000,
      "b" : 125,
      "b-14" : 125
    },
    "1416841200000-1416844799999" : {
      "b-11" : 4500,
      "b" : 4500,
      "f" : 330,
      "f-3" : 330
    },
    "1417096800000-1417100399999" : {
      "a" : 10000,
      "a-2" : 10000,
      "b-1" : 410000,
      "b" : 410000
    },
    "1417690800000-1417694399999" : {
      "f" : 15500,
      "f-3" : 15500
    }
  },
  "sumWeights" : NumberLong(898910),
  "sumRegistries" : 1102,
  "devices" : ["53dd2ad887fa7085-TA929042MQ", "d4d319a79746e0bc-
0601bc2300511c48", "46a97885ae5989d5-unknown"]
}
```

## A2.4. Exemplo de conteúdos da base de dados

**devices.json**


```
{
  "_id" : "6b00e63d-51b1-4ced-a00f-54171e13648d",
  "idDevice" : "d4d319a79746e0bc-0601bc2300511c48",
  "isblocked" : false,
  "ismobile" : true,
  "lastLocation" : {
    "date" : NumberLong("1417262239510"),
    "mcc" : 268,
    "mnc" : 3,
    "lac" : 255,
    "cid" : 350487
  },
  "locations" : [{
      "date" : 0,
      "lon" : -8.32293598,
      "lat" : 40.30468414,
      "src" : "fast",
      "mcc" : 268,
      "mnc" : 3,
      "lac" : 255,
      "cid" : 145675
    }, {
      "date" : 0,
      "lon" : -8.27548884,
      "lat" : 40.27638446,
      "src" : "fast",
      "mcc" : 268,
      "mnc" : 3,
      "lac" : 55000,
      "cid" : 5218935
    }, {
      "date" : NumberLong("1416844163187"),
      "lon" : -8.4127556,
      "lat" : 40.192289,
      "src" : "last",
      "mcc" : 268,
      "mnc" : 3,
      "lac" : 255,
      "cid" : 143126
    }, {
```

```
      "date" : NumberLong("1417262239510"),
      "mcc" : 268,
      "mnc" : 3,
      "lac" : 255,
      "cid" : 350487
    }],
  "campaigns" : ["a702b35f-10d9-45da-8d67-75e88c3fc63f"]
}
```


**personalStats.json**


```
{
  "_id" : "484c331b-7b35-4330-b744-942ae398ee4e",
  "idCampaign" : "a702b35f-10d9-45da-8d67-75e88c3fc63f",
  "idDevice" : "d4d319a79746e0bc-0601bc2300511c48",
  "sumWeights" : 892910,
  "sumRegistries" : 1099
}
```


**conflicts.json**


```
{
  "_id" : ObjectId("5441cb42f548c30d042ac942"),
  "ean" : "123456789",
  "resolved" : true,
  "entries" : [{
      "type" : "a-1",
      "weight" : 100,
      "unit" : "gr",
      "realWeight" : 100,
      "modified" : NumberLong("1413597575000"),
      "name" : "Iogurte (100gr)",
      "owner" : "asdfgh"
    }, {
      "type" : "a-1",
      "weight" : 200,
      "unit" : "gr",
      "realWeight" : 200,
      "modified" : NumberLong("1413597786000"),
      "name" : "Iogurte (200gr)",
      "owner" : "qwerty"
    }],
  "choosed" : 1
}
```

**products.json**


```
{
  "_id" : "05aa07aa-405b-4ef0-9258-fd03e0d2d5f3",
  "ean" : "5601227022742",
  "type" : "f-3",
  "weight" : 330,
  "unit" : "ml",
  "realWeight" : 330,
  "modified" : NumberLong("1416844389315"),
  "name" : "Sumos (L&iacute;quidos)",
  "src" : "mobile",
  "idDevice" : "d4d319a79746e0bc-0601bc2300511c48",
  "idCampaign" : "Campanha Teste"
}
```

**registries.json**


```
{
  "_id" : "90ccb0a7-c52f-4935-9f18-45170ed19284",
  "idCampaign" : "9a744f22-37d8-473f-82f5-8f16c2982b40",
  "idDevice" : "829237a75f0f9e6e-e4ddf26b",
  "date" : NumberLong("1417794027107"),
  "dateReceived" : NumberLong("1417794499353"),
  "location" : {
    "date" : NumberLong("1417794027107"),
    "lon" : -8.399662,
    "lat" : 40.197198,
    "src" : "fast",
    "mcc" : 268,
    "mnc" : 3,
    "lac" : 255,
    "cid" : 47125
  },
  "products" : [{
      "ean" : "5607684019257",
      "quantity" : 4
    }, {
      "type" : "a-1",
      "weight" : 112,
      "unit" : "gr",
      "realWeight" : 112,
      "quantity" : 10
    }, {
      "type" : "g-3",
      "weight" : 200,
      "unit" : "ml",
      "realWeight" : 200,
      "quantity" : 3
    }, {
      "type" : "f-3",
      "weight" : 200,
      "unit" : "ml",
      "realWeight" : 200,
      "quantity" : 1000
    }]
}
```

**unknownProducts.json**


```
{
  "_id" : ObjectId("5481c32bfccb6b3477194cf0"),
  "ean" : "5607684019257",
  "quantity" : 5,
  "date" : NumberLong("1417789861582"),
  "idDevice" : "829237a75f0f9e6e-e4ddf26b",
  "idCampaign" : "9a744f22-37d8-473f-82f5-8f16c2982b40",
  "location" : {
    "date" : NumberLong("1417789861582"),
    "lon" : -8.399662,
    "lat" : 40.197198
  }
}
```


**types.json**


```
{
  "_id" : ObjectId("537ab8e2f548c31428f6a10a"),
  "symbol" : "a",
  "name" : "Leite e Derivados",
  "subtypes" : [{
      "id" : 0,
      "name" : "Outros"
    }, {
      "id" : 1,
      "name" : "Iogurtes"
    }, {
      "id" : 2,
      "name" : "Leite"
    }, {
      "id" : 3,
      "name" : "Leite com chocolate"
    }, {
      "id" : 4,
      "name" : "Leite condensado"
    }, {
      "id" : 5,
      "name" : "Leite em p&oacute;"
    }, {
      "id" : 6,
      "name" : "Manteiga"
    }]
}
```

### Qualquer collection de celltowersids


```
{
  "_id" : "ad679f06-1031-4068-a0a1-31ad840008e6",
  "mcc" : 268,
  "mnc" : 3,
  "lac" : 55000,
  "cid" : 5203435,
  "lon" : -8.441294,
  "lat" : 40.173265,
  "nUsed" : 3
}
```

## A2.5. Ficheiro de conteúdo dos ecrãs de introdução manual ("screen.json")


```
{
  "main":[
    {
      "name":"Leite e &#13;&#10;derivados",
      "symbol":"a"
    },
    {
      "name":"Mercearia",
      "symbol":"b"
    },
    {
      "name":"Cereais",
      "symbol":"c"
    },
    {
      "name":"Hortali&ccedil;as&#13;&#10;e Frutas",
      "symbol":"d"
    },
    {
      "name":"Enlatados",
      "symbol":"e"
    },
    {
      "name":"L&iacute;quidos",
      "symbol":"f"
    },
    {
      "name":"Produtos&#13;&#10;de&#13;&#10;Higiene",
      "symbol":"g"
    },
    {
      "name":"Produtos&#13;&#10;de&#13;&#10;Limpeza",
      "symbol":"h"
    },
    {
      "name":"null"
    },
    {
      "name":"Charcutaria",
      "symbol":"i"
```

```
    }
  ],
  "subtypes":{
    "a":[
      {
        "name":"Iogurtes",
        "symbol":"a-1"
      },
      {
        "name":"Leite",
        "symbol":"a-2"
      }
    ],
    "b":[
      {
        "name":"Achocolatado",
        "symbol":"b-1"
      },
      {
        "name":"A&ccedil;ucar&#13;&#10;amarelo",
        "symbol":"b-2"
      }
    ]
  },
  "quantities":{
    "a-1":[
      {
        "name":"40/50gr",
        "value":45,
        "unit":"gr",
        "minWeight":40,
        "maxWeight":50,
        "realWeight":45
      },
      {
        "name":"100/125gr",
        "value":112,
        "unit":"gr",
        "minWeight":100,
        "maxWeight":125,
        "realWeight":112
      }
```

```
    ],
    "a-2":[
      {
        "name":"200ml",
        "value":200,
        "unit":"ml",
        "minWeight":200,
        "maxWeight":200,
        "realWeight":200
      },
      {
        "name":"700ml",
        "value":700,
        "unit":"ml",
        "minWeight":700,
        "maxWeight":700,
        "realWeight":700
      },
      {
        "name":"1L",
        "value":1000,
        "unit":"ml",
        "minWeight":1000,
        "maxWeight":1000,
        "realWeight":1000
      },
      {
        "name":"1.5L",
        "value":1500,
        "unit":"ml",
        "minWeight":1500,
        "maxWeight":1500,
        "realWeight":1500
      }
    ]
  }
}
```

# A3. Lista de Bens (Categorias e Subcategorias)

| Leite e Derivados | Quantidades | | | | | | |
|---|---|---|---|---|---|---|---|
| Iogurtes | 40/50gr | 100/125gr | | | | | |
| Leite | 200ml | 700ml | 1L | 1,5L | | | |
| leite com chocolate | 200ml | 250ml | | | | | |
| Leite condensado | 370/397gr | | | | | | |
| Leite em pó | 200gr | 400gr | 750/800gr | 900gr | 1Kg | | |
| Manteiga | 125gr | 250gr | 500gr | | | | |

| Mercearia | Quantidades | | | | | | |
|---|---|---|---|---|---|---|---|
| Achocolatado | 250/300gr | 360/460gr | 800/900gr | | | | |
| Açucar amarelo | 1Kg | | | | | | |
| Açucar branco | 1Kg | | | | | | |
| Açucar caramelizado | 400gr | | | | | | |
| Amêndoas | 180/200gr | | | | | | |
| Bacalhau | 1un | 400/700gr | | | | | |
| Batata frita | 80/120gr | 150/200gr | 300/500gr | | | | |
| Bolachas | 80/100gr | 200gr | 260gr | 300/400gr | 450/500gr | 600gr | |
| Bollycao | 50/60gr | | | | | | |
| Bolo rei | 1un | | | | | | |
| Café | 80/100gr | 200gr | 250gr | | | | |
| Caldos /Temperos | 8/16 un | 24/30 un | 20/49gr | | | | |
| Chás | 10/20 un | | | | | | |
| Chocolate barra | 100/200gr | 400gr | 4/16 un | | | | |
| Chocolate pó | 125gr | | | | | | |
| Coco ralado | 200gr | | | | | | |
| Compotas | 230/375gr | 450/500gr | | | | | |
| Creme chocolate barrar | 200gr | 400gr | 750gr | | | | |
| Cremes/ sopas | 50/75gr | 60/80gr | | | | | |
| Enfeites bolo | 0,05Kg | | | | | | |
| Especiarias | 6/48gr | 50/100gr | | | | | |
| Feijão branco pacote | 500gr | 1Kg | | | | | |
| Feijão encarnado pacote | 500gr | 1Kg | | | | | |
| Feijão frade pacote | 500gr | 1Kg | | | | | |
| Feijão preto pacote | 500Gr | | | | | | |
| Fermento em pó | 113gr | 226gr | | | | | |
| Gelatinas | 80/100gr | 100/170gr | | | | | |
| Grão pacote | 500gr | 1Kg | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Ketchup | 225/500gr | | | | | | |
| Maionese | 225/500gr | | | | | | |
| Marmelada | 250gr | 350gr | 400/450gr | 800/900gr | | | |
| Mel | 150gr | 350/500gr | | | | | |
| Mini fofos | 250/300gr | | | | | | |
| Mostarda | 190/250gr | | | | | | |
| Mousse | 150gr | | | | | | |
| Ovos | 6 un | 12 un | 24 un | | | | |
| Polpa Tomate | 210/550gr | 1Kg | | | | | |
| Queques/ madalenas | 250/300gr | 600gr | | | | | |
| Sal | 250gr | 1Kg | | | | | |
| Sobremesas/pudins | 20/23gr | 80/90gr | 111/200gr | | | | |
| Tortas/bolo | 300gr | 400gr | | | | | |
| Waffle | 165gr | | | | | | |

| Cereais | Quantidades | | | | | | |
|---|---|---|---|---|---|---|---|
| Arroz | 175gr | 250gr | 325gr | 500gr | 1Kg | 2,5Kg | 5K |
| Cereais/ Nestum | 300gr | 360/375gr | 400gr | 450gr | 500gr | 600gr | 750gr |
| Farinha | 200/291gr | 400/450gr | 500gr | 700gr | 1Kg | | |
| Farinha láctea/ Papas | 200gr | 250/300gr | 500/600gr | 750gr | 1Kg | | |
| Massa | 330/375gr | 500gr | 1Kg | | | | |
| Pão ralado | 200gr | | | | | | |
| Pevide | 250gr | | | | | | |
| Pipocas | 85gr | 250/400gr | | | | | |
| Tostas | 100/300gr | | | | | | |

| Hortaliças e Frutas | Quantidades | | | | | | |
|---|---|---|---|---|---|---|---|
| Abóboras | 1 un | | | | | | |
| Alho francês | 1 un | | | | | | |
| Alhos | 1 un | | | | | | |
| Amendoim | 100/250gr | 500gr | | | | | |
| Batatas | 2,5/3Kg | 5Kg | 10Kg | 20Kg | | | |
| Cebolas | 2Kg | | | | | | |
| Cenoura | 1Kg | | | | | | |
| Courgete | 1Kg | | | | | | |
| Couves | 1un | | | | | | |
| Figos | 500gr | | | | | | |
| Kiwi | 2Kg | | | | | | |
| Laranjas | 2Kg | | | | | | |
| Limão | 1Kg | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Maças | 2Kg | | | | | | |
| Nabos | 1Kg | | | | | | |
| Nozes | 1Kg | | | | | | |
| Peras | 2Kg | | | | | | |
| Romã | 1Kg | | | | | | |
| Tangerinas | 2Kg | | | | | | |
| Tremoço | 500gr | | | | | | |

| Enlatados | Quantidades | | | | | | |
|---|---|---|---|---|---|---|---|
| Atum | 120gr | 200gr | 385gr | | | | |
| Boião de fruta | 1 un | 100/130gr | | | | | |
| Chispalhada | 420gr | | | | | | |
| Cogumelos | 170/185gr | 280/355gr | 780gr | | | | |
| Dobrada | 420gr | | | | | | |
| Ervilhas | 200gr | 400/420gr | | | | | |
| Feijão branco | 200gr | 410/420gr | 540/560gr | 800/845gr | | | |
| Feijão encarnado | 200gr | 410/425gr | 540/560gr | 800/860gr | | | |
| Feijão frade | 410/420gr | 540/5460gr | 800/845gr | | | | |
| Feijão preto | 410/420gr | 820/860gr | | | | | |
| Feijoada | 420gr | | | | | | |
| Frutas em calda | 410/425gr | 820/860gr | | | | | |
| Grão | 200gr | 410/420gr | 800/845gr | 500/560Gr | | | |
| Jardineira | 420gr | | | | | | |
| Lulas | 120gr | | | | | | |
| Milho | 150gr | 300gr | | | | | |
| Patê | 75/100gr | | | | | | |
| Polvo | 111gr | | | | | | |
| Salsichas | 4/8 un | 10 un | 180/380gr | | | | |
| Sardinha | 125gr | | | | | | |
| Tomate pelado | 390/780gr | | | | | | |

| Líquidos | Quantidades | | | | | | |
|---|---|---|---|---|---|---|---|
| Azeite | 50cl | 75cl | 1L | 2L | 3L | 5L | 10L |
| Óleo | 1L | 3L | 5L | | | | |
| Sumos | 20cl | 20/25cl | 33cl | 30/40gr | 1L | 1,25/1,5L | 2L |
| Vinagre | 250ml | 500ml | 75cl | 1L | | | |

| Produtos de Higiene | Quantidades | | | | | | |
|---|---|---|---|---|---|---|---|
| Algodão | 1un | | | | | | |
| Champô | 25/300ml | 400/500ml | 750ml | 1L | | | |
| Colónia | 200ml | | | | | | |
| Cotonetes | 50/200un | | | | | | |
| Creme hidratante | 200/250ml | | | | | | |
| Creme muda fraldas | 100ml | | | | | | |
| Cueca adulto | 12un | 14un | 16un | | | | |
| Cueca banho criança | 10/15un | | | | | | |
| Desodorizante | 1un | | | | | | |
| Escova de dentes | 1un | | | | | | |
| Esponja | 1un | | | | | | |
| Fralda adulto | 10/14un | 20/28un | 40/46un | 52un | | | |
| Fralda criança | 20/35un | 40/64un | 70/90un | | | | |
| Gel de banho | 250ml | 300/500ml | 750ml | | | | |
| Giletes | 1un | | | | | | |
| Loção corporal | 300ml | 400ml | | | | | |
| Óleo corporal | 250ml | 500ml | | | | | |
| Papel higiénico | 4/9un | 12/18un | | | | | |
| Papel higiénico húmido | 1un | | | | | | |
| Pasta de dentes | 1un | | | | | | |
| Pensos higiénicos | 10/20un | 24/40un | | | | | |
| Resguardos | 10un | 20un | | | | | |
| sabonete | 1un | | | | | | |
| Sabonete líquido | 1un | | | | | | |
| Soro fisiológico | 60ml | | | | | | |
| Toalhitas | 20/30un | 40un | 50/64un | 72un | | | |

| Produtos de limpeza | Quantidades | | | | | | |
|---|---|---|---|---|---|---|---|
| Amaciador roupa | 1/3L | | | | | | |
| Ambientador | 1un | | | | | | |
| Detergente louça | 500ml | 750ml | 1L | | | | |
| Detergente roupa liquido | 1un | | | | | | |
| Detergente roupa pó | 1un | | | | | | |
| Detergente W.C. | 1L | | | | | | |
| Esfregão | 1un | | | | | | |
| Guardanapos | 1un | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Lava tudo | 750ml | 1/2L | 3L | | | | |
| Limpa móveis | 1un | | | | | | |
| Limpa vidros | 1un | | | | | | |
| Lixivia | 1/2L | 4L | | | | | |
| Rolos de cozinha | 1un | | | | | | |
| Sabão roupa | 1un | | | | | | |

| Charcutaria | Quantidades | | | | | | |
|---|---|---|---|---|---|---|---|
| Chouriço | 13/250gr | | | | | | |
| Farinheira | 150/230gr | | | | | | |
| Fiambre | 100/200gr | | | | | | |
| Linguiça | 150gr | | | | | | |
| Paio | 80gr | | | | | | |
| Presunto | 70gr | | | | | | |
| Queijo | 100gr | 140gr | 200gr | 300/400gr | 500/700gr | 1Kg | |

# A4. Teste em terreno

Durante a visita à sede da Caritas Diocesana de Coimbra a 18/12/2014, fez-se uma pequena simulação de recolha na presença de pessoas que participaram na inventariação da campanha de recolha que tinha terminado na semana anterior.

Segundo foi mencionado, na divisão estava presente um total de 210 cabazes, cada um com uma certa quantidade de produtos por categoria.

A experiência feita, consistiu em fazer a leitura de alguns produtos dos cabazes e anotar o tempo despendido para tal. De acordo com a Tabela 51, mesmo com a limitação da camara dos *Smartphones* não ser especificamente para leitura de código de barras, este método mostrou ser mais rápido.

| | **Recolha apenas por introdução manual** | **Recolha apenas por introdução automática** |
|---|---|---|
| **Produtos inseridos** | 1un bolo-rei<br>1un bacalhau<br>2kg peras<br>2kg maçãs<br>2kg laranjas<br>2kg kiwis<br>20kg batatas<br>1un abobora<br>6x1L leite<br>3x350gr Salsichas<br>750ml azeite<br>355gr cogumelos | 500g cereais<br>375gr cereais<br>6x1L leite<br>200gr cevada (chá)<br>500gr mel<br>550gr queijo<br>1kg farinha<br>1L óleo alimentar<br>750ml azeite<br>355gr cogumelos<br>3x120gr atum<br>2x830gr feijão cozido<br>100g chocolate<br>3x350gr Salsichas |
| | **Tempo decorrido:** 8 minutos | **Tempo decorrido:** 7 minutos |

*Tabela 51 - Produtos e tempos registados na simulação*

Durante a simulação foram também detetados os seguintes problemas:

1. Supondo que se demora 7 minutos para introduzir os produtos equivalentes a um cabaz, isso representa 1470 minutos (24.5h) para introduzir todos os produtos
2. Existe um certo intervalo de tempo entre leituras de códigos de barras de produtos
3. Num *Smartphone* Huawei, a Webview encarregue de apresentar a vista do ecrã de recolha, não teve o comportamento esperado, permitindo fazer *scroll* em áreas onde tal não seria suposto

Para tal, pode-se apresentar as seguintes soluções:

1. Agrupar e contar os produtos semelhantes, inserindo no sistema através de introdução manual, todos os produtos de uma subcategoria em uma só ação e usar introdução automática apenas para produtos presentes com muita recorrência
2. A completa integração da biblioteca ZXing no código da aplicação pode permitir uma leitura sequenciada mais rápida, com a desvantagem da aplicação ficar desatualizada sempre que for lançado uma atualização da biblioteca
3. O uso de alguma framework destinada a criar *User Interfaces* nas Webview's pode ajudar a uniformizar estes comportamentos

# A5. Participação no concurso eVida Dev Challenge

Próximo à data de finalização da release 5, surgiu a oportunidade de aplicar o projeto desenvolvido numa plataforma agregadora de um conjunto de soluções em vista a saúde e bem-estar das pessoas. Esta oportunidade advém do concurso eVida Dev Challenge[42], com vista a demonstrar o conceito do eVida, uma plataforma desenhada e implementada no projeto TICE.Healty, um grupo de entidades sediadas no IPN.

De forma mais detalhada, este concurso pretende estimular empresas, universidades e pessoas a embarcar do desenvolvimento de soluções que permitam melhorar a qualidade de vida de pessoas. Para isso deverão seguir um dos caminhos:

- Fazer um sistema que permita responder a um problema apresentado por uma instituição de apoio social;
- Mostrar um projeto que não respondendo a um problema proposto, mostre ser inovador dentro do âmbito do concurso e de toda a plataforma.

Mesmo não respondendo a nenhum dos problemas em concreto, este projeto tem tudo que é necessário para mostrar ser uma aplicação inovadora, motivo pelo qual aproveitou-se de imediato esta oportunidade.

Isto deu origem à necessidade de criar uma release, com capacidade de providenciar suporte ao que já estava feito para a Caritas e ao mesmo tempo incorporar todos os requisitos inerentes à participação neste concurso, os quais serão descritos mais em diante.

De entre o que foi possível concluir até ao momento:

- O acesso à plataforma é feito por pedidos REST;
- Existe a hipótese de integrar a parte de servidor e a parte de cliente com a plataforma;
- Toda a autentificação é feita com a plataforma eVida, através de um sistema de autenticação OAuth;
- Tanto a aplicação servidor como cliente, ficarão disponíveis na plataforma para qualquer pessoa ou entidade experimentar.

[42] http://tice.healthy.ipn.pt/index.php/noticias/57 [Último acesso a 31 de Janeiro de 2015]

# A6. Publicação no Cidadania 2.0

Outra forma de dar a conhecer o projeto, passou pelo Cidadania 2.0[43], um projeto desenvolvido pela empresa knowman[44], com os objetivos de:

- Enfatizar a importância do diálogo entre Governo, Administração Pública, ONG's e os cidadãos em geral;
- Dar a conhecer exemplos concretos do que pode ser feito nesse sentido, alertando para os desafios, mostrando oportunidades, e partilhando resultados;
- Inspirar as organizações portuguesas a experimentar as ferramentas sociais para iniciar o diálogo com aqueles que servem, ou com aqueles de quem precisam para ir ao encontro dos seus propósitos;
- Oferecer pontes, contactos, orientações que permitam alavancar projetos idealizados ou já implementados de cidadania 2.0.

Na sua página oficial, disponibilizam uma lista de projetos que se enquadram nestes mesmos objetivos, os quais os utilizadores podem conhecer, experimentar e dar um feedback. Achou-se que seria de todo o proveito publicar este projeto, não só por se tratar de uma ferramenta para o bem público, mas também para se puder obter um feedback de outros utilizadores que não estejam tão por dentro do projeto.

Para efeitos de publicação, usou-se igualmente a release 6 deste projeto, numa extensão inteiramente pública e devidamente protegida contra alterações.

[43] http://cidadania20.com/ [Último acesso a 31 de Janeiro de 2015]
[44] http://knowman.pt/ [Último acesso a 31 de Janeiro de 2015]

# A7. Publicação de artigo para a conferência ISAmI 2015

O International Symposium on Ambient Intelligence[45] é um evento a nível internacional, que pretende dar a conhecer estudos feitos nas áreas de inteligência artificial e na interação homem-máquina. Este ano será albergado pela Universidade de Salamanca, na sua 6ª edição, a decorrer de 3 a 5 de Junho.

De acordo com a secção "Call for Papers[46]", é possível submeter artigos nas áreas:

- Applications
- Ambient Assisted Living
- Ubiquitous Computing
- Artificial Intelligence for AmI
- Distributed Computing
- Domotics (Home Automation)
- Pervasive Computing
- Context Aware Computing
- Agent & Multiagent Systems for AmI
- Mobile Computing
- Robotics
- Computational Creativity
- Sentient Computing
- e-Health
- Context Modelling
- e Learning
- Memory Assistant

Enquadrando-se este projeto nalgumas dessas áreas, optou-se por fazer um artigo científico e submeter para aprovação do júri do evento, o qual poderá ser consultado no futuro, com o nome "Mobile Crowd Sensing for Solidarity Campaigns".

[45] http://isami.usal.es/ [Último acesso a 31 de Janeiro de 2015]

[46] http://isami.usal.es/cfp [Último acesso a 31 de Janeiro de 2015]

# A8. Release 6

A necessidade de disponibilizar esta plataforma para o domínio público e para o concurso eVida, ao mesmo tempo que se disponibilizava para a Caritas, levou a que fossem feitas modificações a nível do servidor, tanto a nível de Webservices, como de página web de administração. Foram também criadas páginas de administração diferentes para cada entidade e uma página de apresentação genérica. Em paralelo com isto, criou-se também uma versão alternativa da versão mobile, sem os logotipos da Caritas Diocesana de Coimbra.

## A8.1. Requisitos

Com estas alterações, os requisitos para esta nova release podem resumir-se a:

- Manter a disponibilidade e interface de toda a plataforma direcionada à Caritas
- Criar cópia da plataforma atual, com integração na plataforma eVida
    - Suporte a Login na aplicação cliente via OAuth
    - Acesso e atualização de dados do eVida por parte da aplicação servidor, via Interface REST
    - Possibilidade de ver dados estatísticos de uma campanha em decurso, diretamente na *frontpage*
- Criar cópia da plataforma atual, para o domínio publico
    - Exclusão de funções que alterem dados para evitar abusos do sistema
    - Acesso à interface de administração unicamente em modo só de leitura
    - Ocultação de dados sensíveis que possam levar à identificação das pessoas
    - Possibilidade de ver dados estatísticos de uma campanha a decorrer, diretamente na *frontpage*

## A8.2. Desenho da Arquitetura da Release 6

Estes requisitos implicaram mudanças consideráveis na globalidade do projeto e mais uma vez, a obtenção de novo conhecimento. Contrariamente ao que se pudesse pensar, a maioria das mudanças apenas tem relevância de uma prespectiva superior e não em tão baixo nível, justificado pelo facto de a maioria do código apenas ter sido replicado e só em pequenas circunstancias, tenha havido alterações.

No seguimento deste subcapítulo, vai-se descrever as mudanças efetuadas nas diferentes áreas do projeto.

### Estrutura do servidor

Semelhante à estrutura da release 5, as alterações mais relevantes foram precisamente a replicação do conjunto de Verticles e Processors (Workers), e a adição das novas resources ao serviço REST no Verticle RestServer.

A estrutura resultante pode ser vista na Figura 73.

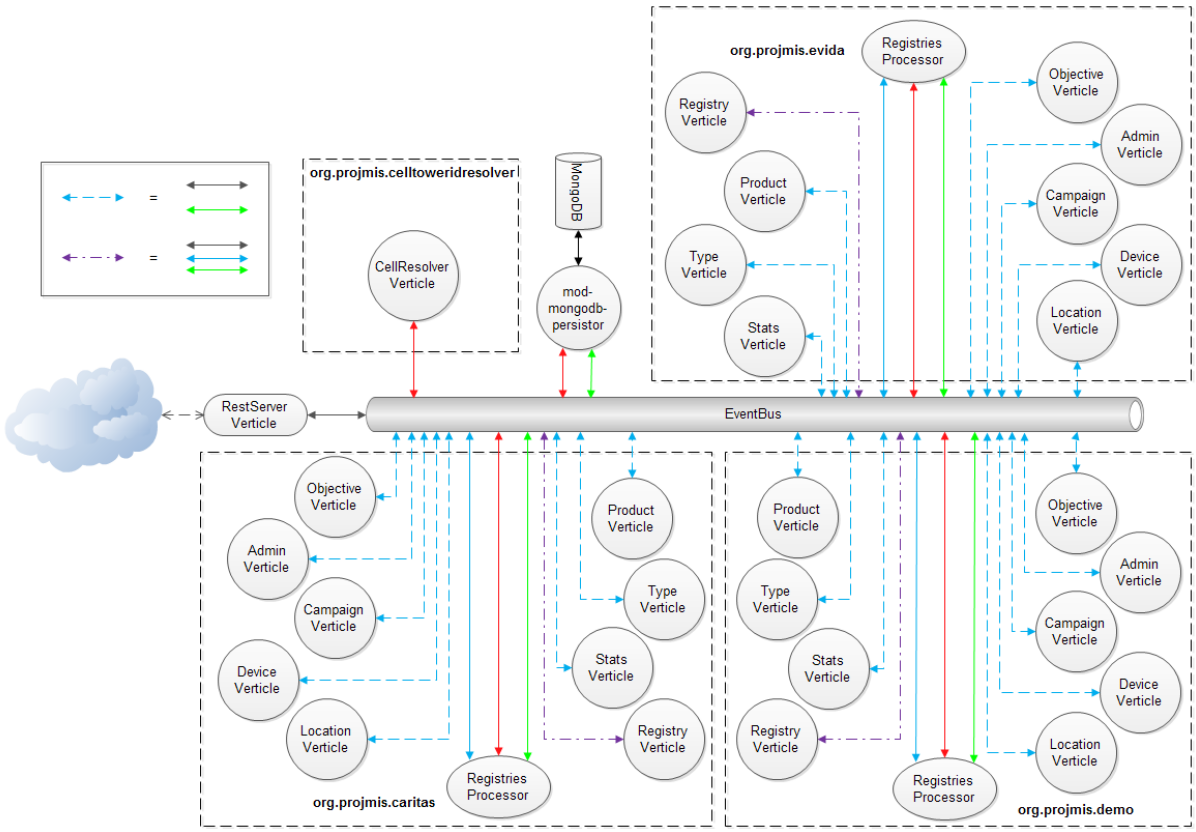



*Figura 73 - Estrutura do servidor correspondente á release 6*

### Base de dados MongoDB (Servidor)

Derivado da necessidade de manter os dados referentes à Caritas, concurso eVida e acesso público devidamente separados, a melhor opção passou por manter uma única base de dados “celltowersids” para providenciar a resolução de localizações baseadas no id das torres GSM para todo o sistema e replicar as collections da antiga base de dados “caritas” por duas novas bases de dados. Ficando assim com as bases de dados:

- celltowersids
- caritas
- evida
- rb_demo

O requisito de permitir a visualização das estatísticas na interface de acesso público, também acarretou uma pequena modificação na base de dados. Para que seja possível selecionar a campanha sobre a qual os dados serão apresentados, adicionou-se um campo

booleano de nome “viewBrief” com valor “true” na campanha. Ficando assim de acordo com o exemplo que se pode ver na Tabela 52.

```
{
  "_id" : "98b81029-72ef-4270-93cc-4fab0985d19a",
  "idCampaign" : "a702b35f-10d9-45da-8d67-75e88c3fc63f",
  "name" : "Campanha Jan 2015 (cj2015)",
  "password" : "cj2015",
  "warehouse" : "Principal",
  "dateStart" : NumberLong("1420070400000"),
  "dateEnd" : NumberLong("1422748799999"),
  "visible" : true,
  "numDevices" : 0,
  "objectives" : {
    "a" : 19000,
    "b" : 11000,
    "c" : 15000,
    "d" : 14000,
    "e" : 20000,
    "f" : 18000,
    "g" : 12000,
    "h" : 10000,
    "i" : 13000
  },
  "stats" : {
    "a" : 500,
    "b" : 1000,
    "c" : 0,
    "d" : 0,
    "e" : 0,
    "f" : 0,
    "g" : 0,
    "h" : 0,
    "i" : 0
  },
  "detailedStats" : { },
  "sumWeights" : NumberLong(0),
  "sumRegistries" : 0,
  "devices" : [],
  "viewBrief" : true
}
```

*Tabela 52 - Exemplo de campanha da qual as estatísticas devem ser apresentadas na página inicial*

### Alterações ao conf.json

Com a replicação das resources dos Webservices e das ligações de base de dados, foi benéfico ajustar o ficheiro de configuração, para facilmente ajustar os parâmetros de cada um destes serviços com facilidade no futuro. O ajuste passou por passar todas as configurações referentes ao acesso à base de dados, ligação ao Worker e credenciais de administração para um objeto distinto, conforme este se refira à entidade Caritas, acesso publico ou acesso eVida. Além deste ajuste, também se adicionou o campo para definir a "consumer-key" no acesso eVida, necessário para permitir a comunicação entre o servidor e a plataforma eVida, com recurso a OAuth.

Com isto, o ficheiro de configuração ficou com o seguinte presente na Tabela 53.

```
{
  "listeningPort" : 8080,
  "cellid-resolver" : {
    "address" : "cellid-resolver",
    "collectionOpenCellID" : "opencellid",
    "collectionSlashGPS" : "slashgps",
    "collectionGoogleMmapCache" : "mmapcache",

    "mongo_address": "mongodb-persistor-celltower",
    "mongo_host": "localhost",
    "mongo_port": 27017,
    "mongo_pool_size": 10,
    "mongo_db_name": "celltowersids"
  },
  "caritas" : {
    "mongo-persistor" : {
      "address": "mongodb-persistor",
      "host": "localhost",
      "port": 27017,
      "pool_size": 10,
      "db_name": "caritas"
    },
    "work-queue-registries" : {
      "address": "caritas.registry.orderQueue",
      "process_timeout": 120000
    },
    "admin" : {
      "username" : "admin",
      "password" : "pasS!2"
    }
  },
  "demo" : {
    "mongo-persistor" : {
      "address": "mongodb-persistor-demo",
      "host": "localhost",
      "port": 27017,
      "pool_size": 10,
      "db_name": "rb_demo"
    },
    "work-queue-registries" : {
      "address": "demo.registry.orderQueue",
      "process_timeout": 120000
    },
    "admin" : {
      "username" : "demo",
      "password" : "demo"
    }
  },
```

```
  "evida" : {
    "mongo-persistor" : {

      "address": "mongodb-persistor-evida",
      "host": "localhost",
      "port": 27017,
      "pool_size": 10,
      "db_name": "evida"
    },

    "work-queue-registries" : {
      "address": "evida.registry.orderQueue",
      "process_timeout": 120000
    },
    "admin" : {
      "username" : "demo",
      "password" : "demo"
    },
    "consumer-key": "0123456789ABCDEF"
  }
}
```

*Tabela 53 - Ficheiro de configuração adaptado para suportar todas as entidades e respetivos parâmetros*

### Uso de bootstrap para criar as páginas

De acordo com o regulamento[47] do concurso eVida Dev Challenge, um dos fatores de avaliação dos projetos é precisamente o modo como são implementados, especificamente a usabilidade e o design. Em específico para este segundo ponto, foi disponibilizado uma *styleguide*[48], juntamente com uma biblioteca de estilos[49], baseado em Bootstrap.[50]

O Bootstrap trata-se de uma framework em HTML, CSS e JavaScript, que permite acelerar o processo de construção do front-end de aplicações para web, seguindo um modelo responsivo e adaptado tanto para mobile, como para desktop.

Mesmo não sendo o desenvolvimento Web um dos pontos fundamentais deste trabalho, o uso desta framework revelou-se simples e bastante benéfico, ao ponto de num curto prazo de tempo, ter sido possível adaptar todo o código encarregue de apresentar as páginas da área de administração, ao mesmo tempo que se conferiu um aspeto muito mais profissional (Figura 74 e Figura 75) e que se adapta perfeitamente para consulta em dispositivos de ecrã pequeno (Figura 76).

O mesmo foi usado para criar uma *frontpage* de apresentação do projeto, embora para este fim específico, se tenha optado por usar um tema disponível (Figura 77).

[47] http://tice.healthy.ipn.pt/images/devchallenge/tice%20regulamento%20evida%20dev%20challenge%20v3.pdf [Último acesso a 31 de Janeiro de 2015]

[48] https://developer.evida.pt/docs/front-end-style-guides/ [Último acesso a 31 de Janeiro de 2015]

[49] https://github.com/evida/Styleguides [Último acesso a 31 de Janeiro de 2015]

[50] http://getbootstrap.com/ [Último acesso a 31 de Janeiro de 2015]


*Figura 74 - Página de login da área de administração com dados estatísticos*

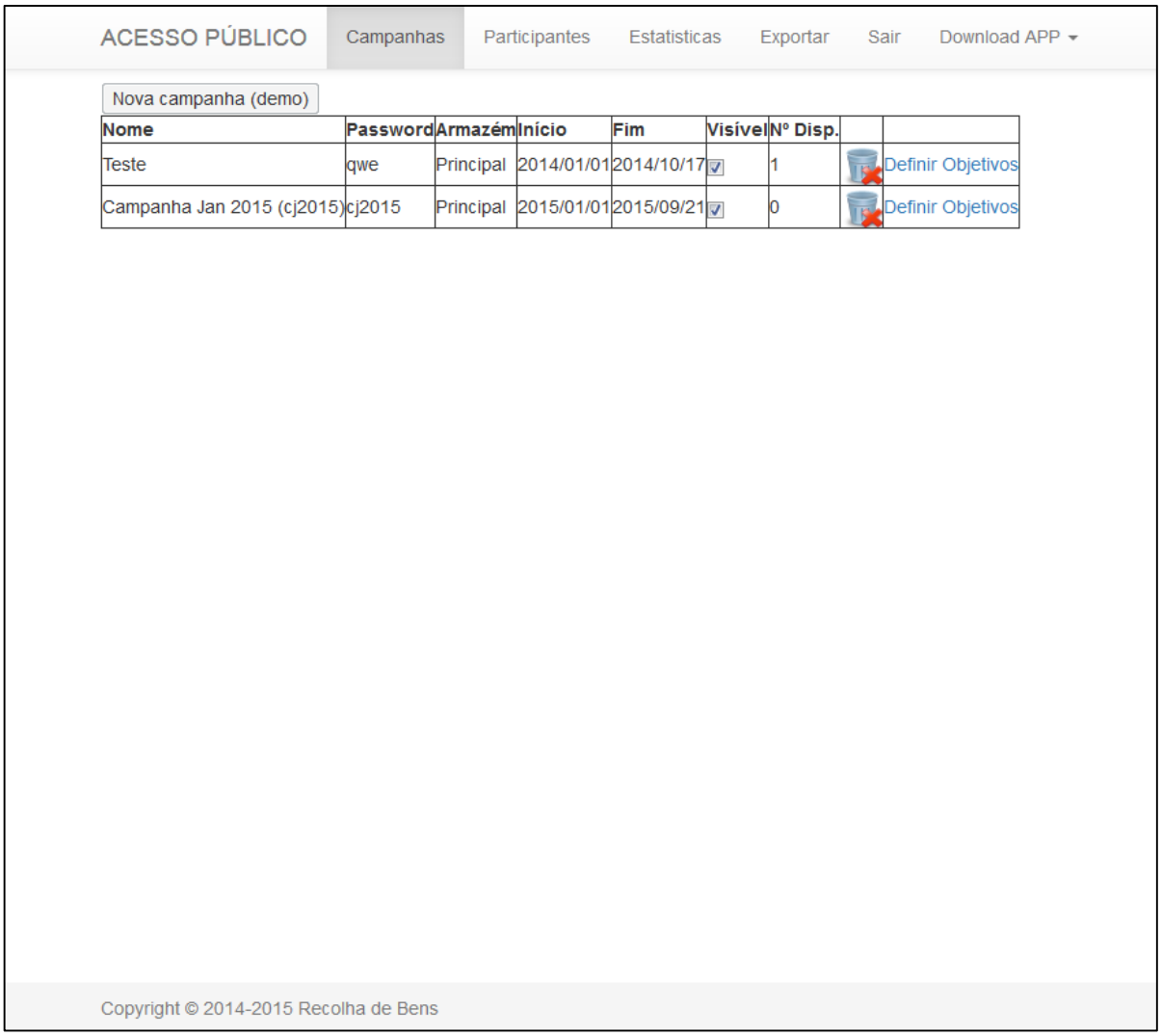



*Figura 75 - Área de visualização de campanhas disponíveis na interface administrativa*

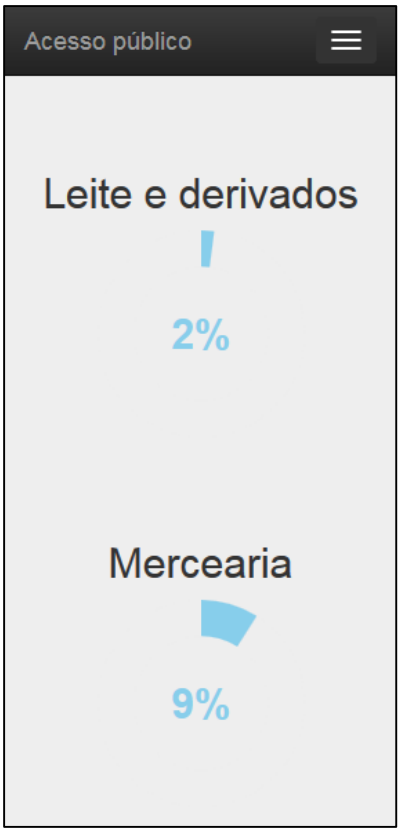



*Figura 76 - Página de login da área de administração com dados estatísticos vista num dispositivo móvel*


*Figura 77 - Fração da página de apresentação*

**Frontpage com novos links**

A introdução do Bootstrap e das diversas páginas de administração levou a que houvesse a necessidade de fornecer um novo conjunto de resources. De forma resumida, o conjunto completo de resources, pode ser na Tabela 54 e Tabela 55.

| **Método GET** | |
|---|---|
| **URL** | **Descrição** |
| / | Redireciona para /frontpage/index.html |
| /frontpage | Redireciona para /frontpage/index.html |
| /frontpage/ | Redireciona para /frontpage/index.html |
| /frontpage/index.html | Página de apresentação do projeto |
| /frontpage/css/(file) | Devolve o conteúdo do ficheiro css css/(file) |
| /frontpage/fonts/(file) | Devolve o conteúdo do ficheiro font fonts/(file) |
| /frontpage/images/(file) | Devolve o conteúdo da imagem images/(file) |
| /frontpage/js/(file) | Devolve o conteúdo do ficheiro JavaScript js/(file) |
| /[entity] | Redireciona para /[entity]/index.html |
| /[entity]/ | Redireciona para /[entity]/index.html |
| /[entity]/index.html | Página inicial de administração |
| */favicon.ico | Devolve o conteúdo do ficheiro images/favicon.ico |
| /[entity]/login.html | Página de login |
| /[entity]/logout.html | Invalida o cookie de sessão e redireciona para /[entity]/login.html |
| /[entity]/images/(image) | Devolve o conteúdo da imagem images/(image) |
| /[entity]/campaigns.html | Página de gestão de campanhas |
| /[entity]/devices.html | Página de gestão de dispositivos |
| /[entity]/conflicts.html[1] | Página de gestão de conflitos |
| /[entity]/stats.html | Página de seleção de estatísticas a visualizar |
| /[entity]/objectives.html | Página de seleção de objetivos a visualizar |
| /[entity]/export.html | Páginas de exportação de dados |

*Tabela 54 - Métodos GET suportados para acesso à interface Web*

| **Método POST** | |
|---|---|
| **URL** | **Descrição** |
| /[entity]/login.html | Envio de dados de autenticação |
| /[entity]/campaigns.html[1] | Alterar visibilidade de campanhas |
| /[entity]/newcampaign.html[1] | Criação de campanha |
| /[entity]/delcampaign.html[1] | Eliminação de campanha |
| /[entity]/devices.html[1] | Bloqueio de dispositivos |
| /[entity]/conflicts.html[1] | Decisão para resolução de conflitos |
| /[entity]/objectives.html[1] | Definição de valor objetivo |

*Tabela 55 - Métodos POST suportados para acesso às funções da interface Web*

[1] – Não disponível em modo público

[entity] – “caritas-admin”, “demo” ou “evida”

### Interface REST

Tal como aconteceu nos links para as páginas de administração, a replicação de Webservices levou à adição de mais resources disponíveis para acesso. Estas resumem-se ao conteúdo da Tabela 56.

| | URL | Método | Descrição |
|---|---|---|---|
| **API Caritas** | http://server/ping | POST | Informa se o servidor está pronto a processar pedidos |
| | http://server/caritas/startup | GET | Obtém todos os dados referentes ao estado do dispositivo, campanhas, objetivos, estatísticas e tipos, e também regista o dispositivo no sistema se necessário |
| | http://server/caritas/campaign | POST | Regista o dispositivo numa nova campanha |
| | http://server/caritas/product | GET | Obtém um conjunto de produtos |
| | http://server/caritas/registry | POST | Adiciona um conjunto de registos para processamento |
| | http://server/caritas/stats | GET | Obtém estatísticas atuais |
| **API demonstração pública** | http://server/pingdemo | POST | Informa se o servidor está pronto a processar pedidos |
| | http://server/demorb/startup | GET | Obtém todos os dados referentes ao estado do dispositivo, campanhas, objetivos, estatísticas e tipos, e também regista o dispositivo no sistema se necessário |
| | http://server/demorb/campaign | POST | Regista o dispositivo numa nova campanha |
| | http://server/demorb/product | GET | Obtém um conjunto de produtos |
| | http://server/demorb/registry | POST | Adiciona um conjunto de registos para processamento |
| | http://server/demorb/stats | GET | Obtém estatísticas atuais |
| **API concurso eVida** | http://server/pingevida | POST | Informa se o servidor está pronto a processar pedidos |
| | http://server/evida/startup | GET | Obtém todos os dados referentes ao estado do dispositivo, campanhas, objetivos, estatísticas e tipos, e também regista o dispositivo no sistema se necessário |
| | http://server/evida/campaign | POST | Regista o dispositivo numa nova campanha |
| | http://server/evida/product | GET | Obtém um conjunto de produtos |
| | http://server/evida/registry | POST | Adiciona um conjunto de registos para processamento |
| | http://server/evida/stats | GET | Obtém estatísticas atuais |

*Tabela 56 - Métodos suportados pela interface REST*

# A9. Proposta de projeto

DEPARTAMENTO DE
ENGENHARIA INFORMÁTICA
E DE SISTEMAS
INSTITUTO SUPERIOR DE
ENGENHARIA DE COIMBRA

# PROPOSTA DE ESTÁGIO

## MESTRADO em INFORMÁTICA E SISTEMAS

## especialização em Desenvolvimento de Software

**Ano Lectivo de 2013/2014**

**TEMA**

## Desenvolvimento de um sistema de *CrowdSensing* para Campanhas de Solidariedade

**SUMÁRIO**

*Crowdsensing* é uma área de investigação e desenvolvimento em Computação Ubíqua que permite entender os padrões de participação dos utilizadores em tempo real através de uma rede distribuída de sensores. Atualmente existem diversas campanhas de solidariedade para a doação de bens alimentares e outros a nível nacional, algumas das quais só podem ser avaliadas quanto ao seu sucesso depois de um processo moroso de agregação da informação dos diversos pontos de recolha. Este trabalho pretende desenvolver um sistema baseado em *Web Services* e aplicações móveis que permita recolher, catalogar e disponibilizar informação em tempo real sobre a evolução e adesão de uma dada campanha de solidariedade na Cidade de Coimbra. Além da vertente informativa, esta informação permitirá orientar os utilizadores para que tipo de produtos estão sendo doados em menor quantidade permitindo assim orientá-los para a doação conjunta de uma maior variedade e mais equilibrada de bens.

**Palavras-chave:** Computação Ubíqua, *Crowdsensing*, Tempo Real

# 1. Âmbito

*Crowdsensing*, também conhecido como *community sensing* (1), *participatory sensing* (2) ou *opportunistic sensing* (3), varia desde a participação ativa dos utilizadores para contribuir com dados (e.g. ao tirar uma foto, relatar o fecho de uma estrada, reportar um acidente), até uma recolha mais autónoma através de sensores e com um mínimo de envolvimento (e.g. recolha contínua de dados de localização sem uma ação explícita do utilizador). Esta recolha é classificada a partir do meio pelo qual os utilizadores participam com dados: a partir de dispositivos móveis (*Mobile Crowdsensing*) ou da Web (*Web Crowdsensing*).

O funcionamento típico de aplicações móveis de *Crowdsensing* é baseado na recolha de dados simples de sensores nos dispositivos e processam estes dados localmente a fim de produzir informação útil a enviar a um serviço/servidor para agregação e análise.

*Web crowdsensing* é a recolha de dados de voluntários onde a aquisição de dados proprietários torna-se extremamente cara (e.g. Pontos-de-Interesse introduzidos no Foursquare1[1], Códigos da barra e descrições de produtos submetidos numa base de dados universal[2]).

Devido ao contexto social e económico a nível nacional, diversas campanhas de solidariedade têm surgido nos últimos anos de âmbito nacional para a recolha de bens alimentares e outros (livros, produtos de higiene, material escolar, etc.). Estas campanhas podem ser realizadas por um curto período de tempo e promovida em meios de comunicação social com a associação de grandes superfícies (e.g. Banco Alimentar contra a Fome[3], Cruz Vermelha[4], Cáritas[5]) ou podem ter um caráter mais permanente na recolha de bens através de sites Web e pontos de recolha fixos espalhados pelo país (e.g. Reutilizar[6]).

Tendo sido aplicado com sucesso em projetos de mobilidade e planeamento urbano (4, 5, 6, 7), este trabalho pretende aplicar o conceito de *Mobile Crowdsensing* em tempo real às campanhas de solidariedade, um novo campo para uma prova de conceito. Além da sua vertente de Desenvolvimento de Software, este projeto irá contribuir para a inventariação de bens, análise dos dados recolhidos e inovar pelo objectivo social envolvido.

# 2. Objectivos

O presente projeto pretende atingir os seguintes objectivos genéricos:

[1] http://www.foursquare.com

[2] http://upcdatabase.org/

[3] http://www.bancoalimentar.pt/

[4] http://www.cruzvermelha.pt/ultimas-noticias/1809-continente-e-cruz-vermelha-promovem-recolha-de-alimentos-contra-a-fome.html

[5] http://www.jb.pt/2013/10/anadia-alunos-ajudam-caritas-a-recolher-donativos-atraves-de-campanha-partilhar-e-urgente/

[6] http://www.reutilizar.org/REUTILIZAR.ORG/Bancos.html

- Utilizar, testar e (eventualmente) melhorar algoritmos de reconhecimento de códigos de barra em dispositivos móveis;
- Desenvolver um serviço para agregação de informação para inventariação em tempo real de bens recolhidos;
- Visualizar a informação recolhida em tempo real;
- Analisar os dados recolhidos de forma a apresentar a participação temporal/espacial da campanha de recolha.

Estes objectivos intermédios vão permitir atingir o objectivo principal deste projeto, o desenvolvimento de uma aplicação móvel de recolha de dados sobre bens doados e um serviço/servidor agregador e disponível para consulta de informação através de uma API.

# 3. Programa de trabalhos

O Projeto consistirá nas seguintes atividades e respectivas tarefas:

- T1 – Estudo e caracterização das tecnologias envolvidas
- T2 – Análise de requisitos
- T3 – Desenvolvimento
- T4 – Testes (Arquitetura/ Funcionais/ Usabilidade)
- T5 – Refinamento e correções
- T6 – Desenvolvimento da interface de visualização
- T7 – Teste em campo e Recolha de dados
- T8 – Análise espacial e temporal dos dados
- T9 – Documentação (manuais, artigo e dissertação)

# 4. Calendarização Das Tarefas

As Tarefas acima descritas, incluindo os testes de validação de cada módulo, serão executadas de acordo com a seguinte calendarização:

O plano de escalonamento dos trabalhos é apresentado em seguida:

| | **Meses** | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Tarefas** | | N | | N+1 | | N+2 | | N+3 | | N+4 | | N+5 | | N+6 | | N+7 | | N+8 | |
| T1 | | | | | | | | | | | | | | | | | | | |
| T2 | | | | | | | | | | | | | | | | | | | |
| T3 | | | | | | | | | | | | | | | | | | | |
| T4 | | | | | | | | | | | | | | | | | | | |
| T5 | | | | | | | | | | | | | | | | | | | |
| T6 | | | | | | | | | | | | | | | | | | | |
| T7 | | | | | | | | | | | | | | | | | | | |
| T8 | | | | | | | | | | | | | | | | | | | |
| T9 | | | | | | | | | | | | | | | | | | | |
| **Metas** | INI | | | M1 | M2 | | | | | | | M3 | | | | M4 | | | M5 |

INI Início dos trabalhos

M1 (INI + 6 Semanas) Tarefa T1 terminada

M2 (INI + 8 Semanas) Tarefa T2 terminada

M3 (INI + 22 Semanas) Tarefa T3 terminada

M4 (INI + 28 Semanas) Tarefa T6 terminada

M5 (INI + 36 Semanas) Tarefa T9 terminada

# 5. Resultados

Os resultados do estágio serão consubstanciados num conjunto de documentos a elaborar pelo estagiário de acordo com o seguinte plano:

**M1:**

Relatório técnico com a descrição da tecnologias estudadas

**M2**:

Relatório técnico com a análise de requisitos e especificação da aplicação e serviço a desenvolver.

**M3:**

Relatório técnico com as etapas de desenvolvimento e metodologia seguida.

**M4:**

Documento a descrever os testes efectuados

**M5**:

Relatório final de estágio incluindo o manual de instalação e utilização da aplicação, assim como a documentação do funcionamento do serviço desenvolvido.

# 6. Local de Trabalho

DEIS

# 7. Metodologia

Organização de um Dossier de Projeto e reuniões semanais.

# 8. Orientação

Ana Cristina Oliveira Alves (aalves@isec.pt)

Professora Adjunta

# 9. Caracterização

- Data de início: Fevereiro de 2014
- Data de fim: Outubro de 2014

# 10. Referências

(1) Rheingold, H. 2002. Smart Mobs: The Next Social Revolution. New York: Basic Books.

(2) Mobile crowdsensing: current state and future challenges.Raghu K. Ganti, Fan Ye, Hui Lei. IEEE Communications Magazine 01/2011.

(3) J. Burke et al., “Participatory sensing,” Workshop on World- Sensor-Web, co-located with ACM SenSys, 2006. [Online]. Available: http://www.sensorplanet.org/wsw2006

(4) A holistic framework for the study of urban traces and the profiling of urban processes and dynamics, Vaccari, A., Liang, L., Biderman, A., Ratti, C., Pereira, F., Oliveirinha, J., Gerber, A., 12th International IEEE Conference on Intelligent Transportation Systems, 2009.

(5) Fabien Girardin, Andrea Vaccari, Alexandre Gerber, Assaf Biderman, Carlo Ratti, "Quantifying urban attractiveness from the distribution and density of digital footprints", International Journal of Spatial Data Infrastructures Research, 2009.

(6) Collective Mobility Patterns from Smart Card Records: A Case Study in Shenzhen", 12th International IEEE Conference on Intelligent Transportation Systems, October 3-7, 2009, St. Louis, MO, USA.

(7) Rodrigues, F. and Alves, A.O. and Polisciuc, E. and Jiang, S. and Ferreira, J. and Pereira, F.C., "Estimating disaggregated employment size from Points-of-Interest and census data: From mining the web to model implementation and visualization", International Journal on Advanced Intelligent Systems, vol. 7, 2013

# A10. Propostas de projeto de licenciatura (Proposta 1)



**Departamento de Engenharia**
Informática e de Sistemas

# Proposta de Projecto / Estágio

**Ano Lectivo de 2014/2015**

## Desenvolvimento de Ferramenta de *Backoffice* para Gestão de Bens Doados

### SUMÁRIO

Em cada ano a Caritas realiza várias campanhas de recolha de bens de diferentes categorias, distribuindo posteriormente esses bens a pessoas carenciadas. A catalogação, gestão de stocks e registos de entradas e saídas de armazém são as maiores dificuldades sentidas, agravadas pelo facto das recolhas serem feitas em diferentes locais da Diocese de Coimbra.
Um projeto de desenvolvimento de um aplicativo Android e serviço web para armazenamento de dados recolhidos para catalogação de bens[1], por parte de um aluno de mestrado do ISEC, veio ao encontro das dificuldades sentidas facilitando a parte inicial do processo. Todavia, para que seja melhorada a eficiência em termos de gestão de produtos (produtos alimentares, roupas, brinquedos, mobiliário, ajudas técnicas, etc.) é necessário que os dados recolhidos através do aplicativo Android sejam possíveis de gerir em backoffice, possibilitando eventual integração com outros sistemas de informação da Caritas.
Salienta-se que o interesse do aplicativo Android e da solução backoffice não se esgotam na Caritas de Coimbra, sendo muito úteis também para as restantes 20 caritas diocesanas e eventualmente para outras ONGs.
Este é um projecto que claramente se enquadra no ramo Desenvolvimento de Aplicações da Licenciatura em Engenharia Informática.

[1] Serviço e aplicação disponíveis em http://kenobi.dei.uc.pt:8080

# 1. Âmbito

O desenvolvimento de uma Ferramenta de Backoffice para gestão e armazenamento de dados consiste num interface web que permita a diferentes entidades (p.ex: diferentes Caritas Diocesanas):

- Criar campanhas de recolhas de diferentes tipologias por um determinado período;
- Associar dispositivos móveis para catalogação e registo dos donativos;
- Autenticar e gerir os dispositivos móveis autorizados a participar nas campanhas;
- Gerir os utilizadores por campanha desses dispositivos móveis;
- Gestão de utilizadores da plataforma backoffice;
- Gestão de bens por armazém;
- Possibilidade de transferir bens entre armazéns;
- Registo de saída dos bens;
- Mapas de stocks e de bens entregues;
- Gestão de zonas de campanha (associação de pontos de recolha a armazéns);
- Exportação de dados para ficheiros estruturados (e.g., Comma Separated Values (csv), MS Excel);

A integração dos dados recolhidos pela aplicação móvel e armazenados actualmente numa base de dados não-relacional (no SQL) optimizada para um número elevado de acessos concorrentes em tempo real, deverão ser integrados num novo modelo conceptual optimizado para consultas que envolvam o cruzamento de várias tabelas (SQL).

# 2. Objectivos

O presente projecto/estágio pretende atingir os seguintes objectivos genéricos:

- Criar plataforma de suporte a campanhas de recolha de donativos que complemente uma aplicação Android e serviço web já existentes para este efeito;
- Colaborar na resolução dos problemas sentidos por organizações do 3º Setor, contribuindo uma maior eficiência de gestão e de resposta às pessoas carenciadas;

- Facilitar e agilizar o tratamento dos dados relativos aos resultados das campanhas, permitindo também uma maior transparência e celeridade na divulgação junto de quem colabora (voluntários) e de quem contribui;
- Reduzir custos com o processo de catalogação e de gestão de donativos que atualmente ainda é feito na maioria das vezes em suporte papel.

# 3. Programa de trabalhos

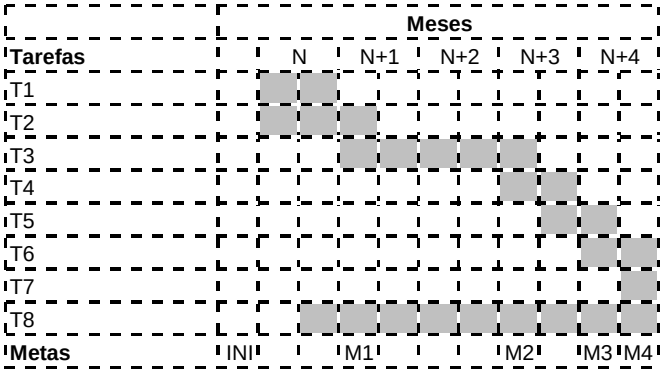
O projecto/estágio consistirá nas seguintes actividades e respectivas tarefas:

- T1 – Identificação de requisitos funcionais – Identificação e caracterização dos requisitos funcionais para desenvolvimentos do interface, bem como dos workflows associados a cada uma das funcionalidades e das permissões dos utilizadores.
- T2 – Identificação de requisitos de programação – identificação de tecnologias web e linguagens de programação adequadas aos objetivos, de eventuais tecnologias que possam ser integradas e dos requisitos de compatibilização com os SI existentes nas organizações destinatárias.
- T3 – Desenvolvimento aplicacional – programação e design gráfico da solução.
- T4 – Testes – Testes de funcionalidade e usabilidade da solução com a Organização destinatária.
- T5 – Correção de falhas – Correção de eventuais falhas e aprimoramento de aspectos de usabilidade;
- T6 – Entrada em produção: Fase piloto– Instalação/implementação da solução para utilização por uma número restrito de utilizadores mas em contexto de produção.
- T7 – Avaliação – Avaliação contínua de acordo com as quick wins estabelecidas e avaliação final do projeto.
- T8 – Escrita do relatório

NOTA: Algumas destas tarefas podem ser agrupadas (T1=T1+T2), outras podem ser desenvolvidas em simultâneo.

# 4. Calendarização das Tarefas

O plano de escalonamento dos trabalhos é apresentado em seguida (a adaptar em função do projecto/estágio proposto):




| | | |
|---|---|---|
| INI | | Início dos trabalhos |
| M1 | (INI + 6 Semanas) | Tarefas T1 e T2 terminadas |
| M2 | (INI + 14 Semanas) | Tarefa T3 terminada |
| M3 | (INI + 18 Semanas) | Tarefa T4 e T5 terminada |
| M4 | (INI + 20 Semanas) | Tarefa Ts6, T7 e T8 terminada |

# 5. Local e Horário de Trabalho

Na Sede da Caritas Diocesana de Coimbra, sito na Rua D. Francisco de Almeida, nº 14, 3030-382 COIMBRA, em horário flexível, nos dias úteis entre as 9h e as 17:30h.O horário de trabalho pressupõe um total de 720 horas dividido em 3,5 meses ou 18 semanas de efectivas de 40 horas.

# 6. Metodologia

Elaboração de dossier de projeto incluindo: requisitos da aplicação; calendarização das diferentes fases do projeto; memorandos das reuniões de projeto; outra documentação produzida no âmbito do projeto.

Realização de reuniões periódicas para definição de tarefas e para avaliação do cumprimento do plano de trabalho acordado;

Realização de reuniões periódicas com orientador de estágio para avaliação regular do desenvolvimento do projeto;

Comunicação entre estagiários e responsáveis da Organização pelo acompanhamento do projeto sempre com conhecimento ao orientador do ISEC e presidente da Caritas de Coimbra.

# 7. Orientação

DEIS-ISEC:

Ana Cristina da Costa Oliveira Alves (aalves@isec.pt)

Professora Adjunta

Co-orientação do Mestre David João Almeida dos Santos e Silva (a21170222@alunos.isec.pt) ex-aluno do MIS e responsável pelo desenvolvimento da primeira versão da aplicação de recolha de bens

Caritas Diocesana:

Pedro Balhau (pedrobalhau@caritascoimbra.pt) e Carlos Ribeiro (carlosribeiro@caritascoimbra.pt)

Categoria: Responsável do Depto de RH e Responsável pelo Gab. De Informática

# A11. Propostas de projeto de licenciatura (Proposta 2)



**Departamento de Engenharia**
Informática e de Sistemas

# Proposta de Projecto

**Ano Lectivo de 2014/2015**

# Extensão de uma Aplicação Móvel, Integração e Análise de Dados na Recolha de Donativos

## SUMÁRIO

Em cada ano, a Caritas realiza várias campanhas de recolha de donativos de diferentes espécies, distribuindo posteriormente esses bens a pessoas carenciadas. Além de bens doados a Caritas também dinamiza a campanha denominada Peditório Nacional[1], onde num âmbito alargado, voluntários associados a todas as caritas diocesanas no país recolhem donativos em espécie directamente dos cidadãos que queiram contribuir.
Um projeto de desenvolvimento de um aplicativo Android e serviço web de armazenamento de dados recolhidos para a catalogação de bens não-monetários[2], por parte de um aluno de mestrado do ISEC, veio ao encontro das dificuldades sentidas no que toca a recolha de produtos facilitando a parte inicial do processo. Todavia, para que seja possível incluir campanhas de contribuições monetárias, é necessário estender a aplicação de forma a ser possível que cada voluntário, previamente registado numa campanha, possa introduzir ao fim do seu período recolha a quantia por si recolhida. O posterior tratamento e divulgação destes dados aos responsáveis da Caritas é um factor importante, uma vez que a rapidez de resposta da efectividade de uma campanha deste tipo é fundamental para responder aos meios de comunicação social, que até durante o dia do Peditório fazem estimativas sobre o total recolhido até ao momento.
Salienta-se que o interesse do aplicativo Android não se esgota na Caritas de Coimbra, sendo muito úteis também para as restantes 20 caritas diocesanas e eventualmente para outras ONGs.

[1] http://www.caritas.pt/site/coimbra/index.php?option=com_content&view=article&id=4152:peditorio-caritas-angaria-cerca-de-30000-em-coimbra&catid=534:informacao&Itemid=1

[2] Serviço e aplicação disponíveis em http://kenobi.dei.uc.pt:8080 para download e utilização.

Este é um projecto que claramente se enquadra no ramo Desenvolvimento de Aplicações da Licenciatura em Engenharia Informática.

# 1. Âmbito

Existe actualmente um aplicativo Android para a recolha de bens (alimentares, material escolar, produtos de higiene, etc.) que comunica com um servidor Web para o armazenamento centralizado dos dados recolhidos. A extensão pretendida irá ser integrada neste aplicativo e deverá permitir as seguintes funcionalidades:

- Autenticar de forma segura o responsável dos voluntários/utilizador do aplicativo;
- Autenticar e gerir os dispositivos móveis autorizados a participar nas campanhas;
- Registar além da quantia introduzida ao fim de um turno de participação da campanha pelo responsável dos voluntários, o instante e local da recolha;
- Disponibilizar um formulário web que possa ser alternativo à introdução de dados através dos dispositivos móveis;
- Comunicar os dados a um servidor centralizado que integra toda a informação recebida;

Além da extensão do Aplicativo Android, pretende-se que o aluno explore os dados recolhidos de forma alargada a bens monetários e não-monetários para que seja possível:

- *Dashboard* com indicadores dos volumes recolhidos nas campanhas por tipo de bem, zonas de recolha;
- Visualizar os dados recolhidos de forma a avaliar a distribuição geográfica e evolução temporal ao longo do dia da Campanha;
- Agrupar (*clustering*) e analisar os dados recolhidos para a criação de perfis de contribuição por região;
- Exportação de dados para ficheiros estruturados (e.g., Comma Separated Value (csv), MS Excel);

# 2. Objectivos

O presente projecto/estágio pretende atingir os seguintes objectivos genéricos:

- Permitir o registo de donativos (bens monetários) em campanhas de solidariedade através da extensão de um Aplicativo Android para a recolha de bens.

- Facilitar e agilizar o tratamento dos dados relativos aos resultados das campanhas, permitindo também uma maior transparência e celeridade na divulgação junto de quem colabora (voluntários) e de quem contribui;
- Reduzir custos com o processo de registo deste tipo de donativos que atualmente ainda é feito na maioria das vezes em suporte papel;
- Integrar os dados recolhidos na base de dados actualmente existente;
- Aplicar técnicas de visualização e análise de dados dos bens (monetários e não-monetários) recolhidos;
-

# 3. Programa de trabalhos

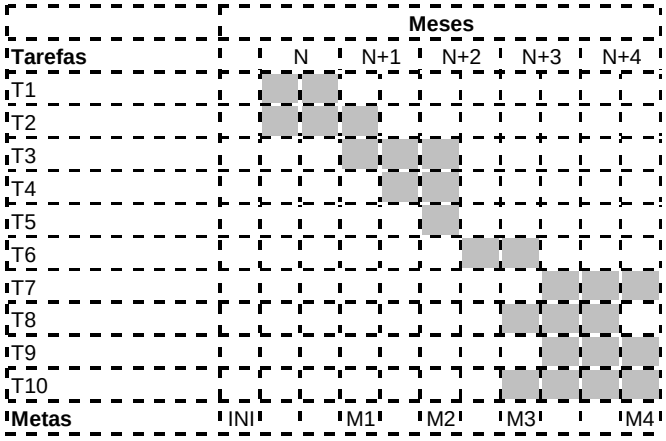
O projecto/estágio consistirá nas seguintes actividades e respectivas tarefas:

- T1 – Estudo da aplicação actualmente desenvolvida – identificação dos aspectos principais na arquitectura do aplicativo e comunicação;
- T2 - Identificação de requisitos funcionais e de programação -Identificação e caracterização dos requisitos funcionais e tecnologias para desenvolvimento e integração da funcionalidade no aplicativo, bem como dos workflows associados e das permissões dos utilizadores.
- T3 – Desenvolvimento aplicacional – programação e design gráfico da solução.
- T4 – Testes – Testes de funcionalidade e usabilidade da solução com a Organização destinatária.
- T5 – Correção de falhas – Correção de eventuais falhas e aprimoramento de aspectos de usabilidade;
- T6 – Entrada em produção: Fase piloto– Instalação/implementação da solução para utilização por uma número restrito de utilizadores mas em contexto de produção.
- T7 – Disseminação – Alargamento da solução a outras entidades; Divulgação da aplicação em fóruns, seminários técnicos e através de outros canais que fomentem a divulgação da inovação; Pesquisa de possíveis interessados na integração desta solução com outras tecnologias de mercado;
- T8 – Estudo do modelo dos dados recolhidos – Representação do modelo conceptual e físico da base de dados no servidor; Migração dos dados armazenados numa BD não-relacional otimizada para a recolha em tempo real, para uma BD relacional otimizada para consultas e atualizações.
- T9 – Visualização e análise dos dados – Aplicação de técnicas existentes para a importação automática, visualização e análise exploratória dos dados
- T10 – Escrita do relatório

NOTA: Algumas destas tarefas podem ser agrupadas (T1=T1+T2), outras podem ser desenvolvidas em simultâneo.

# 4. Calendarização das Tarefas

O plano de escalonamento dos trabalhos é apresentado em seguida (a adaptar em função do projecto/estágio proposto):




| | | |
|---|---|---|
| INI | | Início dos trabalhos |
| M1 | (INI + 6 Semanas) | Tarefas T1 e T2 terminadas |
| M2 | (INI + 14 Semanas) | Tarefas T3, T4 e T5 terminadas |
| M3 | (INI + 18 Semanas) | Tarefa T6 terminada |
| M4 | (INI + 20 Semanas) | Tarefa T7, T8, T9 e T10 terminadas |

# 5. Local e Horário de Trabalho

No DEIS- ISEC e na Sede da Caritas Diocesana de Coimbra, sito na Rua D. Francisco de Almeida, nº 14, 3030-382 COIMBRA, em horário flexível, nos dias úteis entre as 9h e as 17:30h.O horário de trabalho pressupõe um total de 720 horas dividido em 3,5 meses ou 18 semanas de efectivas de 40 horas.

# 6. Metodologia

Elaboração de dossier de projeto incluindo: requisitos da aplicação; calendarização das diferentes fases do projeto; memorandos das reuniões de projeto; outra documentação produzida no âmbito do projeto.

Realização de reuniões periódicas para definição de tarefas e para avaliação do cumprimento do plano de trabalho acordado;

Realização de reuniões periódicas com orientador de estágio e orientador do ISEC para

avaliação regular do desenvolvimento do projeto;

Comunicação entre estagiários e responsáveis da Organização pelo acompanhamento do projeto sempre com conhecimento ao orientador do ISEC e presidente da Caritas de Coimbra.

# 7. Orientação

DEIS-ISEC:

Ana Cristina da Costa Oliveira Alves (aalves@isec.pt)
Professora Adjunta

Co-orientação do Mestre David João Almeida dos Santos e Silva (a21170222@alunos.isec.pt) ex-aluno do MIS e responsável pelo desenvolvimento da primeira versão da aplicação de recolha de bens

Caritas Diocesana:

Pedro Balhau (pedrobalhau@caritascoimbra.pt) e Carlos Ribeiro (carlosribeiro@caritascoimbra.pt)
Categoria: Responsável do Depto de RH e Responsável pelo Gab. De Informática