Relatório de Projeto

Mestrado em Engenharia Informática – Computação Móvel

Urban Management and Inspection App:

(Sistema móvel de suporte a atividades de gestão de licenciamento urbano)

Rui Inácio Ferreira

Leiria, *Novembro* de *2013*

Relatório de Projeto

Mestrado em Engenharia Informática – Computação Móvel

Urban Management and Inspection App:

(Sistema móvel de suporte a atividades de gestão de licenciamento urbano)

Rui Inácio Ferreira

Dissertação de Mestrado realizada sob a orientação do Doutor Rui Rijo, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, *Novembro* de *2013*

À Minha Família

Esta página foi intencionalmente deixada em branco

Agradecimentos

Ao Professor Rui Rijo, orientador desta tese de mestrado, pela orientação, apoio disponibilizado e contributo na área de gestão de projetos informáticos.

Aos orientadores da empresa Link, Helena Oliveira e Valter Morais, pelo contributo na área da gestão de urbanismo e pela disponibilidade e acompanhamento prestado ao longo da tese.

Aos colaboradores da empresa Link, pelo apoio e ajuda na resolução de problemas relacionados com o projeto.

À coordenação do curso de Mestrado, em especial ao Professor Nuno Costa e Carlos Grilo, pelo acompanhamento através das contas de email e partilha de informação.

À Escola Superior de Tecnologia e Gestão de Leiria, pelas instalações e a todos os professores do curso de Mestrado em Computação Móvel, pela formação e conhecimentos transmitidos durante este Mestrado.

À empresa Link, por ter cedido as instalações e fornecido o material para desenvolvimento desta tese.

Por fim, e não menos importante, aos meus pais, que sempre me deram apoio incondicional e força para atingir os meus objetivos ao longo da vida.

A todos, muito obrigado.

Esta página foi intencionalmente deixada em branco

Resumo

A revolução da tecnologia móvel na sociedade estimulou o uso de novos métodos de tratamento de informação que são independentes do tempo e do espaço. Para as empresas, estes métodos traduzem-se em aliciantes oportunidades de negócio.
Este trabalho apresenta um sistema móvel para dispositivos *Android*, com suporte multiplataforma. O sistema destina-se a atividades de fiscalização de obras em regime urbanístico e tem como objetivo otimizar e automatizar os processos de negócio, de fiscalização de obras no terreno, através da integração com a solução eUrban da empresa Link. Inicialmente, neste documento, o tema do projeto é enquadrado com o objetivo da aplicação, através da apresentação de vários conceitos relacionados com os processos de negócio e o uso de tecnologia móvel no campo de trabalho.
O trabalho desenvolvido apresenta as tecnologias necessárias à implementação do sistema, segundo o tipo de aplicação a desenvolver e os padrões de desenho para soluções generalizadas de reutilização de código.
Para a construção dos protótipos da aplicação investigaram-se as normas utilizadas para desenvolvimento de interfaces gráficas de utilizador. É avaliado um estudo de caso, onde o contributo de uma aplicação móvel na fiscalização de obras é muito satisfatório.
Na realização da aplicação adotou-se o seguinte processo de desenvolvimento de *software*: o levantamento de requisitos e funcionalidades; a descrição dos casos de uso; o diagrama de classes do sistema; os protótipos das funcionalidades para os dispositivos; os formatos de comunicação de dados entre o sistema móvel e servidor; arquitetura do sistema com as tecnologias de desenvolvimento multiplataforma *Xamarin* e *MvvmCross*.
A Interface gráfica para *Android* está implementada através da linguagem *C#* do *Visual Studio 2010* e da tecnologia *Xamarin*. O sistema usa o padrão de arquitetura *Model-View-ViewModel (M-V-VM)* da tecnologia *MvvmCross* para partilha/separação do código, o repositório de dados *Sqlite* para guardar o modelo de objetos e sincroniza com o servidor através de *webservices Json*. Embora existam algumas dificuldades na implementação da aplicação multiplataforma, verifica-se que a metodologia de desenvolvimento escolhida foi uma boa opção. A nível de comportamento gráfico, desempenho face á metodologia escolhida, e permite desenvolver de forma rápida e com custos reduzidos, versões do sistema para outras plataformas. Como trabalho futuro, podem ser desenvolvidas as funcionalidades em falta e implementar a aplicação para outras plataformas. Conclui-se que os sistemas móveis, para fiscalização de obras no terreno, aumentam a qualidade de um produto, porque permitem reduzir o tempo entre os processos de negócio. Facilitam a manutenção de um projeto e permitem que os novos processos adicionados tenham menor riscos e custos de implementação.

Palavras-chave: processos de negócio, fiscalização, aplicação móvel, multiplataforma, Android ,Xamarin, MvvmCross.

Esta página foi intencionalmente deixada em branco

Abstract

The mobile technology' revolution in society encourages the use of new methods of information treatment that are independent of time and space. For companies, these methods translate into exciting business opportunities. This thesis presents a mobile system for Android devices, with a multi-platform support. The system is intended for building inspection activities under urban policies and aims to streamline and to automate business processes of ground building inspection, throughout integration of the solution eUrban Company Link.
Initially, in this paper, the topic of the project is centered with the purpose of the application, through the presentation of various concepts related to business processes and the use of mobile technology in the field.
The work presents the technologies necessary for the system implementation, following the standards and architecture of generalized solutions of coding/re-coding for the type of application to develop.
For the execution of the application's prototype standards used for developing graphical user interfaces were investigated, and also a case study is evaluated, where a mobile application for supervision matters has a very satisfactory contribution, to this paper.
To implement this application the following software development processes were followed: listing requirements and functionality; description of used test-cases; the system classification diagram; the functionality' prototypes for devices; data communication formats between mobile system and server; system architecture with the multiplatform development technologies Xamarin and MvvmCross.
The graphical interface for Android is implemented through C # Visual Studio 2010 and Xamarin. The system uses standard Model-View-ViewModel (M-V-VM) architecture technology MvvmCross for sharing / partition coding, the database SQLite to store the objects' template and also synchronizes with the server via JSON web services.
Although there are some difficulties in implementation of a cross-platform application, it appears that the development methodology chosen was the most suitable. Graphical behavior-wise, performance compared to a single platform methodology allows developing system versions for other platforms quicker and cost-effective. Developing missing features and implementing the app to other platforms may be a solution for the Future. It's clinched that mobile systems for ground work supervision, increases the product's quality since it's possible to save time between business processes and it also facilitates the project maintenance and allow the new established processes to have lower risks and lower implementation costs.

Key-Words: business processes, inspection, mobile aplication, multiplatform, Android ,Xamarin, MvvmCross.

Esta página foi intencionalmente deixada em branco

Índice de Figuras

Figura 1 – Definição de processo de negócio 3

Figura 2 - A web como uma plataforma de desenvolvimento de aplicações 17

Figura 3 - Comparação entre aplicações nativas, web e hibridas 19

Figura 4 - Arquitetura Xamarin 21

Figura 5 - Como escolher o tipo de aplicação a desenvolver 24

Figura 6 - Objetivo do padrão MVC 25

Figura 7 - Arquitetura MVC simplificada 27

Figura 8 - Estrutura de classes do padrão MVP 30

Figura 9 - Arquitetura MVVM 31

Figura 10 - Padrões primários de navegação 35

Figura 11 - Exemplo de Navegação Springboard do Facebook 36

Figura 12 - Arquitetura geral do sistema 39

Figura 13 - Modelo de dados da aplicação 40

Figura 14 - Arquitetura de Software 42

Figura 15 - Detalhe do relatório 43

Figura 16 - Adicionar novo relatório diário 44

Figura 17 - Diagrama de casos de uso do sistema eUrban Móvel 59

Figura 18 - Diagrama de classes do sistema eUrbanMobile 61

Figura 19 - Integração via VPN+Wi-Fi entre o sistema móvel e fixo 63

Figura 20 - Integração via Wi-Fi com zona DMZ e LAN interna 63

Figura 21 - Integração em rede local por Wi-Fi entre os sistemas..... 64

Figura 22 - Integração via ligação de cabo de dados entre os sistemas 64

Figura 23 - Arquitetura geral do sistema..... 65

Figura 24 - Arquitetura do sistema..... 67

Figura 25 - Padrão M-V-VM do MvvmCross..... 68

Figura 26 - Padrão M-V-VM detalhado do MvvmCross 68

Figura 27 - Estrutura da projeto eUrbanMobile 73

Figura 28 - Frameworks suportadas pelo projeto..... 74

Figura 29 - Efetuar login 101

Figura 30 - Visualizar tarefas (modo normal)..... 101

Figura 31 - Visualizar tarefas (agrupadas por data) 102

Figura 32 - Menu da aplicação na lista de tarefas 102

Figura 33 - Configurar filtros de tarefas..... 102

Figura 34 - Sair da sessão..... 102

Figura 35 - Sincronizar tarefas (enviar e receber) 103

Figura 36 - Sincronização de tarefas 103

Figura 37 – Ecrã de sincronização de tarefas 103

Figura 38 - Resumo do processo 103

Figura 39 - Visualizar documentos da tarefa 104

Figura 40 - Adicionar ficheiros a uma tarefa 104

Figura 41 - Vistoria da tarefa 104

Figura 42 - Adicionar ficheiros (Perfil Chefia) 104

Figura 43 - Despachar uma tarefa (Perfil chefia) 105

Figura 44 – Ecrã de Configurações 105

Figura 45 - Efetuar login 106

Figura 46 - Visualizar tarefas (modo normal) 106

Figura 47 - Visualizar tarefas (agrupadas por data) 107

Figura 48 - Configurar filtros de tarefas 107

Figura 49 - Resumo do processo 108

Figura 50 - Visualizar documentos da tarefa 108

Figura 51 - Vistoria da tarefa 109

Figura 52 - Adicionar ficheiros a uma tarefa 109

Figura 53 - Despachar uma tarefa (Perfil chefia) 110

Figura 54 - Configurar sistema 110

Figura 55 - Ecrã de sincronização de tarefas 111

Figura 56 -Sincronização de tarefas 111

Figura 57 - Sair da sessão 112

Figura 58 - Versão Android (ecrã de sincronização) com controladores e sem MvvmCross 113

Figura 59 - Ecran de Login - Windows Phone 7 115

Figura 60 - Ecran de configurações - Windows Phone 7 116

Figura 61 - Tempos médio de algumas funcionalidades da aplicação (s-segundos; ms-milissegundos) consoante o número de tarefas envolvidas 117

Figura 62 - Tempo médio (milissegundos) de visualização da informação em alguns écrans, consoante o número de tarefas envolvidas 118

Figura 63 – Comparação do tempo médio (milissegundos) de execução de operações de inserção e escrita com operações de visualização de dados 118

Esta página foi intencionalmente deixada em branco

Índice de Quadros

Tabela 1 - Descrição do caso de uso – Efetuar login 92

Tabela 2 - Descrição do caso de uso - Agrupar tarefas 92

Tabela 3 - Descrição do caso de uso - Filtrar tarefas 93

Tabela 4 - Descrição do caso de uso - Configurar sistema 94

Tabela 5 - Descrição do caso de uso – Sincronizar 95

Tabela 6 - Descrição do caso de uso - Sair da sessão 95

Tabela 7 - Descrição do caso de uso - Escolher tarefa 96

Tabela 8 - Descrição do caso de uso - Ver resumo do Processo 97

Tabela 9 - Descrição do caso de uso - Visualizar documentos 97

Tabela 10 - Descrição do caso de uso - Carregar documentos 98

Tabela 11 - Descrição do caso de uso - Executar tarefa 99

Tabela 12 - Descrição do caso de uso - Despachar tarefa 100

Esta página foi intencionalmente deixada em branco

Lista de Siglas

ADT *Android Developer Tools*

API *Application Programming Interface*

BD *Base de Dados*

BPM *Business Process Management*

CRUD *Create, Read, Update and Delete*

CSS *Cascading Style Sheets*

DMZ *Demilitarized Zone*

ERP *Enterprise Resource Planning*

FFA *Field Force Automation*

GPS *Global Positioning System*

HTML *HyperText Markup Language*

IDE *Integrated Development Environment*

MVC *Model–View–Controller*

MVP *Model–View–Presenter*

MVVM *Model-View-View-Model*

PC *Personal Computer*

PCL *Portable Class Library*

PME *Pequenas Médias Empresas*

SDK *Software Development Kit*

UI *User interface*

USB *Universal Serial Bus*

VPN *Virtual Private Network*

W3C *World Wide Web Consortium*

WP *Windows Phone*

WPF *Windows Presentation Foundation*

XML *Extensible Markup Language*

Esta página foi intencionalmente deixada em branco

Índice

AGRADECIMENTOS ..... **III**

RESUMO ..... **V**

ABSTRACT ..... **VII**

ÍNDICE DE FIGURAS ..... **IX**

ÍNDICE DE QUADROS ..... **XIV**

LISTA DE SIGLAS ..... **XVI**

ÍNDICE ..... **XIX**

1. INTRODUÇÃO **1**

1.1 CONTEXTUALIZAÇÃO **1**

1.1.1 SISTEMAS MÓVEIS DE SUPORTE A ATIVIDADES DE GESTÃO DE LICENCIAMENTO URBANO 2

1.1.2 PROCESSO DE NEGÓCIO 3

1.1.3 GESTÃO DE PROCESSOS DE NEGÓCIO 4

1.1.4 AUTOMAÇÃO DO CAMPO DE TRABALHO 4

1.1.5 IMPACTO DAS TECNOLOGIAS MÓVEIS NAS EMPRESAS 5

1.1.6 USO DE DISPOSITIVOS MÓVEIS 6

1.2 CARACTERIZAÇÃO GERAL DO ESTUDO **6**

1.2.1 MOTIVAÇÃO DO PROJETO 6

1.2.1 OBJETIVOS DA INVESTIGAÇÃO DE SUPORTE AO PROJETO 7

1.3 ESTRUTURA DA TESE **7**

1.4 TERMINOLOGIA USADA **9**

2. REVISÃO DA LITERATURA **11**

2.1 TECNOLOGIAS DE DESENVOLVIMENTO **11**

2.1.1 APLICAÇÕES NATIVAS 12

2.1.2 APLICAÇÕES WEB PORTÁTEIS 14

2.1.3 APLICAÇÕES HÍBRIDAS 18
2.1.4 APLICAÇÕES MULTIPLATAFORMA COMPILADAS 20
2.2 ESTRATÉGIA DE DESENVOLVIMENTO 21
2.3 PADRÕES DE DESENHO 25
2.3.1 MVC 25
2.3.2 OBSERVER 29
2.3.3 MODEL VIEW PRESENTER 29
2.3.4 MODEL-VIEW-VIEWMODEL 30
2.3.5 INTERFACE DO UTILIZADOR 31
2.3.5.1 Avaliação heuristica 32
2.3.5.2 Interfaces de utilizador em plataformas móveis 33
2.4 ESTUDO DE CASO: "UM SISTEMA MÓVEL PARA INSPEÇÃO DE PROJETOS EM OBRAS DE CONSTRUÇÃO NO TERRENO" 36
3. METODOLOGIA DO PROJETO 45
3.1 LEVANTAMENTO DE REQUISITOS 45
3.1.2 REQUISITOS FUNCIONAIS 46
3.1.3 REQUISITOS NÃO FUNCIONAIS 49
3.2 DIAGRAMA DE CASOS DE USO 57
3.3 DESCRIÇÃO DE CASOS DE USO 59
3.4 DIAGRAMA DE CLASSES 60
3.5 PROTÓTIPOS FUNCIONAIS 61
3.6 ARQUITETURA 62
3.6.1 COMUNICAÇÃO ENTRE SISTEMAS 62
3.6.2 ARQUITETURA GERAL 64
3.6.3 ARQUITETURA DO SISTEMA 66
3.6.4 MVVMCROSS 67
3.7 FERRAMENTAS E TECNOLOGIAS DE DESENVOLVIMENTO 69
3.8 IMPLEMENTAÇÃO DO PROJETO 70
3.8.1 EURBANMOBILE.DROID 71
3.8.2 EURBANMOBILE.CORE 73
3.8.3 REGISTAR E USAR SERVIÇOS 75

3.8.4 DATABINDING 76

4. DISCUSSÃO DE RESULTADOS 79

4.1 METODOLOGIA ESCOLHIDA 79

4.2 LIMITAÇÕES E DIFICULDADES SENTIDAS 81

4.3 COMPARAÇÃO DOS RESULTADOS OBTIDOS 81

4.4 IMPLEMENTAÇÃO PARA OUTRAS PLATAFORMAS 83

5. CONCLUSÕES E TRABALHO FUTURO 85

5.1 CONCLUSÕES 85

5.2 TRABALHO FUTURO 86

BIBLIOGRAFIA 87

ANEXOS 91

ANEXO 1 – CASOS DE USO 91

ANEXO 2 – INTERFACES 101

ANEXO 3 – ANÁLISE DE DESEMPENHO 117

1. Introdução

Este capítulo começa com uma introdução, onde se procede à contextualização, breve alusão ao tema do projeto, o conceito de processo de negócio, a definição de gestão de processos de negócio, automação do campo de trabalho, o impacto das tecnologias móveis nas empresas e o uso de dispositivos móveis. De seguida apresentamos a caracterização geral do estudo, a exposição das questões de investigação e dos objetivos da investigação. Este capítulo termina com a apresentação da estrutura da tese e da terminologia usada.

1.1 Contextualização

Esta tese enquadra-se no mestrado em Engenharia Informática - Computação Móvel, lecionado na Escola Superior de Tecnologia e Gestão de Leiria. A Link, empresa de consultoria na área de Tecnologias de Informação e Comunicação, possui uma solução designada *eUrban*, destinada à automação de processos para a gestão do urbanismo, suportada essencialmente por tecnologias *Microsoft* e ferramentas *Business Process Management* (*BPM*) que permitem tramitar e controlar os processos administrativos da área do urbanismo e fiscalização. A tese descreve a criação de um módulo sob a forma de uma aplicação móvel, para *tablets* e *smartphones*, que sirva de interface com a solução eUrban, possibilite o funcionamento *online* e *offline* e que suporte no terreno as atividades de fiscalização urbanística.

Os técnicos/inspetores de fiscalização deslocam-se às obras com a documentação, em formato papel, necessária à vistoria e execução de tarefas de fiscalização. Assim que a vistoria é realizada o técnico necessita de voltar ao escritório onde atualiza os processos de fiscalização com os dados recolhidos no local da obra. Posteriormente, a chefia através do

sistema *eUrban*, dá a ordem de despacho a cada uma das tarefas executadas pelos técnicos, permitindo que os processos de fiscalização avancem.

Para simplificar e otimizar o processo de fiscalização de obras pretende-se que a aplicação móvel permita aos inspetores ter acesso, no seu dispositivo móvel, à informação do processo necessária à execução das suas tarefas de fiscalização (contexto do processo: dados, decisões, histórico, documentos), permitindo diversos mecanismos de interação, nomeadamente o preenchimento de formulários e relatórios, anexação de fotos capturadas com o dispositivo móvel, entre outros dados obtidos no dispositivo móvel. Para garantir o uso da aplicação no local da obra e atingir o objetivo pretendido, esta deverá possibilitar o funcionamento em modo *offline* e permitir sincronização com o eUrban.

O uso de tecnologia móvel no terreno ou local de obras de construção é um mercado com grande potencial. Estas aplicações podem ser acedidas pelos inspetores a qualquer instante, proporcionando um método mais conveniente de comunicação de dados de construção. Além disso, os dispositivos móveis oferecem várias vantagens adicionais, como a capacidade de tirar fotos, gravar vídeos, entre outras tarefas, potencialmente úteis aos inspetores de construção.

1.1.1 Sistemas móveis de suporte a atividades de gestão de licenciamento urbano

Os sistemas de suporte a atividades de gestão de licenciamento urbano são métodos e técnicas utilizadas para facilitar as ações de gestão e fiscalização de obras em regime urbanístico. Estes sistemas de suporte agrupam um conjunto ferramentas informatizadas que tem o objetivo de simplificar, ou otimizar os processos de negócio da empresa. Para a área de gestão de licenciamento urbano, os sistemas móveis de suporte, permitem aos seus utilizadores o acesso, no dispositivo móvel, e no local da obra, à informação do processo necessária à execução das suas tarefas, assim como, o preenchimento de formulários e relatórios digitais que substituem os métodos tradicionais de gestão e fiscalização.

1.1.2 *Processo de negócio*

Existem vários conceitos para processo de negócio, as definições de Davenport, Hammer e Champy são as mais usadas e aceites atualmente. Segundo Davenport (1993), um processo de negócio é uma ordenação específica das tarefas de trabalho no tempo e no espaço, com um começo, um fim, entradas e saídas claramente identificadas (…). Para Hammer e Champy (1993), os processos de negócio são um grupo de atividades realizadas numa determinada sequência lógica, com o objetivo de criar um serviço ou um produto de valor para grupo específico de clientes. Para Willoch (1994), consiste numa sequência de atividades coerentes que criam um valor apreendido pelos clientes. Já Rummler e Brache (1995) consideram que é uma série de etapas destinadas a produzir um produto ou serviço (…).

Em suma, podemos afirmar que um processo de negócio é um conjunto de ações relacionadas e interligadas entre si, que tem como objetivo determinar como será realizado o trabalho na empresa, originar um produto de qualidade para a organização e obter a satisfação por parte do cliente (ver figura 1).

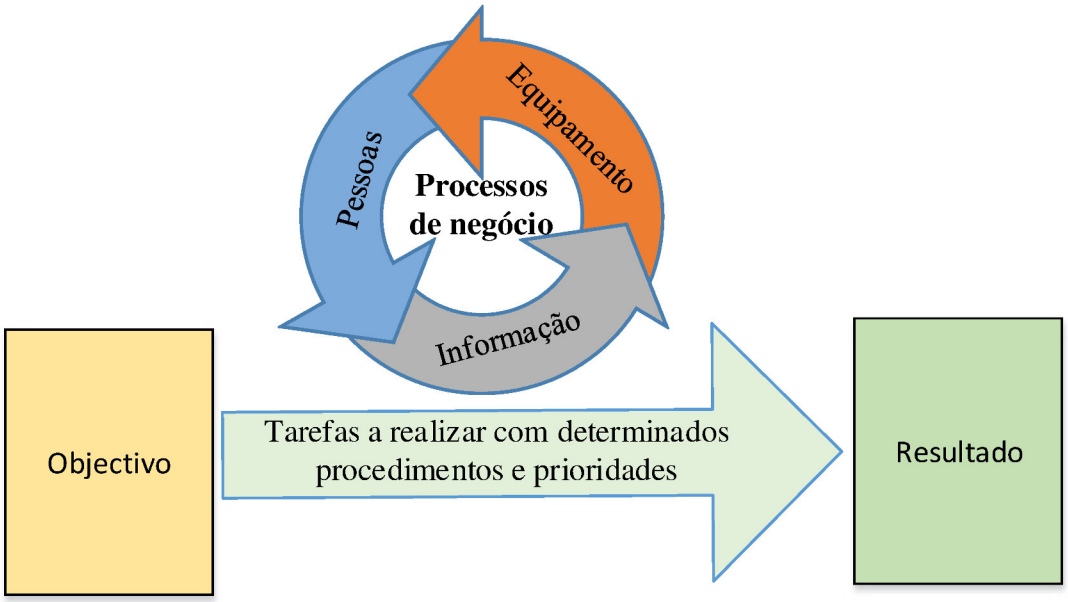


Figura 1 – Definição de processo de negócio

1.1.3 Gestão de processos de negócio

O conceito de gestão de processos de negócio tornou-se popular na década de 90. Consiste numa metodologia fundamental de análise, métodos e controle do desenvolvimento de processos e fluxos de trabalho numa organização (Hammer e Champy, 1993; Rummler e Brache, 1995). Para Ould (2005), esta é uma área que evolui a parte central do desenvolvimento do negócio, da organização e dos recursos de informação, através dos processos que constituem um ponto de partida na modelação do negócio e dos sistemas de informação.

A gestão de processos de negócio relaciona a gestão de negócios e as tecnologias de informação existentes. Permite otimizar e simplificar os processos de negócio através de técnicas de análise, modelação e controlo dos processos operacionais das organizações. Visa alcançar os objetivos das organizações, através de uma gestão e controlo eficaz dos seus processos de negócio, obtendo pontos de produtividade e eficiência.

Com o desenvolvimento de sistemas *Enterprise Resource Planning* (*ERP*) complexos, usados para controlo de estratégias de negócio das empresas, os *Business Process Management* (*BPMs*) surgiram devido à necessidade de extrair dos *ERPs* as regras de administração organizacionais, por forma a automatizar fluxos de forma rápida, eliminando o uso de outras aplicações customizadas e redução de custos às organizações.

1.1.4 Automação do campo de trabalho

Automação do campo de trabalho é um termo genérico para designar os serviços de aplicações móveis usados no campo de trabalho, que dão suporte, em tempo real, aos pedidos, agendamento, supervisão e elaboração de relatórios. Sobre este conceito, Nilsson, et al (2003), efetuaram um estudo sobre a viabilidade destes sistemas na indústria de construção da Suécia. Este estudo pretendia mostrar às pessoas, o impacto económico em operações especificas desta natureza, assim como, as possibilidades de melhorar as tarefas no campo de trabalho.

Por outras palavras, *Field Force Automation* (*FFA*) é a designação dada para a substituição dos processos tradicionais de captura de informação no campo de trabalho, em tempo real, pelo uso da tecnologia móvel.

1.1.5 Impacto das tecnologias móveis nas empresas

O crescimento da quantidade de dispositivos móveis disponibilizados pelos fabricantes, a convergência de informação e comunicação e os investimentos efetuados em infraestruturas sem fios, são exemplos de uma sociedade de internautas móveis, cada vez mais dependentes. Na comunidade académica também se tem notado uma clara evidência do uso desta tecnologia, através da conceção de sistemas de informação e as investigações efetuadas em vários domínios: o impacto dos dispositivos móveis, as aplicações desenvolvidas, as funcionalidades de localização geográfica, e o uso das comunicações nos dispositivos.

Uma perspetiva técnica de mobilidade de informação que está associada ao termo de computação móvel foi dada por Perry, et al (2001), (…) o acesso a pessoas e informação em qualquer hora e em qualquer lugar (…).

De la Garza e Howitt (1998), veem as tecnologias móveis como um meio para melhorar o fornecimento de informações e comunicação nas unidades operacionais do campo de trabalho com o escritório. Para Eisenblatter (2001), a obtenção de informações precisas no momento certo e no lugar certo é indispensável para o sucesso de um projeto de construção civil. Já Kimoto, Endo, Iwashita e Fujiwara (2005), referem que a utilização de papel num local de obras provoca uma baixa eficiência, devido à diferença entre o relacionamento do tempo e espaço para obter e processar as informações. Loefgren (2006) descreve que uma aplicação móvel utilizada por gestores de construção, a funcionar num *tablet*, permite reduzir o trabalho redundante e os tempos para obter informação.

As empresas ao adotarem dispositivos móveis conseguem elevar os padrões de produtividade dos seus colaboradores. Eles podem tomar as suas decisões de negócio mais rapidamente e em qualquer lugar, ganhando tempo de execução, portanto, reduzindo custos para as organizações.

1.1.6 Uso de dispositivos móveis

Segundo um estudo de pesquisa recente efetuado pelo Gartner Group, prevê-se que 1,2 mil milhões de dispositivos serão comprados pelas empresas em 2013, e que 66% dos trabalhadores móveis vão possuir um *smartphone* em 2016 [13]. Carolina Milanesi, vice-presidente de pesquisa do Gartner, afirma que os *smartphones* e *tablets* não vão substituir os *Personal Computers* (*PCs*) na maioria das empresas, mas a capacidade de processamento a custos razoáveis por parte destes aparelhos móveis tem mudado a forma como as organizações olham para as suas estratégias de negócio [13].

1.2 Caracterização geral do estudo

Nesta secção enquadramos o trabalho do projeto realizado e a pesquisa desenvolvida para o seu suporte. Mencionamos as questões e os objetivos da investigação.

1.2.1 Motivação do projeto

As questões de investigação surgem da conceção da aplicação que sustenta a tese. A implementação de uma aplicação móvel com suporte multiplataforma exige um estudo de investigação aplicada para que seja bem-sucedida. A importância do uso dos *smartphones,* e o papel ativo que tem na sociedade, tem contribuindo para o desenvolvimento de muitas tecnologias que dão suporte e simplificam cada vez mais os processos de negócio das empresas. Essas tecnologias admitem ser a resolução, contribuindo para o aumento da produção nos locais onde o uso de tecnologia móvel está pouco acessível. Técnicos de vistoria de obras e trabalhadores similares estão habituados ao uso de documentação, logo, a inclusão de tecnologia móvel que simplifique o seu

trabalho tem que ser ajustada às suas necessidades. Ao longo da revisão da literatura tentamos responder a várias questões:

1. Qual a melhor estratégia de desenvolvimento multiplataforma para elaboração da aplicação?
2. Qual a arquitetura multiplataforma que supre as necessidades de reutilização de código e partilha de lógica de negócio, serviços e modelo de dados?
3. Qual a interface gráfica compatível com dispositivos de diferentes características (tamanho, resolução, densidade)?

1.2.1 Objetivos da investigação de suporte ao projeto

A elaboração da investigação tem como principais objetivos:

1. Conhecimento das tecnologias/ferramentas para desenvolvimento de aplicações móveis multiplataforma.
2. Levantamento de padrões de desenho utilizados para soluções generalizadas e reutilização de código.
3. Conhecimento das normas utilizadas em desenvolvimento de interfaces do utilizador para aplicações móveis.
4. Contributo de sistemas móveis utilizados para fiscalização de obras.

1.3 Estrutura da tese

Este relatório organiza-se em cinco capítulos. Este primeiro capítulo divide-se em quatro partes: na primeira procede-se à contextualização do tema, descrevendo alguns conceitos relacionados com os processos de negócio e automatização das forças no campo de trabalho, relacionados com a fiscalização de obras de construção civil. Ainda na primeira parte deste capítulo é abordado o impacto e a previsão do uso das novas

tecnologias nas empresas. Na segunda parte são abordadas as questões e os objetivos da investigação. A terceira parte descreve como é organizada a tese em capítulos e no final deste capítulo concluímos com a referência a alguns termos estrangeiros e terminologia usada neste documento.

No segundo capítulo "Revisão da literatura" começa-se por diferenciar as tecnologias de desenvolvimento de aplicações móveis existentes e apresentamos uma estratégia de escolha da tecnologia mais recomendada para um projeto. Prosseguimos com a descrição de alguns padrões de desenho usados para a separação/reutilização de código em partes, e apresentamos normas usadas por peritos na construção de boas interfaces do utilizador. Este capítulo termina com um estudo de caso sobre o contributo de uma aplicação móvel de fiscalização de obras no local de trabalho.

No terceiro capítulo "Metodologia do projeto" descrevemos os passos necessários para criar uma aplicação móvel. Começamos pelo levantamento dos requisitos funcionais e não funcionais do sistema e a partir deles desenhamos e descrevemos os seus casos de uso. Avançamos para a construção do diagrama de classes e dos protótipos funcionais. De seguida apresentamos a arquitetura do sistema sobre os pontos de vista de comunicação de dados e tecnologias usadas. Mostramos quais as principais ferramentas e tecnologias usadas no projeto. No final deste capítulo expomos a divisão da aplicação em módulos, descrevendo cada um deles, e apresentamos os pontos chaves da tecnologia *M-V-VM*.

No quarto capítulo "Discussão de resultados" apresentam-se as justificações das tecnologias escolhidas, abordam-se as limitações e dificuldades sentidas na implementação do projeto. No final do capítulo discute-se o desempenho da aplicação e a implementação do sistema para outras plataformas.

No quinto capítulo "Conclusões e trabalho futuro" apresentam-se as conclusões e contributo científico do trabalho efetuado. Para terminar, abordam-se as perspetivas de trabalho futuro na aplicação.

1.4 Terminologia usada

Esta dissertação enquadra-se num contexto onde existem muitos termos e tecnologias que não tem tradução para português, por isso optou-se por mantê-los na sua designação original. É frequente apresentarmos abreviaturas ou siglas de termos utilizados na implementação de sistemas, que são muitas vezes comuns no mundo da informática e internautas, mas desconhecidos fora desse ambiente.

Ao longo desta tese referimo-nos a dispositivos móveis como sendo dispositivos de bolso ou aparelhos portáteis onde podemos executar aplicações, navegar na internet, entre outras coisas. Exemplos disso são os termos "*smartphone*" e "*tablet*", designações de dispositivos móveis, mas com dimensões e características diferentes.

Também utilizamos os termos aplicação, aplicação móvel, ou sistema para designar o projeto desenvolvido nesta tese, no entanto, o termo sistema engloba a aplicação e possíveis ficheiros dependentes dela, tais como a Base de Dados (BD).

Os termos "*Android*", "*iOS*" e "*Windows Phone*" (*WP*) não possuem tradução porque são o nome de sistemas operativos de dispositivos móveis. Pequenos excertos da programação da aplicação também estão presentes em sua linguagem natural, porque só assim fazem sentido para a compreensão do contexto onde são usados.

Neste capítulo efetuou-se a introdução e contextualização do tema da tese, descrevendo alguns conceitos relacionados e apresentando a organização do documento. No próximo capítulo aborda-se a revisão da literatura, a investigação do tipo de tecnologias de desenvolvimento de aplicações móveis e o seu contributo na área de fiscalização de obras de construção.

Esta página foi intencionalmente deixada em branco

2. Revisão da literatura

Este capítulo divide-se em quatro partes. Na primeira descrevem-se os tipos de aplicações móveis existentes: apresentando as suas vantagens e desvantagens, as ferramentas necessárias para implementa-los, e em que contexto estes modelos de aplicações são utilizados. São referidas as normas usadas para a construção de modelos de aplicações baseados na *web* e o contributo do *World Wide Web Consortium* (*W3C*) no desenvolvimento deste tipo de aplicações. Na segunda parte são descritas as técnicas/regras para a escolha do tipo de aplicação a desenvolver. A terceira parte expõe os padrões de desenvolvimento de aplicações que permitem a reutilização de código e separação em módulos. São descritas as normas para a conceção de interfaces gráficas de utilizador.

No final deste capítulo é apresentado um estudo de caso sobre a aplicabilidade de um sistema móvel na área de fiscalização de obras de construção, indicando os objetivos da aplicação, as tecnologias usadas, os modelos de dados da aplicação, a descrição da arquitetura do sistema e exemplos de interação do utilizador na aplicação.

2.1 Tecnologias de desenvolvimento

Nos últimos anos, fabricantes de dispositivos móveis colocaram no mercado uma imensa variedade de dispositivos, fabricados para diferentes tecnologias e sistemas operativos móveis, com o intuito de satisfazer diferentes perfis de utilizadores e empresas. Uma aplicação desenvolvida para um sistema operativo é apenas compatível com a gama de dispositivos fabricados para esse sistema. No entanto, muito se tem investigado e feito

com o objetivo de criar uma plataforma de desenvolvimento de aplicações móveis que sejam compatíveis com a maior quantidade de dispositivos existentes no mercado. Este é o maior e mais aliciante desafio da atualidade, monopolizar o mercado de desenvolvimento de aplicações móveis.

Vários modelos de desenvolvimento emergiram, alegando ser a solução para desenvolvimento de aplicações multiplataforma. Existem quatro tipos de categorias de aplicações para dispositivos móveis: Nativas, *Web* (portáteis), Híbridas e Multiplataforma (compiladas).

2.1.1 Aplicações nativas

Aplicações nativas são desenvolvidas usando linguagem específica para um tipo de plataforma (*Java* para *Android, Objective-C* para *iOS*, etc…) [14].

Vantagens:

- Dão ao utilizador a melhor experiencia de iteração com o seu dispositivo;
- São mais rápidas e fluidas do que aplicações não nativas;
- Permitem utilizar todos os recursos existentes nos dispositivos (camara, acelerômetro, bússola, *Global Positioning System* (*GPS*), etc);
- Podem ser vendidas nas lojas de aplicações do respetivo fabricante, e,g., (*Android Market*, ou a *App Store*). [15]

Desvantagens:

- É uma aplicação restrita à plataforma escolhida;
- Não permite reutilização de código em aplicações multiplataforma;
- Pode potenciar um custo elevado de desenvolvimento e manutenção;

- A atualização não é transparente ao utilizador, necessita de instalação de nova versão;
- Necessita de recursos humanos experientes para desenvolvimento em cada linguagem [15].

Aplicabilidade:

- Aplicações de realidade aumentada;
- Aplicações com necessidade a recursos de baixo nível, como a localização, bússola, giroscópio, camara, sensores de movimento, entre outros e acesso a funcionalidades do dispositivo, como o livro de endereços, e os contatos;
- Jogos de alto desempenho;
- Aplicações que envolvem integração de sistemas (notificações, contatos);
- Aplicações focadas no utilizador.

Ferramentas necessárias para desenvolvimento:

- *Android*
  - ✓ *Eclipse Integrated Development Environment (IDE)*;
  - ✓ *Android Software Development Kit (SDK)*;
  - ✓ *Android Developer Tools (ADT) Plugin*.
- *iOS*
  - ✓ *Mac OSX*;
  - ✓ *Xcode*;
  - ✓ *Interface Builder*;
  - ✓ *iOS Simulator*.
- *WP*
  - ✓ *Visual Studio 2010/12*

- ✓ *WP Emulator*
- ✓ *Silverlight* para *WP*
- ✓ *WP SDK*

2.1.2 Aplicações web portáteis

São aplicações que usam padrões abertos e podem ser executadas pelos navegadores dos dispositivos móveis. São desenvolvidas em linguagens compreendidas pelos navegadores *web* e por *Application Programming Interfaces* (*APIs)* que permitem a interação entre a aplicação e o dispositivo móvel [14].

Vantagens:

- São portáveis, estão limitadas apenas à compatibilidade dos navegadores com as suas funcionalidades;
- Permitem reutilização de código em ambiente multiplataforma;
- As atualizações para novas versões são transparentes ao utilizador [15];
- Aplicações móveis multiplataforma são mais fáceis de desenvolver usando linguagens *web*, porque não necessitam de recursos especializados direcionados para uma plataforma específica. [14]

Desvantagens:

- Não tem acesso direto a todos os recursos de físicos disponíveis nos dispositivos móveis, como: o acelerómetro, bússola, câmara, meios de comunicação, etc;
- Necessita de uma ligação quase constante à Internet, o que impossibilita a utilização da aplicação sem conexão à internet [14];

- A velocidade e desempenho são inferiores às aplicações nativas;
- As interfaces gráficas são menos ricas, proporcionando baixa usabilidade;
- Não podem ser colocadas na *App Store*;
- A especificação do *HyperText Markup Language 5* (*HTML5)* utilizada no desenvolvimento ainda não se encontra finalizada;
- Existe uma dificuldade em adaptar para dispositivos diferentes, com resoluções de tela variadas e diferentes navegadores [15];
- O suporte a vários navegadores pode provocar maiores custos de implementação e manutenção.

Ferramentas para desenvolvimento:

- *JQueryMobile*1
- *jQtouch*[2]
- *Sencha Touch*3

Aplicabilidade:

- Aplicações simples e com curto período de vida;
- Aplicações que requerem sempre uma ligação à internet;
- Aplicações que não envolvam o acesso aos recursos físicos do dispositivo.

Projeto MobileWebApp

Hoje em dia a web é uma ferramenta muito importante para o desenvolvimento de aplicações e serviços de internet, num ambiente *desktop*. No entanto, os serviços de

[1] JQueryMobile web site - http://jquerymobile.com/
[2] jQtouch web site - http://jqtjs.com/
[3] SenchaTouch web site - http://www.sencha.com/products/touch

internet para dispositivos móveis são maioritariamente oriundos de lojas de aplicações desenvolvidas por uma linguagem de programação nativa, ao invés de uma tecnologia universal. As aplicações *web* podem ser desenvolvidas rapidamente e com baixo custo, devido à quantidade de desenvolvedores existentes. Além disso, as pequenas médias empresas (PME) tem todo o interesse numa tecnologia web unificada, porque muitas vezes não dispõem de recursos especializados que reimplementem uma aplicação móvel para cada tipo de plataforma nativa.

O projeto *MobileWebApp* [16] suporta o uso de tecnologia *web* para o desenvolvimento de serviços de internet móvel, trazendo as vantagens de aplicações *web* do ambiente *desktop* para o mundo móvel. *MobileWebApp* tem suporte para a promoção de eventos, divulgação de imprensa, formação sobre a tecnologia, desenvolvimento de padrões e conjunto de testes na área de aplicações da *web* móveis. Este projeto é patrocinado e liderado pelo *W3C*, que agrupa várias centenas de investigadores, a indústria e é dirigido por Tim Berners-Lee, o inventor da *web* [17].

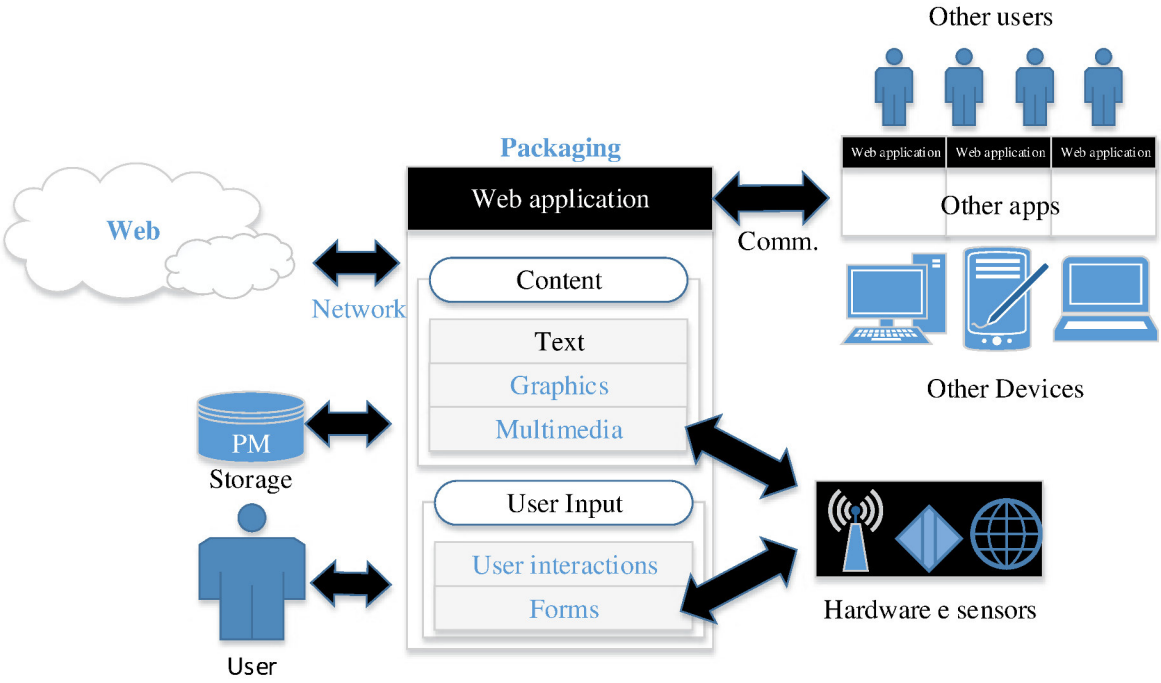
A *web* está sendo reestruturada, tecnologias como *HTML5* estão ajudando a criar padrões que deem suporte à maior variedade de navegadores e serviços de internet, com a finalidade de proporcionar um comportamento adequado a qualquer dispositivo.

O projeto *MobileWebApp* tem quatro principais objetivos [17] [18]:

Normalização das aplicações web móveis: consiste no aumento dos recursos e padrões para a web, permitindo que aplicações *web* móveis consigam competir com a capacidade das aplicações nativas. Através do desenvolvimento de *APIs* dispositivo, pretende-se que as aplicações *web* consigam uma melhor iteração entre os navegadores e os recursos dos dispositivos (camara, bateria, nível de rede, livro de endereços, agenda, etc..) usando *javascript*. O *W3C* desenvolve trabalho nessa área e tem obtido alguns progressos [19]. O *W3C* também é responsável pela criação de uma enorme quantidade de especificações que aumentam as potencialidades das aplicações *web*, e determinam o modo como estas se aplicam no domínio da mobilidade.

O projeto *MobileWebApp* desenvolveu e publicou um estado de normalização da arte, com mais de 50 especificações, que determina quais as normas mais relevantes para o

desenvolvimento de aplicações móveis [20]. Estas especificações estão distribuídas por várias categorias: gráficos, multimédia, formulários, interação com o utilizador, armazenamento de dados, gestão da informação, sensores e integrações de *hardware*, cobertura de rede, desempenho e otimização, etc. (ver figura 2).



Fonte: Forgue, M.C.; Hazaël-Massieux, D. - Mobile Web Applications: Bringing Mobile Apps and Web Together

Figura 2 - A web como uma plataforma de desenvolvimento de aplicações

Testes aos padrões das aplicações web móveis: os navegadores *web* possuem algumas incompatibilidades provocadas por inconformidades de padrões. Um conjunto de testes mais rigoroso e extenso abrange um maior conjunto de padrões e permite maior interoperabilidade entre as tecnologias *web* e os dispositivos móveis.

Formação ao nível do desenvolvimento: aumentar o número de pessoas com capacidade para desenvolver aplicações *web* móveis, permite uma redução de custos às empresas e aumento da velocidade de desenvolvimento. O projeto *MobileWebApp* criou dois cursos *online* destinados às pessoas que pretendam implementar aplicações *web* para dispositivos móveis:

- *Mobile Web* 1: Melhores Práticas [21].

- *Mobile Web* 2: Aplicações [22].

Alcance da divulgação: consiste em promover, consciencializar e informar as empresas europeias sobre os objetivos, potencialidades e resultados do trabalho do *W3C* em aplicações *web* móveis. [23]

2.1.3 Aplicações híbridas

São aplicações desenvolvidas com linguagens que a web entende, por exemplo, *HTML*, *Cascading Style Sheets* (*CSS*) e *javascript*, mas utilizam *APIs* nativas de cada plataforma para interagir com os recursos de *hardware* dos dispositivos. É efetuada uma separação entre a interface do utilizador (diferente em cada plataforma) e o código do corpo da aplicação. Isto permite que as aplicações sejam exportadas para diferentes plataformas e sejam executadas dentro do dispositivo [14].

Vantagens:

- Permite a reutilização de código da lógica de negócio da aplicação, e obter uma experiência do utilizador semelhante às aplicações nativas [24];
- Conseguem aceder e manipular alguns recursos físicos dos dispositivos;
- Podem ser usadas em múltiplas plataformas [14].

Desvantagens:

- Ainda não conseguem obter um nível de desempenho semelhante às aplicações nativas. [14]

Aplicabilidade:

- Aplicações empresariais pouco complexas.
- Aplicações multimédia.

Ferramentas para desenvolvimento:

- *Phonegap* (*Apache Cordova*) 4– esta *framework open source* usa linguagens padão *web*, como *HTML5*, *Javascrit* e *CSS* e suporta vários sistemas operacionais móveis (*iOS*, *Android*, *Blackberry*, *WP*, *Palm WebOS*, *Bada* e *Symbian*). Além disso possui *APIs* para trabalhar com alguns recursos dos dispositivos (acelerômetro, *GPS* / localização, camara, som, etc…)5 .
- *Appcelerator Titanium*6.
- *Rho Mobile*[7]

De seguida é apresentado um quadro que ilustra a comparação entre os principais pontos das várias vertentes abordadas anteriormente (ver figura 3).

	Acesso ao dispositivo	Velocidade	Custo de Desenvolvimento	Colocar aplicativos na loja online
Nativa	Total	Muito rápida	Muito caro	Disponível
Hibrida	Total	Muito rápida	Moderado	Disponível
Web	Parcial	Rápida	Moderado	Não disponível

Fonte: http://www.academia.edu/4233075/Cross-Platform_Mobile_Application_Development

Figura 3 - Comparação entre aplicações nativas, web e hibridas

[4] Phonegap - http://www.phonegap.com/
[5] Funções suportadas por PhoneGap - http://www.phonegap.com/about/feature
[6] Appcelerator Titanium - http://www.appcelerator.com/
[7] Rho Mobile. Rho Mobile Inc., http://www.rhomobile.com

2.1.4 Aplicações multiplataforma compiladas

São desenvolvidas usando uma linguagem única e *APIs* que acedem às funcionalidades nativas da plataforma. As *UI* são estritamente nativas, mas o código da aplicação é comum a todas as plataformas [24].

Vantagens:

- Custo de implementação é menor comparado às aplicações nativas;
- A aplicação é desenvolvida usando apenas uma única linguagem (*c#*, *c++*,etc…).

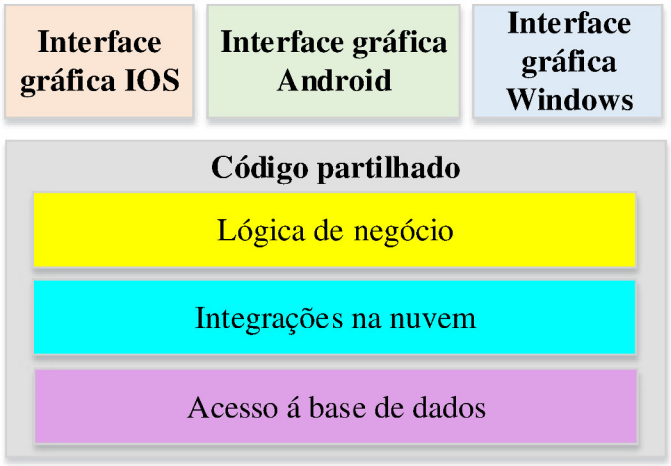
Desvantagens:

- É necessária a construção de uma interface gráfica para cada plataforma móvel.
- Exige a instalação da aplicação em cada dispositivo, como as aplicações nativas.
- Algumas ferramentas de desenvolvimento são pagas.

Ferramentas para desenvolvimento:

- *Xamarin*[8] – é uma biblioteca que permite o desenvolvimento de aplicações multiplataforma para *iOS*, *Android* e *WP*. O desenvolvimento é feito em *C#* com recurso às *APIs* (*MonoTouch*, *MonoDroid* e *WP SDK*) que acedem às funcionalidades nativas dos dispositivos.

[8] Xamarin site oficial - http://xamarin.com/



Fonte: Xamarin - http://xamarin.com/Windows

Figura 4 - Arquitetura Xamarin

Como mostra a figura, para cada plataforma é desenvolvida uma interface gráfica distinta. A Plataforma *Xamarin* permite desenvolver um bloco de código partilhado que poderá ser reutilizado para ambas as plataformas. O bloco de código comum acede aos componentes das interfaces gráficas através de *APIs* integradas. Para *iOS* é necessário o programa *MonoDevelop* integrado com as *APIs Monotouch* (*interface iOS*) e *MonoDroid* (*interface Android*). Em sistemas operativos *Windows* é possível desenvolver as aplicações *Android* e *WP* no *Microsoft Visual Studio*, ou com o programa *MonoDevelop*. O código compartilhado em *C#* juntamente com o módulo da interface gráfica é passado por um compilador que gera uma versão para a plataforma pretendida. A grande desvantagem para quem desenvolve usando *Xamarin* é que esta *API* necessita de uma licença paga para funcionar em dispositivos e sem limitações [25].

2.2 Estratégia de desenvolvimento

A escolha do (s) modelo (s) de desenvolvimento de uma aplicação móvel pode ser mais difícil do que se imagina. É claro que todos desejamos ter uma aplicação rápida, que possua a melhor experiência de utilizador possível, que consiga manipular todos os recursos e funcionalidades dos dispositivos, que tenha um custo de implementação razoável e que funcione em todas as plataformas e dispositivos existentes. Devido à grande

diversidade de dispositivos espalhados por várias linguagens de programação, ainda não existe uma solução que consiga obter essas características numa aplicação móvel. Algumas aplicações devem ser desenvolvidas em formato nativo para cada plataforma, enquanto outras podem ser implementadas com soluções híbridas.

Segundo Longhorn (2012), não existe uma solução perfeita apropriada que cumpra todos os requisitos das aplicações do Mundo, no futuro poderá existir, mas para já ainda existe um lugar para cada tipo de aplicação. A seleção deve ser cuidada e baseada nas necessidades reais do projeto.

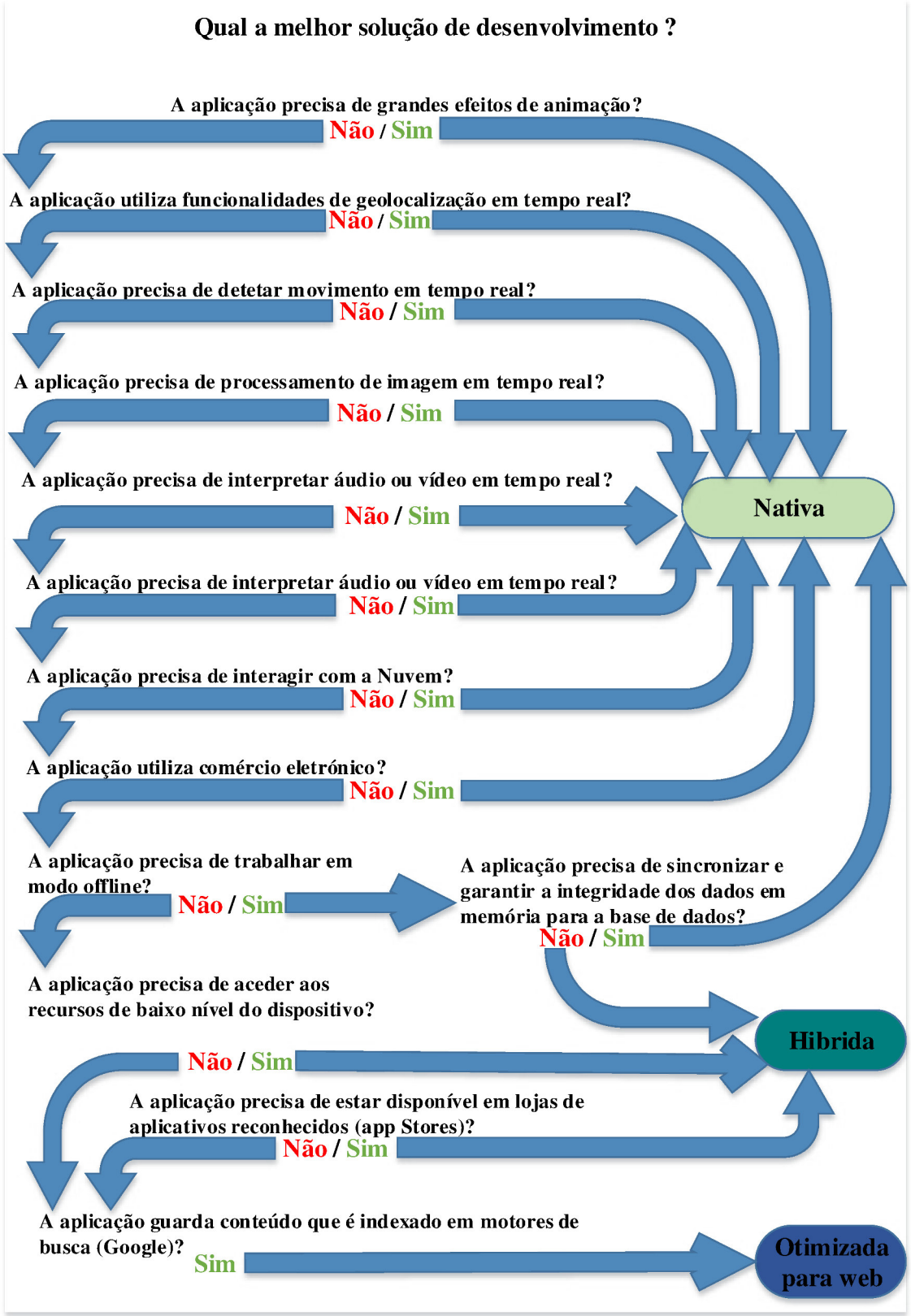
Basta olhar para o caso do *Facebook*, por questões de desempenho, refez sua aplicação *iOS* em código *Objective-C* nativo [27]. Numa entrevista dada durante a conferência "Disrupt" promovida pela portal "TechCrunch", em meados do mês de Setembro do ano passado, Zuckerberg, fundador do *Facebook*, afirma que "'Perdemos dois anos. As aplicações nativas serão agora a nossa nova estratégia para o *iOS* e o *Android"*" [28].

Antes de decidirmos qual a arquitetura a adotar no desenvolvimento de uma aplicação móvel, devemos tentar responder a uma série de questões que definem a melhor estratégia a seguir [29]:

- A aplicação precisa de grandes efeitos de animação?
- A aplicação utiliza funcionalidades de localização em tempo real?
- A aplicação precisa de detetar movimento em tempo real?
- A aplicação precisa de processamento de imagem em tempo real?
- A aplicação precisa de interpretar áudio ou vídeo em tempo real?
- A aplicação precisa de interagir com a *iCloud*?
- A aplicação utiliza comércio eletrónico?
- A aplicação precisa de trabalhar em modo *offline*?

- A aplicação precisa de sincronizar e garantir a integridade dos dados em memória para a BD?
- A aplicação precisa de aceder aos recursos de baixo nível do dispositivo?
- A aplicação precisa de estar disponível em lojas de aplicações reconhecidas (*app Stores*)?
- A aplicação guarda conteúdo que é indexado em motores de busca (*Google*)?

"Sushi mobile", uma empresa que desenvolve aplicações móveis, tem um fluxograma que ajuda a escolher a estratégia de desenvolvimento a seguir: nativa, *web* ou hibrida (ver figura 5) [29].



Fonte: Sushimobile - http://sushmobile.co.nz/mobile-application-development-native-hybrid-or-web-optimised-mobile/

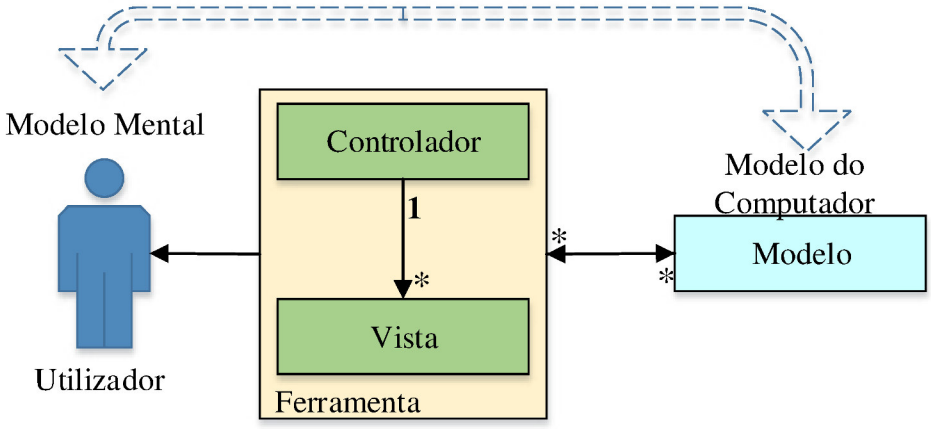
Figura 5 - Como escolher o tipo de aplicação a desenvolver

2.3 Padrões de desenho

Um padrão de desenho é um modelo que descreve a solução generalizada e reutilizável para um problema recorrente no desenvolvimento de aplicações orientadas a objetos [30]. É uma solução modelo documentada para um problema que já alguém teve e conseguiu resolver, permitindo que seja usada e adaptada para resolver problemas semelhantes em outras aplicações.

2.3.1 MVC

O *Model-View-Controller* (*MVC*) é um padrão de arquitetura para soluções de aplicações que separa diferentes aspetos de uma aplicação (a entrada lógica dos dados, as regras de negócio e a interface do utilizador), mas fornece um acoplamento entre os elementos [31]. Este padrão foi criado por Trygve Reenskaug (1979), enquanto trabalhava no ambiente "Smalltalk", na "Xerox PARC" [33]. O padrão de desenvolvimento é composto por três camadas (modelo, vista e controlador). O objetivo essencial do *MVC* é preencher a lacuna entre o modelo mental do utilizador e o modelo digital que existe no computador [34] (ver figura 6).



Fonte: http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

Figura 6 - Objetivo do padrão MVC

Modelo

Segundo ReensKaug (1979), o modelo é uma representação de uma abstração na forma de dados em um sistema computacional.

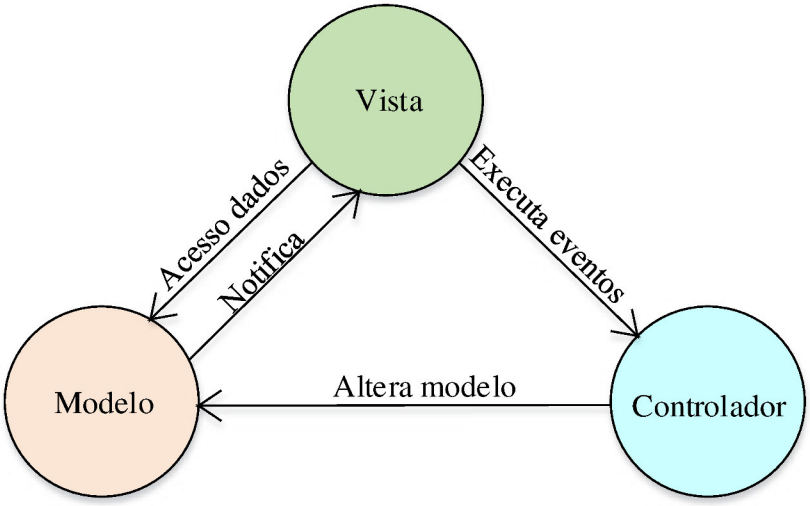
O Modelo é o componente que trata da persistência dos dados da aplicação. Ele representa os dados do domínio e contém as regras para acesso e manipulação dos dados da BD, servindo de interface entre o repositório de dados e o Controlador. O Modelo não pode ter qualquer referência para a Vista ou Controlador da aplicação [35].

Vista

A Vista é responsável por mostrar os dados, obtém os dados do modelo e especifica a forma como são apresentados ao utilizador. A Vista pode ser qualquer interface do utilizador que respeite o Modelo de dados da aplicação [35].

Controlador

O Controlador é responsável por gerir os pedidos realizados na Vista e as consultas ao Modelo. Ele transfere as iterações do utilizador na Vista, para ações que são realizadas pelo Modelo. Também interpreta as ações realizadas pelo Modelo e atualiza a Vista com as alterações (ver figura 7) [35].



Fonte: http://www.codeproject.com/Articles/42830/Model-View-Controller-Model-View-Presenter-and-Mod

Figura 7 - Arquitetura MVC simplificada

O aparecimento da internet veio impulsionar este padrão, devido à necessidade de executar aplicações com a lógica do negócio separada da interface com o utilizador. Assim os utilizadores de internet apenas necessitam de um navegador *web* para executar os seus serviços/aplicações, ao invés de serem instalados na própria máquina. Este padrão é muito utilizado, porque permite desenvolver mais facilmente as aplicações que são independentes do contexto, ambiente, ou dispositivo, onde são executadas. Por exemplo, as aplicações móveis multiplataforma podem ter uma arquitetura e comportamento muito semelhante, no entanto, a interface de utilizador é diferente entre as plataformas móveis.

MVC e Aplicações móveis

O modelo *MVC* é utilizado em aplicações móveis. Através deste padrão de desenvolvimento facilmente criamos uma aplicação dividida em módulos independentes, que permitem uma melhor escalabilidade e reutilização de código da aplicação, até mesmo para projetos posteriores. Já que cada módulo não contém referências dos outros, podem ser desenvolvidas e acrescentadas novas funcionalidades a um módulo, deixando os outros praticamente inalterados. Por exemplo, numa aplicação *Android*, a Vista corresponde à interface de utilizador construída em *Extensible Markup Language (XML)*, o Modelo constrói e manipula as representações dos objetos através do *SQLite* e o Controlador, em

linguagem *Java*, trata dos eventos despoletados na interface do utilizador, alternando entre a atualização da interface e as alterações ao modelo de dados.

Vantagens

Utilizando o *MVC* consegue-se gerir e manter mais facilmente uma aplicação devido à separação em módulos estruturados. A longo prazo é mais fácil identificar erros e adicionar novas funcionalidades à aplicação. *MVC* permite gerir múltiplas interfaces utilizando o mesmo Modelo, porque o Controlador é que seleciona a Vista apropriada para mostrar os dados, ou seja, através de um mecanismo de notificação, diferentes interfaces de utilizador podem ser sincronizadas com o mesmo modelo de dados. A transferência da aplicação para outra plataforma não afeta o núcleo funcional da aplicação, são necessárias apenas implementações nas Vistas e alteração dos componentes nos Controladores [35].

Desvantagens

A estrutura *MVC* nem sempre é a mais aconselhada, a separação de uma aplicação simples em camadas, aumenta a complexidade sem receber qualquer flexibilidade. Em algumas aplicações existe a possibilidade de ocorrer um grande número de atualizações, no entanto, poderá haver Vistas que não estão interessadas na propagação das mudanças efetuadas no Modelo, então o Modelo precisa de saltar as desnecessárias notificações de alteração. Outras aplicações possuem um grande número de Vistas e Controladores, a passagem desses componentes para outra plataforma requer a separação do código dependente da plataforma antes da reescrita, a fim de encapsular dependências da plataforma. [35]

Além disso, para implementar o modelo *MVC* em aplicações complexas são necessários recursos humanos especializados para análise do sistema e estruturação da arquitetura.

2.3.2 Observer

O padrão Observer é um padrão de desenho de *software* utilizado no padrão *MVC*. Segundo Gamma, E., et all (1994), o padrão Observer "*… define uma dependência um-para-muitos entre objetos para que quando um objeto mudar de estado, todos os seus dependentes sejam notificados e atualizados automaticamente*...". Ele contém uma lista de observadores que são notificados automaticamente de eventuais mudanças de estado, geralmente chamando um de seus métodos. Este padrão permite que sejam manipulados os eventos de sistema, atualizando as diversas interfaces de utilizador que dependem de um modelo de dados. Desta forma é possível que diferentes Vistas sejam alteradas simultaneamente.

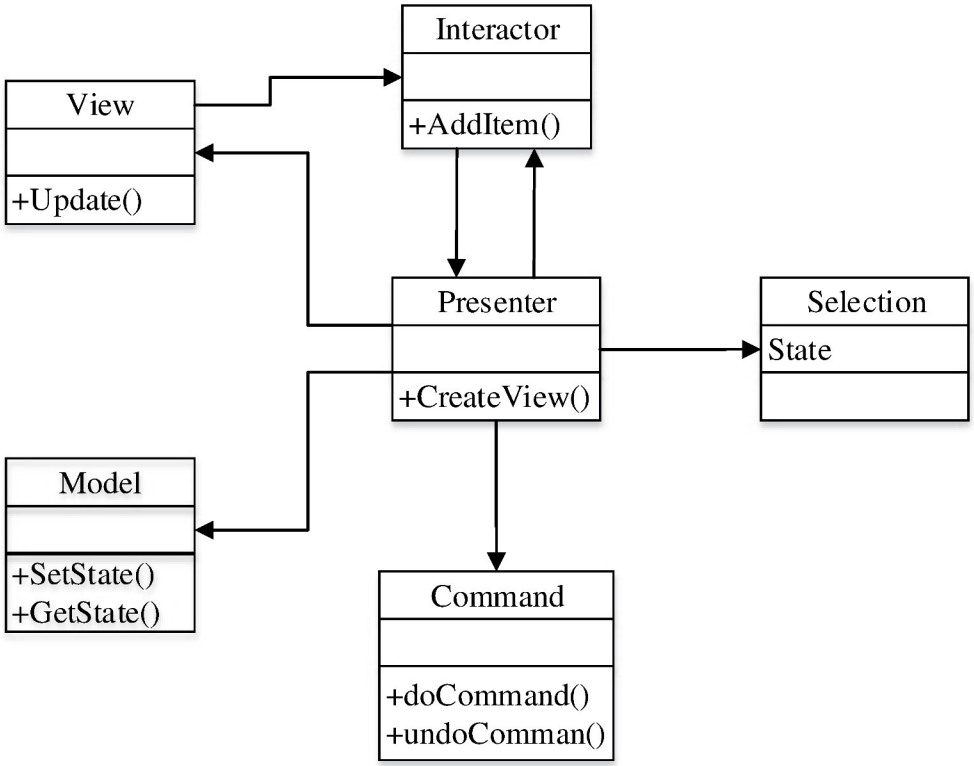
2.3.3 Model View Presenter

O padrão de desenho *Model View Presenter* (*MVP*) foi apresentado por Potel (1996) como uma variante ao padrão *MVC*. Segundo Potel (1996), como as interfaces de utilizador dos sistemas operativos modernos transferiram para a Vista uma parte das funcionalidades e inteligência existentes no Controlador, ele propôs um modelo que servisse de suporte, tanto para as Aplicações Internet Ricas, como para as aplicações *desktop* tradicionais. Este padrão, muito semelhante ao anterior, é composto por: Modelo, Vista e Apresentador. Tal como no padrão *MVC*, o Apresentador, equiparado ao Controlador, age como um mediador entre o Modelo e a Vista. Potel (1996), após analisar os tipos de interações entre a Vista e o Modelo, criou classes distintas para agrupar as ações do utilizador de seleção, execução de comandos e geração de eventos, e encapsulou-as na classe Apresentador (ver figura 8).

Na versão de Andy Bower e Blair McGlashan (2000), a Vista absorve a funcionalidade do Controlador e exibe os dados, o Apresentador pode aceder e atualizar diretamente a Vista e o Modelo.

Segundo Martin Fowler (2006), existem duas abordagens distintas para o padrão *MVP*, a abordagem de Potel preocupava-se com a eliminação do Controlador, delegando mais trabalho para a Vista. Fowler (2006) chamou-lhe abordagem Controlador de

Supervisão. A segunda abordagem, Vista Passiva, a de Bower e McGlashan, em que o Apresentador consegue atualizar a Vista diretamente.



Fonte: http://www.codeproject.com/Articles/42830/Model-View-Controller-Model-View-Presenter-and-Mod

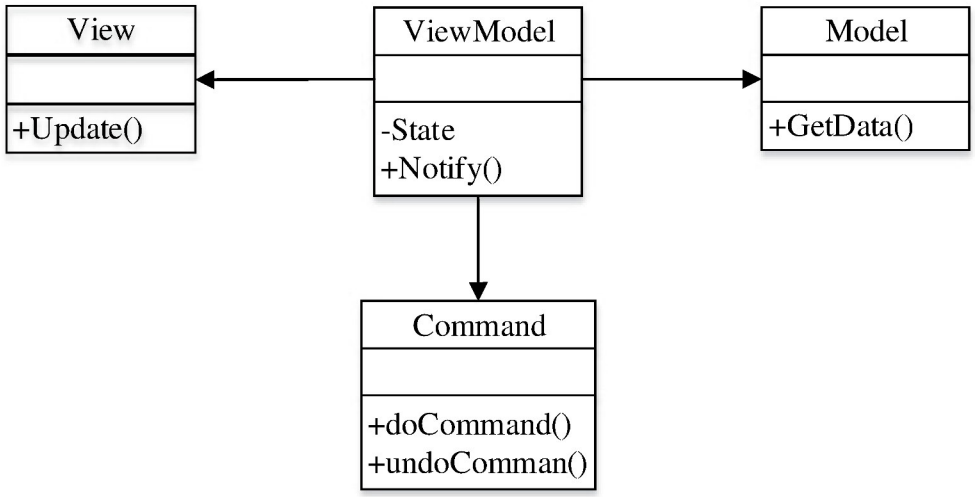
Figura 8 - Estrutura de classes do padrão MVP

2.3.4 Model-View-ViewModel

Outra variante do *MVC* é o *Moodel-View-ViewModel* (*MVVM*). Grossman (2005), arquiteto *Windows Presentation Foundation (WPF)* e *Silverlight*, foi o primeiro a revelar o padrão Modelo-Vista-VistaModelo no seu *blog*. Mais tarde, Smith (2009) descreveu o padrão em profundidade no seu artigo "*WPF Apps with the Model-View-ViewModel Desing Pattern*".

Segundo Smith (2009), a implementação da Vista deve ser da responsabilidade dos *designers* de *software*, isenta de qualquer código. Desenvolvedores *WPF* usam um *DataContext* para criar uma referência da Vista para a VistaModelo. Esta por sua vez, apresenta à Vista os dados contidos em objetos do Modelo. Quando os valores das

propriedades da VistaModelo mudam, os novos valores são propagados automaticamente para a Vista, através da ligação de dados *WPF*. A VistaModelo executa todas as modificações feitas ao modelo de dados. As classes da Vista não têm conhecimento da existência das classes do Modelo, nem o Modelo sabe da existência das outras partes envolvidas [42] (ver figura 9). Este padrão torna um projeto muito flexível, porque as partes envolvidas podem ser desenvolvidas e testadas separadamente.



Fonte: http://www.codeproject.com/Articles/42830/Model-View-Controller-Model-View-Presenter-and-Mod

Figura 9 - Arquitetura MVVM

2.3.5 *Interface do Utilizador*

Com o desenvolvimento da tecnologia móvel, os *smartphones*, ao contrário dos telemóveis tradicionais, já não são usados apenas para telefonar e enviar chamadas. Hoje em dia, podemos encontrar no mercado dispositivos móveis com capacidade de processamento diversificada e com telas de diferentes tamanhos e resoluções suportadas. Desenvolver *interfaces* de utilizador para este tipo de dispositivos constitui um grande desafio, porque para além da necessidade de adaptação às características dos dispositivos, ainda tem que garantir níveis de usabilidade com os utilizadores [43] [30].

Segundo Moran (1981), "*a interface de utilizador deve ser entendida como sendo a parte de um sistema computacional com a qual uma pessoa entra em contato — física, percetiva ou conceitualmente*". Esta definição da perspetiva para *interface* de utilizador é

caracterizada por duas partes: uma física, que o utilizador entende e manipula, e outra parte conceitual, em que o utilizador interpreta, processa e raciocina.

2.3.5.1 Avaliação heuristica

Desenvolvida por Jakob Nielsen e Rolf Molich (1990), esta é uma técnica de avaliação de usabilidade onde especialistas procuram identificar problemas de usabilidade numa *interface* com o utilizador, por exemplo, nos menus, nas caixas de diálogo, ou na navegação, através da análise e interpretação de um conjunto de princípios ou heurísticas [45]. Este método ainda é hoje aplicado, tanto para *interfaces desktop*, como para *interfaces* de dispositivos móveis.

Nielsen e Molich (1990) desenvolveram os dez princípios fundamentais da usabilidade, que avaliam a conceção de uma *interface* de utilizador:

- **Visibilidade do estado do sistema** – O sistema deve manter os utilizadores informados do que está a acontecer através de informação relevante fornecida em tempo útil. Por exemplo, o utilizador deve ser informado do estado de uma tarefa quando a sua execução dura um tempo considerável;
- **Relação entre o sistema e o mundo real** – O sistema deve falar a língua do utilizador, através de palavras e conceitos que lhe sejam familiares, em vez de palavras técnicas. Devem ser seguidas as convenções do mundo real, fazendo com que a informação surja de uma forma lógica e ordenada;
- **Controlo e liberdade do utilizador** – Os utilizadores cometem erros frequentemente e necessitam de uma saída de emergência claramente definida no sistema, sem que haja necessidade de passar por um extenso diálogo. Ele deve poder anular ou refazer as suas ações;
- **Consistência e padrões** : Os utilizadores não deverão ter de adivinhar se certas palavras, situações ou ações são sinónimos de termos que lhe são

familiares. Devem ser seguidas as convenções da plataforma para a qual se produz;

- **Prevenção de erros** – Melhor do que uma boa mensagem de erro, é um desenho cuidado das funções que impeçam o erro ocorrer em primeiro lugar;

- **Reconhecimento e não lembranças** – Os objetos, ações ou opções devem estar sempre visíveis. O utilizador não deve ter de se lembrar de informações de uma página para outra para completar a tarefa. As instruções de uso devem estar visíveis e ser facilmente alcançáveis quando necessário;

- **Flexibilidade e eficácia de uso** – Os aceleradores desconhecidos pelos utilizadores menos experientes – são modos de facilitar as ações dos utilizadores, o que possibilita que o sistema comporte à utilização tanto dos utilizadores experientes como dos inexperientes. O sistema deve permitir ao utilizador acelerar ações frequentes;

- **Estética e *design* minimalistas** – Os diálogos não devem conter informação irrelevante ou raramente necessária. A introdução de tais informações compete com a informação relevante, retirando-lhe visibilidade;

- **Ajuda ao reconhecimento, diagnóstico e correção dos erros do utilizador** – As mensagens de erro devem ser escritas em linguagem comum, sem códigos, indicando precisamente o problema encontrado e sugerindo soluções para tal;

- **Ajuda e documentação** – O ideal será se o sistema puder ser utilizado sem documentação, no entanto pode ser necessário fornecer algumas ajudas ao utilizador. Tal informação deve ser fácil de pesquisar, focada na tarefa do utilizador e deve enumerar os passos a seguir sem entrar em grandes delongas e explicações complexas. [46] [43]

2.3.5.2 *Interfaces* de utilizador em plataformas móveis

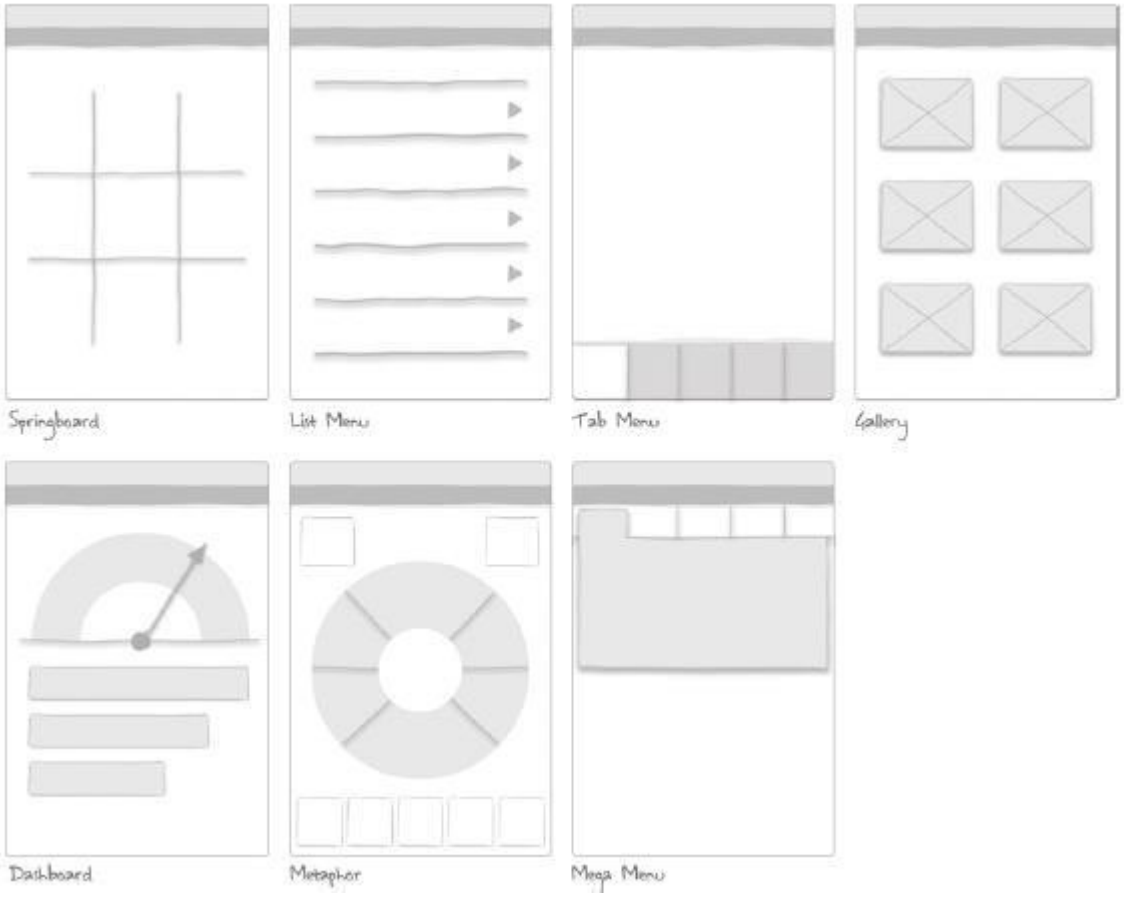
Quando pretendemos desenvolver uma aplicação móvel focada no utilizador, tentar disponibilizar a melhor experiência de utilizador é um dos fatores mais importantes. Cada plataforma móvel tem sua própria *interface* de utilizador: são diferentes tecnologias; componentes com comportamentos nativos da plataforma, que tem a sua própria forma de iteração com o sistema; etc…. Por isso é impensável tentar disponibilizar a mesma experiência de utilizador entre plataformas diferentes. Para resolver este problema existem as orientações/diretrizes (*guidelines*) disponibilizadas pelos fabricantes para a conceção de *interfaces* de utilizador, com os padrões aconselhados para determinada plataforma. [47]

Profissional em *interfaces* de utilizador, Theresa Neil 9 é autora de um livro que ilustra os vários conceitos para conceção de *interfaces* para aplicações móveis em plataformas distintas. Theresa Neil (2012) divide os padrões de desenho, para conceção de *interfaces* móveis, nas seguintes categorias de funcionalidades:

- Menus de navegação;
- Formulários de entrada de dados;
- Tabelas e listas para acesso à informação;
- Pesquisa, ordenação e filtros de dados;
- Ferramentas/controles importantes de suporte;
- Representação da informação através de gráficos;
- Convites para o utilizador interagir com o sistema;
- *Feedback* do resultado das ações do utilizador;
- Ajudar o utilizador a desempenhar uma tarefa;
- Más práticas (maus padrões) para a construção de *interfaces*.

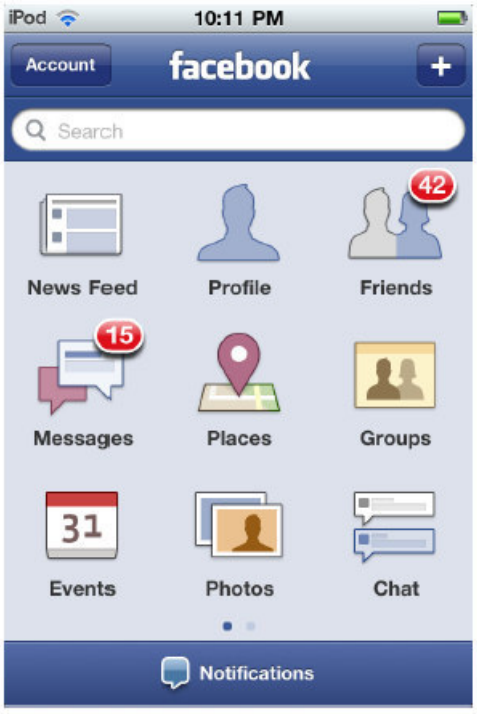
[9] Theresa Neil Web Site: http://www.theresaneil.com/

Cada uma das categorias contém vários padrões de desenho que permite ajustar o tipo de aplicação à funcionalidade, aparência e experiência do utilizador desejada. Por exemplo, Theresa Neil (2012) afirma que para telas de navegação principal existem à disposição seis tipos de padrões: *Springboard , List Menu, Tab Menu, Gallery, Dashboard, Metaphor e Mega Menu* (ver figura 10 e 11).



Fonte: Theresa Neil, "Mobile Design Pattern Gallery, Color Edition", UI Patterns for iOS, Android, and More

Figura 10 - Padrões primários de navegação



Fonte: Theresa Neil, "Mobile Design Pattern Gallery, Color Edition", UI Patterns for iOS, Android, and More

Figura 11 - Exemplo de Navegação Springboard do Facebook

As *Interfaces* de utilizador móveis bem estruturadas têm elevado impacto na experiência do utilizador. O utilizador executa as tarefas com maior facilidade se a *interface* for intuitiva, consistente e simplista.

2.4 Estudo de caso: "Um sistema móvel para inspeção de projetos em obras de construção no terreno"

Este estudo demonstra o contributo do uso de dispositivos móveis para os processos de negócio da organização em causa.

Problema

Inspetores e empreiteiros de obras estão constantemente no local da construção, e portanto, tem acesso limitado a um computador dedicado. A fiscalização de obras de construção pode tornar-se num processo complicado de gerir.

Definição / descrição do estudo de caso

Neste estudo de caso é apresentada uma solução móvel de fiscalização de infraestruturas de construção para os inspetores, a ser usada no local da obra. Esta aplicação móvel para dispositivos com sistema operativo *Android* permite ao inspetor qualificado, a fiscalização e supervisionamento dos materiais usados no projeto de construção e das atividades realizadas pelos empreiteiros e subempreiteiros de obras de construção, no local. Este projeto foi desenvolvido por um grupo de estudantes de uma Universidade da Flórida e teve o apoio da "Info Tech"10, uma empresa conceituada no desenvolvimento de produtos e serviços para agências de transporte norte-americanas, empresas de engenharia e empreiteiros. [49]

Objetivos

A info Tech, empresa patrocinadora do projeto, estipulou os seguintes objetivos [42]:

- Desenvolver um aplicativo móvel diário de fiscalização no terreno para os inspetores de construção;
- Proporcionar o acesso a recursos uteis dos *smartphones*, como a camara e o *GPS*;
- Desenvolver um acessório de *hardware* que permita registar a temperatura, humidade, e outros dados relevantes no local, oferecendo medidas muito precisas aos inspetores;

[10] Info Tech - https://www.infotechfl.com/

- Integrar o sistema desenvolvido com outras soluções da Info Tech.

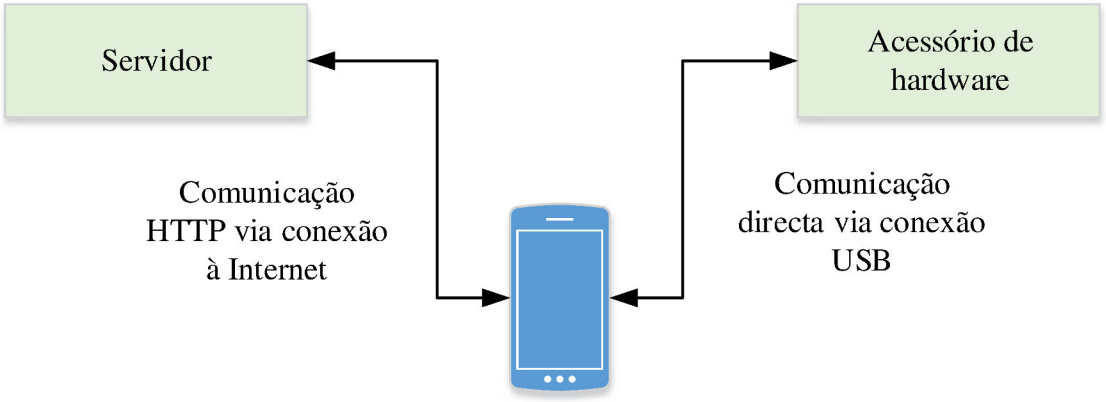
Tecnologias / técnicas utilizadas

Para conseguir estes objetivos foi criada uma aplicação em *Android* que permite aos inspetores: criar, visualizar e modificar relatórios diários a partir do *smartphone*. A aplicação também fornece um acesso fácil à câmara embutida no dispositivo, permitindo otimizar e agilizar o processo de captura de fotos no local. Para monitorizar fatores climáticos, a aplicação desenvolvida é compatível com o protocolo *Android Open Acessory*11, que permite a transferência de dados capturados por um acessório, para o dispositivo, através de uma conexão *Universal Serial Bus* (*USB*). A aplicação também é integrada com sistemas da Info Tech, através de uma aplicação (*Server Backend*) existente nos servidores da empresa. Desta forma, o dispositivo irá puxar os dados do projeto a partir da nuvem e sincronizará os dados do relatório quando forem introduzidos. O aplicativo mantém no dispositivo uma cópia de todos os dados necessários, por forma a permitir o seu funcionamento em modo *offline* [49].

Solução geral do sistema

O sistema móvel de fiscalização no terreno é composto por três componentes principais: a aplicação móvel para *Android*, um acessório de *hardware* e um sistema servidor *backend* (ver figura 12) [49].

[11] Accessory Development Kit - http://developer.android.com/tools/adk/index.html



Fonte: Manuel E. Bermudez. A Mobile Field Diary System for Infrastructure Construction Project Inspections, 10th Latin American and Caribbean Conference for Engineering and Technology.

Figura 12 - Arquitetura geral do sistema

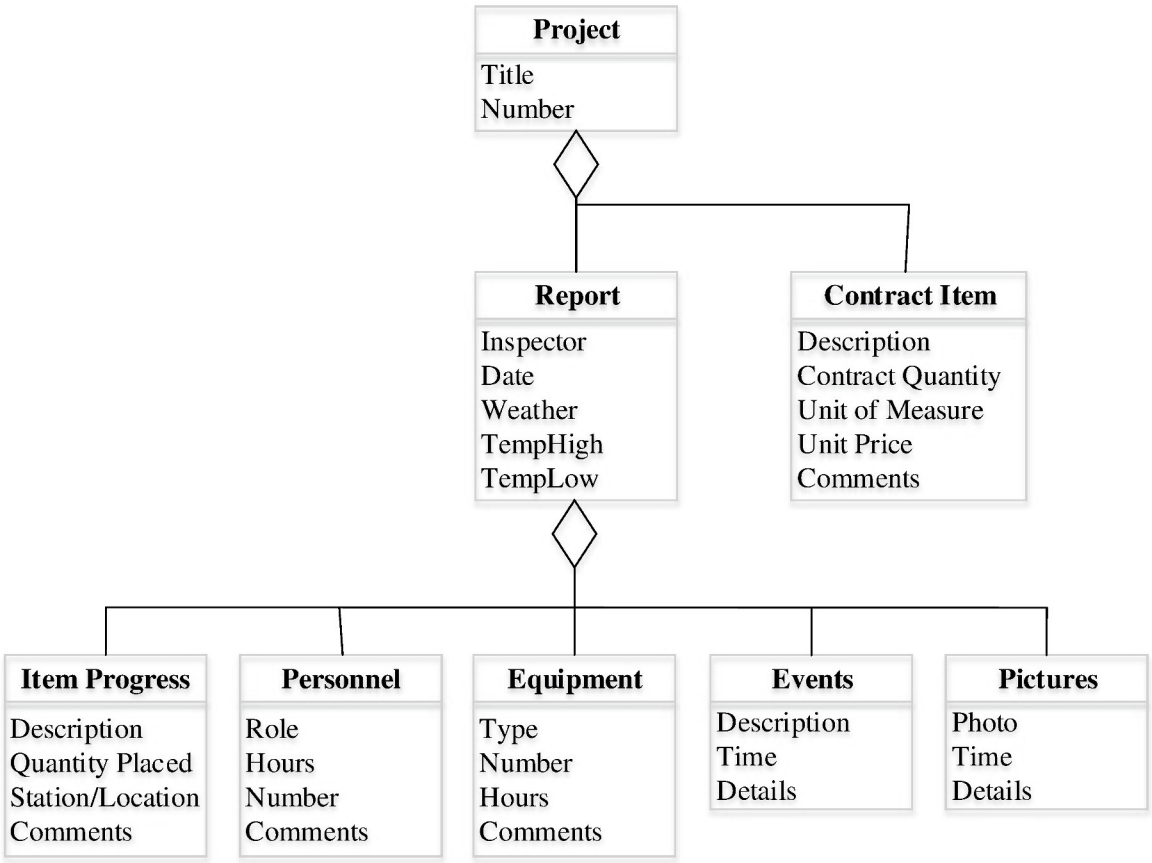
Aplicação móvel – na foto acima, o dispositivo *Android*, é o principal componente do sistema, aquele que interage com o utilizador, recebendo a informação dele proveniente. Ele permite que o inspetor possa gravar relatórios diários para os projetos de construção. Este componente comunica com o servidor de *backend*, enviando e recebendo pedidos *HTTP*, através de uma conexão sem fio. Com o acessório de *hardware*, o dispositivo comunica através de uma conexão *USB*.

Acessório de *hardware* – permite ao inspetor efetuar medições de temperatura e humidade no local. Este acessório necessita de uma conexão *USB* para interagir com o dispositivo através da *interface Android Open Acessory*.

Servidor – contém os dados da aplicação e o mesmo modelo de dados que existe no dispositivo, para que a sincronização entre ambos seja possível. Como a sincronização é feita pela internet, este componente permite que os inspetores acedam aos seus relatórios a partir de qualquer outro dispositivo.

Modelo de dados

Ao analisar o sistema, a equipa extraiu o modelo de dados da aplicação, as entidades presentes na figura seguinte (ver figura 13).



Fonte: Manuel E. Bermudez. A Mobile Field Diary System for Infrastructure Construction Project Inspections, 10th Latin American and Caribbean Conference for Engineering and Technology.

Figura 13 - Modelo de dados da aplicação

A figura anterior apresenta a estrutura dos dados do projeto e do relatório de fiscalização. O projeto é a entidade hierárquica superior, a que contem os relatórios e os itens contratados para um projeto de construção. Um relatório divide-se em várias partes: fotos, eventos, equipamentos, pessoal contratado e progresso dos itens [49].

Project – representa o projeto da construção de uma infraestrutura. Um projeto de construção tem um contrato correspondente que define quais os itens de trabalho necessários ao projeto durante toda a sua implementação, desde materiais de construção, até ao trabalho associado. O inspetor tem que garantir que os itens adquiridos estão no contrato do projeto, isto é conseguido através dos relatórios diários.

Contract Item – são os itens do contrato, material e mão-de-obra necessária à elaboração do projeto de construção.

Report – é o relatório diário produzido pelo inspetor quando examinam aspetos de um projeto**.**

Item Progress – reflete a forma como os materiais são recebidos no local de trabalho, os inspetores necessitam de fiscalizar as quantidades fracionadas de materiais recebidas, para poderem ser comparadas com as quantidades contratadas no projeto.

Personnel – é a forma como os empreiteiros fazem o trabalho no local. Os inspetores devem garantir que os empreiteiros efetuam o trabalho de acordo com o contrato do projeto de construção.

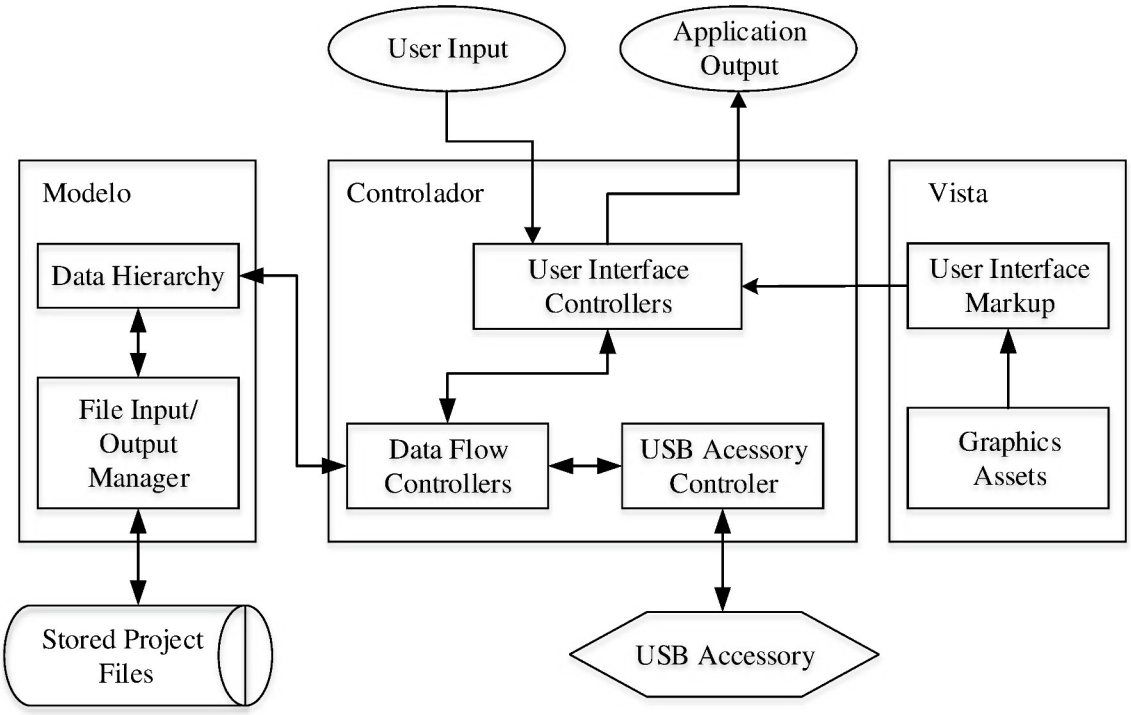
Equipment – todo o equipamento utilizado na obra, como escavadoras e gruas deve ser controlado pelo inspetor.

Events – serve para guardar ocorrências excecionais que surjam numa obra de construção, como: inundações, condição meteorológica adversa, ou acidentes no local de trabalho.

Pictures – os inspetores necessitam de tirar fotos no local da obra e anexa-las aos relatórios. As imagens são geralmente acompanhadas por comentários.

Solução da Arquitetura de Software

A aplicação móvel foi desenvolvida usando o padrão de desenho Model View Controller (MVC) (ver figura 14).



Fonte: Manuel E. Bermudez. A Mobile Field Diary System for Infrastructure Construction Project Inspections, 10th Latin American and Caribbean Conference for Engineering and Technology.

Figura 14 - Arquitetura de Software

A estrutura da aplicação foi dividida em três partes: Modelo, Vista e Controlador [49].

Modelo - trata da manipulação dos dados na BD do dispositivo, necessária à sincronização e funcionamento *offline* da aplicação.

Vista – recebe os dados de entrada na *interface* inseridos pelo utilizador e mostra as alterações efetuadas ao modelo de dados.

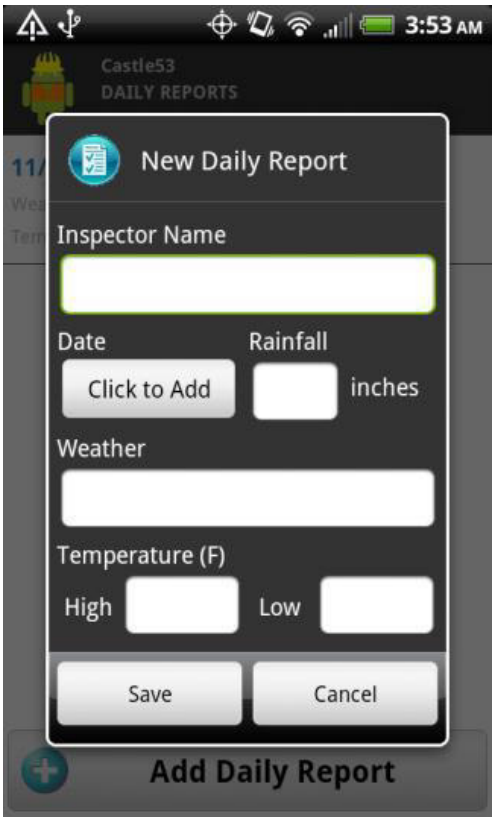
Controlador – redireciona os dados da *interface* para o modelo e apresenta à *interface* as alterações efetuadas no modelo, através do controlador do fluxo de dados e do controlador da *interface*. Além disso também controla o acesso ao acessório *USB*, redirecionado o fluxo de dados proveniente do acessório para o modelo de dados da aplicação.

Interface Gráfica do Utilizador

Esta permite a interação do inspetor com o sistema. Ele realiza todas as funcionalidades que permitem gerir a aplicação móvel de fiscalização, no local da obra. A *interface* é composta por vários tipos de telas, de entrada de informação (ver figura 16), em que o utilizador interage diretamente com o sistema, preenchendo os formulários dos relatórios, ou as telas de saída de informação (ver figura 15), que mostra os dados associados a uma entidade, ou um registo da entidade na BD.

Fonte: Manuel E. Bermudez. A Mobile Field Diary System for Infrastructure Construction Project Inspections, 10th Latin American and Caribbean Conference for Engineering and Technology.

Figura 15 - Detalhe do relatório



Fonte: Manuel E. Bermudez. A Mobile Field Diary System for Infrastructure Construction Project Inspections, 10th Latin American and Caribbean Conference for Engineering and Technology.

Figura 16 - Adicionar novo relatório diário.

Conclusão

Aplicações móveis no terreno permitem que sejam usadas em qualquer momento, fornecendo um método mais conveniente de comunicação entre dados referentes a obras de construção, do que os métodos tradicionais em papel. A inclusão de tecnologia na fiscalização e vistoria de obras permite para além de simplificar os processos de negócio, evitar trabalho redundante, o que por sua vez traduz-se na redução de custos para empresas e organizações.

Neste capítulo abordou-se a contextualização do tema, passando pela descrição de conceitos relacionados com processos de negócio e inclusão de tecnologia móvel em empresas e organizações. No próximo capítulo abordam-se todas as técnicas e métodos utilizados para a conceção da aplicação móvel, desde a fase de levantamento de requisitos até à implementação da aplicação.

3. Metodologia do projeto

Neste capítulo serão abordadas as sete etapas existentes na metodologia de desenvolvimento da aplicação. Na primeira etapa tenta-se compreender o sistema fazendo o levantamento de todos os requisitos funcionais e não funcionais a que deve obedecer. A segunda e terceiras etapas descrevem todas as iterações entre o utilizador e a aplicação, as funcionalidades que a aplicação deve conter à vista do utilizador. Na quarta etapa apresentamos o modelo de dados/objetos da aplicação, ou diagrama de classes. A quinta etapa apresenta as maquetes ou protótipos funcionais da aplicação. A sexta parte apresenta os modelos de arquitetura, na comunicação de dados entre a aplicação e o servidor, a arquitetura geral em termos de tecnologias usadas, a arquitetura do sistema por módulos e a arquitetura da tecnologia *MvvmCross*. Na sétima parte descrevemos as ferramentas e tecnologias usadas para implementação da aplicação. No final do capítulo os pontos-chave do processo de programação da aplicação, tendo em conta a plataforma *Android*, a lógica de negócio e reutilização de código, o funcionamento das *interfaces* e serviços na aplicação e a técnica *DataBinding* do padrão de desenho *M-V-VM* existente na tecnologia *MvvmCross*.

3.1 Levantamento de requisitos

A etapa de levantamento de requisitos de um programa de computação, na área de engenharia de programação, é muito importante para a elaboração de um produto de

qualidade. Esta etapa é a base para a compreensão do sistema a desenvolver, ela pode definir o sucesso/insucesso de um produto, porque clarifica e descreve as condições, capacidades e características qualitativas a que um produto deve obedecer para que seja cumprido um contrato, padrão ou uma especificação entre a empresa e o cliente.

Normalmente, para elaborar o levantamento de requisitos é necessário um trabalho de equipa. Desde reuniões com elementos de desenvolvimento e especialistas na área de fiscalização urbanística da empresa, até à compreensão destes sistemas adaptados à limitação dos dispositivos móveis, é elaborada e validada uma lista de condições e características que devem ser obedecidas. O levantamento de requisitos é um processo mutável, porque a lista de requisitos do sistema vai sofrendo alterações e precisa de afinação durante a fase de desenvolvimento do produto. Para facilitar futuras referências, todos os requisitos descritos foram numerados com um código que o identifica.

3.1.2 Requisitos funcionais

Os requisitos funcionais são condições ou capacidades que o sistema deve ter e obedecer para que as funcionalidades sejam atingidas.

RF1 – O sistema deve permitir que cada utilizador se autentique no dispositivo móvel utilizando as credenciais de autenticação do sistema de suporte de fiscalização eUrban;

RF2 – O sistema deve permitir aos utilizadores a identificação através de *token* de autenticação. Esse *token* é recebido no dispositivo depois de ser cumprido o requisito anterior e contem as credenciais de autenticação do utilizador no sistema. Posteriormente, será usado para que o utilizador não necessite de digitar novamente as credenciais de autenticação;

RF3 – O sistema deve permitir efetuar uma sincronização automática depois do arranque da aplicação, esta sincronização pode ser agendada e pode solicitar ou não a permissão para ser executada;

RF4 – O sistema deve permitir efetuar a sincronização das tarefas marcadas na lista de tarefas do utilizador do eUrban (receber tarefas pendentes do utilizador);

RF5 – O sistema deve permitir efetuar o *upload* dos documentos associados às tarefas marcadas na lista de tarefas do utilizador do eUrban (documentos associados ás tarefas pendentes do utilizador);

RF6 – As tarefas e os documentos marcados para o envio para o dispositivo são selecionados na aplicação onde corre o eUrban;

RF7 – Na *interface* do "posto de trabalho", o eUrban deve bloquear as tarefas que foram marcadas para execução no dispositivo móvel, as tarefas devem estar sinalizadas até que sejam executadas e avançadas no dispositivo móvel;

RF8 – O sistema deve permitir efetuar a sincronização das tarefas que foram realizadas nos dispositivos pelos técnicos (receber tarefas executadas);

RF9 – O sistema deve permitir efetuar a sincronização através da opção sincronizar, para enviar/receber tarefas e documentos do/para o dispositivo;

RF10 – O sistema deve permitir visualizar no dispositivo as tarefas pendentes do utilizador e as tarefas executadas (tarefas sincronizadas no RF4 e RF8);

RF11 – O sistema deve permitir visualizar as tarefas a realizar pelo utilizador, agrupadas por dia, das mais recentes para as mais antigas;

RF12 – O sistema deve permitir ao utilizador escolher por que campo pretende filtrar a lista de tarefas;

RF13 – O sistema deve permitir ao utilizador filtrar a lista de tarefas pelo campo selecionado no requisito anterior. A lista é filtrada pela inserção de texto (pesquisa através de texto) e pelo campo selecionado;

RF14 – O sistema deve permitir visualizar no dispositivo a lista dos documentos que foram associados a cada tarefa no RF6;

RF15 – O sistema deve permitir abrir no dispositivo os documentos que foram associados a cada tarefa no RF6, RF18;

RF16 – O sistema deve permitir que a justificação do insucesso das visitas de vistoria de obras de fiscalização no terreno seja efetuada no dispositivo móvel;

RF17 – O sistema deve permitir completar no dispositivo móvel o preenchimento de documentos que estão anexados à tarefa, é o caso do relatório técnico da tarefa;

RF18 – O sistema deve permitir efetuar o *upload* do relatório técnico da tarefa, depois do documento ser preenchido no dispositivo, entre outros documentos necessários, que ficam anexados à tarefa;

RF19 – O sistema deve avançar as tarefas executadas no dispositivo móvel. Neste caso a tarefa sai da lista de pendentes no móvel e fica numa lista de "pendente de envio" e será realmente avançada no sistema eUrban quando ocorrer a próxima sincronização;

RF20 – O sistema deve permitir à chefia despachar uma tarefa já executada. A chefia ao carregar na tarefa, será apresentado um ficheiro *pdf* com o despacho para ser assinado digitalmente. Assim que este estiver assinado usando um certificado digital, a tarefa avança de forma automática, passando para uma lista de tarefas despachadas pela chefia ou pendentes de envio, à espera de sincronização com o sistema eUrban;

RF21 – O sistema deve permitir configurar se pretende que seja guardada as credenciais do sistema para uma futura utilização pelo utilizador. Em caso afirmativo o sistema guarda o *token* de autenticação e as credenciais de acesso e quando o utilizador pretender efetuar novamente um acesso ao sistema, os campos de utilizador e palavra-chave já estão preenchidos, poupando assim, tempo ao utilizador;

RF22 – O sistema deve permitir configurar opções de sincronização automática: no arranque da aplicação e/ou relativamente aos dias da semana que poderá haver sincronização automática e a que horas desses dias. A sincronização automática pode ser ativada/desativada para ambos os casos;

RF23 – O sistema deve permitir configurar a opção de visualização de tarefas ao utilizador. Listar todas, ou mostrar as tarefas agrupadas por dia;

3.1.3 Requisitos não funcionais

Estes requisitos são as características qualitativas e ou restrições acerca dos serviços ou funções disponibilizadas pelo sistema que complementam os requisitos funcionais. Também são um conjunto de regras que o sistema deve ter para que empresas, clientes e utilizadores fiquem satisfeitos. Os requisitos não funcionais podem ser divididos em várias categorias, apresentadas de seguida.

RNF1 – A arquitetura do sistema deve ser *MVVM*;

Requisitos de apresentação e estilo

Esta seção contém requisitos relacionados ao espírito do produto. Estes requisitos podem em parte ser exigências particulares para o produto que interessam ao cliente, tais como marcas corporativas, cores utilizadas, entre outras:

RNF2 – O sistema deve usar nomes e convenções de modo a que seja percetível para fiscalizadores de obras e trabalhadores da área do urbanismo;

RNF3 – O sistema deve usar uma *interface* gráfica escrita em Língua Portuguesa;

RNF4 – A Interação com o sistema deve seguir os modelos habituais para a plataforma móvel em questão (acesso ao menu de opções, caixas de diálogo quando aplicáveis, listas de itens, ícones alusivos ás funções) de modo a que os utilizadores aprendam com facilidade o modo de funcionamento da aplicação;

RNF5 – A *interface* gráfica deve ter um aspeto atrativo, mas simples, recorrendo aos elementos gráficos habituais nas *interfaces* para a plataforma móvel destino;

RNF6 – A *interface* deve conter um logótipo do eUrban para uma maior identificação como componente do produto, em cabeçalho ou em fundo da aplicação;

RNF7 – A *interface* deve ser adaptável dinamicamente ao tamanho do dispositivo (horizontal e vertical): em termos de fontes de texto; tamanho dos elementos gráficos e arrumação dos mesmos no ecrã;

RNF8 – A *interface* deve seguir as boas práticas de *design* gráfico de modo a garantir a boa usabilidade e legibilidade da informação: um correto contraste entre texto e fundo em termos de cor; não utilizar somente cor para transmitir informação importante; usar também formas diferentes ou ícones - para acomodar a utilização por daltónicos.

Requisitos de usabilidade

Estes requisitos descrevem, em geral, o que deve conter o sistema para a interação do utilizador ser bem-sucedida:

RNF9 – O sistema deverá ter um ecrã de arranque (Inicial / *Login*), onde o utilizador poderá digitar as suas credenciais (nome de utilizador e palavra chave), ou simplesmente submetê-las, caso os campos já estejam preenchidos devido à configuração para guardar as credenciais;

RNF10 – O sistema deverá ter um ecrã principal da aplicação onde serão listadas as tarefas associadas ao utilizador (Lista de Tarefas), explícitas no RF10;

RNF11 – No ecrã da lista de tarefas deverá ser possível alternar entre visualizar as tarefas normalmente, ou agrupar as tarefas por dia, das mais recentes para as mais antigas;

RNF12 – A lista de tarefas terá os seguintes campos: número de processo; nome do processo; número da tarefa; nome da tarefa; número do pedido; nome do pedido e data limite. Deverá ser possível de filtrar por qualquer campo;

RNF13 – No ecrã da lista de tarefas deverá ser possível aceder a um menu de aplicação que contem um item para aceder ao ecrã de sincronização, um item para aceder ao ecrã de configurações da aplicação e outro para permitir ao utilizador fazer *logout* (sair da sessão). O acesso às configurações e outras opções deve seguir a lógica e localização das aplicações da plataforma móvel em questão;

RNF14 – O sistema deverá ter um ecrã de configuração (Configuração). Neste ecrã deverá ser possível configurar as funcionalidades correspondentes ao RF21,RF22 e RF23;

RNF15 – O sistema deverá ter um ecrã de execução de tarefas (Execução). Neste ecrã deverá existir três separadores. Um separador zona que tem os dados "de resumo" (ou da capa) do processo ao qual pertence a tarefa; um separador com a lista de documentos anexos à tarefa; outro com campos a preencher e as questões colocadas acerca da vistoria (Vistoria realizada? Resultado da vistoria?), às quais o utilizador deve responder;

RNF16 – Na zona inferior do ecrã da tarefa deverá aparecer um campo para o *upload* de documentos existentes no dispositivo móvel, e um botão de executar tarefa;

RNF17 – O sistema deverá ter um ecrã de sincronismo (Sincronização). Neste ecrã irá aparecer uma imagem animada, assim como um texto com o nome do processo que se encontra a sincronizar naquele momento. Assim que o sincronismo acabe, será apresentado um quadro resumo com a informação a indicar os processos que foram corretamente ou não sincronizados. Quando um processo falha, o utilizar poderá forçar o sincronismo daquele processo em concreto, e irá conseguir ver em detalhe a mensagem de erro;

RNF18 – No sistema irá existir duas *interfaces* gráficas base: Uma para *tablets* tendo em consideração uma resolução de 7'' e outra para *smartphones* de média / alta dimensão (4" ou mais).

Requisitos de uso confortável

Estes requisitos transmitem os desejos de seus clientes e utilizadores alvos face à complexidade de uso do produto:

RNF19 – O sistema deve ajudar ao utilizador evitar cometer erros;

RNF20 – O sistema deve poder ser utilizado por pessoas com pouca formação informática.

Requisitos de personalização e internacionalização

Estes requisitos descrevem a forma como o produto pode ser alterado ou configurado, para levar em conta as preferências pessoais dos utilizadores ou a escolha do idioma:

RNF21 – O sistema deve manter as preferências de aquisição do cliente e respeitar o idioma principal (Português).

Requisitos de aprendizagem

Enunciam os requisitos que tornam fácil a utilização do produto:

RNF22 – O sistema deve ser fácil para utilizadores alvo aprenderem;

RNF23 – O sistema deve ser usado, testado e validado por pessoas especializadas em testes com conhecimentos na área de urbanismo/fiscalização, antes de ser disponibilizado aos utilizadores alvo.

Requisitos de velocidade e latência

Determinam a quantidade de tempo disponível para completar tarefas específicas. Estes requisitos, frequentemente, referem-se aos tempos de resposta, indisponibilidade ou processamento do sistema face à sua utilização:

RNF24 – Qualquer operação na aplicação deverá ter um tempo máximo de resposta em n segundos. A resposta deve ser rápida suficiente para evitar interrupção no fluxo de pensamento do utilizador;

RNF25 – O sistema deve informar o utilizador da situação da aplicação durante o tempo de indisponibilidade/processamento ou integração.

Requisitos de confiabilidade e disponibilidade

Estes requisitos garantem quando o sistema é confiável e está disponível para ser usado:

RNF26 – O sistema estará disponível assim que instalado e configurado no dispositivo.

Requisitos de robustez ou tolerância a falhas

Estes requisitos especificam a habilidade do produto continuar a funcionar sob circunstâncias anormais:

RNF27 – À exceção das funções de sincronização, o sistema deve permitir funcionamento em modo *offline*;

RNF28 – O sistema deve guardar em memória (base dados do dispositivo) a informação necessária para funcionar em modo *offline*;

RNF29 – O tempo de utilização do sistema está limitado ao tempo de duração da bateria do dispositivo.

RNF30 – O sistema deve guardar operações pendentes assim que um limite de tempo de inatividade for atingido.

Requisitos de capacidade e desempenho

Especifica os volumes de dados que o produto deve ser capaz de administrar, a quantidade de dados transferidos por ele numa ligação, números mensuráveis de parâmetros associados a variáveis do sistema e os tipos de documentos onde pode ser guardada informação:

RNF31 – A dimensão máxima de um processo no sistema pode atingir 25-30 *MBs*;

RNF32 – Numa sincronização podem ser carregados até 8-10 processos (250-300 *MB*);

RNF33 – Um processo tem em média 10 parâmetros;

RNF34 – Um processo tem em média 30 a 40 documentos;

RNF35 – Os documentos podem ser do tipo *PDF*, *CAD*, *JPEG*, *GIF*, *DWG*, *Excel*, *Word*.

Requisitos de ambiente físico esperado

Descrevem em que condições físicas e ambientais o produto pode ser utilizado:

RNF36 – O sistema deve ser construído para que possa ser utilizado em pé, em movimento, em condições de ruido, entre outras condições adversas existentes num local de obras.

Requisitos para as interfaces com os sistemas adjacentes

Estes requisitos descrevem as condições que o sistema precisa e que devem de existir com outros sistemas para ser utilizado com sucesso:

RNF37 – O sistema deve integrar com o sistema de gestão de processos de urbanismo e fiscalização urbanística (eUrban) existente num servidor.

Requisitos de produção

As condições que o sistema deve ter para poder ser produzido e instalado nos dispositivos:

RNF38 – O sistema deve ser distribuído num formato compatível à plataforma destino (*Android*, *iOS*) onde será instalado.

Requisitos de adaptabilidade

Descrevem outras plataformas ou ambientes nos quais o produto precisa funcionar ou se deve ter em conta na sua fase de desenvolvimento:

RNF39 – O sistema deve ser desenvolvido em *C#,* no *Microsoft Visual Studio*;

RNF40 – O sistema é esperado funcionar em *Android* e *iOS*;

RNF41 – O sistema deve ser compatível e escalável para desenvolvimento em *WP*;

RNF42 – O sistema é desenvolvido para funcionar nos locais de obras, mas também pode ser utilizado em escritórios e outros lugares.

Requisitos de acesso

Estes requisitos especificam quem tem acesso autorizado ao sistema, (á informação e funcionalidades) e como é tratada a segurança da informação nas transmissões de dados:

RNF43 – O sistema deve permitir que apenas utilizadores autorizados tenham acesso à informação que apresenta;

RNF44 – O sistema deve possuir um modo de configuração para técnicos e para chefia, deverá existir uma opção no ecrã de configuração que permita o sistema adaptar-se ao tipo de utilizador;

RNF45 – O sistema deve garantir diferentes tipos de segurança conforme o tipo de conexões;

RNF46 - O sistema deve permitir que as sincronizações sejam efetuadas por *wireless* em *Virtual Private Network (VPN)*;

RNF47 – O sistema deve permitir que as sincronizações sejam efetuadas por ligação direta ao computador.

Requisitos de privacidade

Garante que as leis relacionadas à privacidade dos dados dos indivíduos sejam respeitadas:

RNF48 – O sistema deve proteger informações particulares em concordância com leis de privacidade.

Requisitos de documentação

Requisitos referentes à documentação fornecida como parte do produto:

RNF49 – Especificações técnicas que acompanham o produto;

RNF50 – Manual de utilizador e ajuda na aplicação.

3.2 Diagrama de casos de uso

Esta secção descreve as funcionalidades propostas a implementar no sistema. O diagrama de casos de uso contém as ações de iteração entre o utilizador e o sistema. Cada caso de uso, no seguinte diagrama, é identificado no sistema por uma elipse que contém o seu nome dentro. Os utilizadores ou atores, identificados pelos bonecos, são as pessoas que interagem com o sistema obtendo as funcionalidades que lhe permitem executar diversas

tarefas. Desta forma, o utilizador/ator pode efetuar as seguintes funcionalidades principais na aplicação móvel: efetuar *login*; agrupar tarefas; filtrar tarefas; configurar sistema; sincronizar; sair da sessão; escolher tarefa; ver resumo do processo; visualizar documentos; carregar documentos; executar tarefa e despachar tarefa. O perfil do utilizador é composto por duas variantes: técnicos e chefia. Ambos executam em comum a maior parte das funcionalidades, no entanto os técnicos estão mais direcionados para execução de tarefas de vistoria de obras, enquanto a chefia verifica os processos de fiscalização e assina o seu termo de responsabilidade (difere ou indefere) as tarefas executadas pelos técnicos.

Diversos casos de uso podem ter uma sequência no sistema, ou seja, um caso de uso apenas pode ser iniciado assim que outro estiver terminado. Tal como demonstra a figura seguinte, algumas funcionalidades usam ("*uses*") outras funcionalidades. Isto significa, por exemplo que, um utilizador apenas pode configurar o sistema depois de ter efetuado *login* no sistema. Por isso, os casos de uso "Efetuar Login" e "Escolher Tarefa" são essenciais no desenrolar das operações de iteração entre o utilizador e o sistema.

Figura 17 - Diagrama de casos de uso do sistema eUrban Móvel

3.3 Descrição de casos de uso

Os casos de uso correspondem a funcionalidades que o utilizador pode realizar na aplicação com um objetivo específico. Um caso de uso pode ser constituído por várias iterações entre o utilizador e o sistema, até que seja concluído com sucesso, por isso é necessário detalhar cada iteração entre o utilizador e o sistema. Descrever cada caso de uso permite que todas equipas intervenientes no desenvolvimento do sistema tenham a mesma

visão e perspetiva, e garante que os requisitos funcionais do produto estejam de acordo com os objetivos do cliente. Numa descrição de um caso de uso, devem ser enunciados os seguintes aspetos: para que serve, ou qual a sua funcionalidade; quem são os atores (utilizadores do sistema) que participam no caso de uso; o desenrolar das ações que permitem a conclusão com sucesso (fluxo principal); os caminhos alternativos ou exceções que podem ocorrer nas iterações com o sistema (fluxo alternativo). Além disso, podem ainda ser descritas as condições que precisam de existir para que um caso de uso tenha inicio (precondições), as condições que garantem que um caso de uso foi realizado (pós-condições), os requisitos não funcionais que podem estar associados ao caso de uso (Requisitos não funcionais) e restrições impostas pelo negócio que regulam o comportamento de um procedimento ou funcionalidade (Regras de negócio).

Em anexo são apresentadas várias tabelas com a descrição dos casos de uso que compõem o sistema, referido na figura 17.

3.4 Diagrama de classes

O diagrama de classes permite identificar as entidades presentes no sistema e como são guardados os dados para que este seja funcional. Na classe Utilizador são guardados os dados dos utilizadores necessários para aceder o sistema; as classes Configuração, Sincronização e Sessão guardam a configuração do sistema por cada utilizador num dado momento; as restantes classes guardam a informação relativa a todas as tarefas e documentos que são sincronizados e executados no sistema por cada utilizador (ver figura 18).

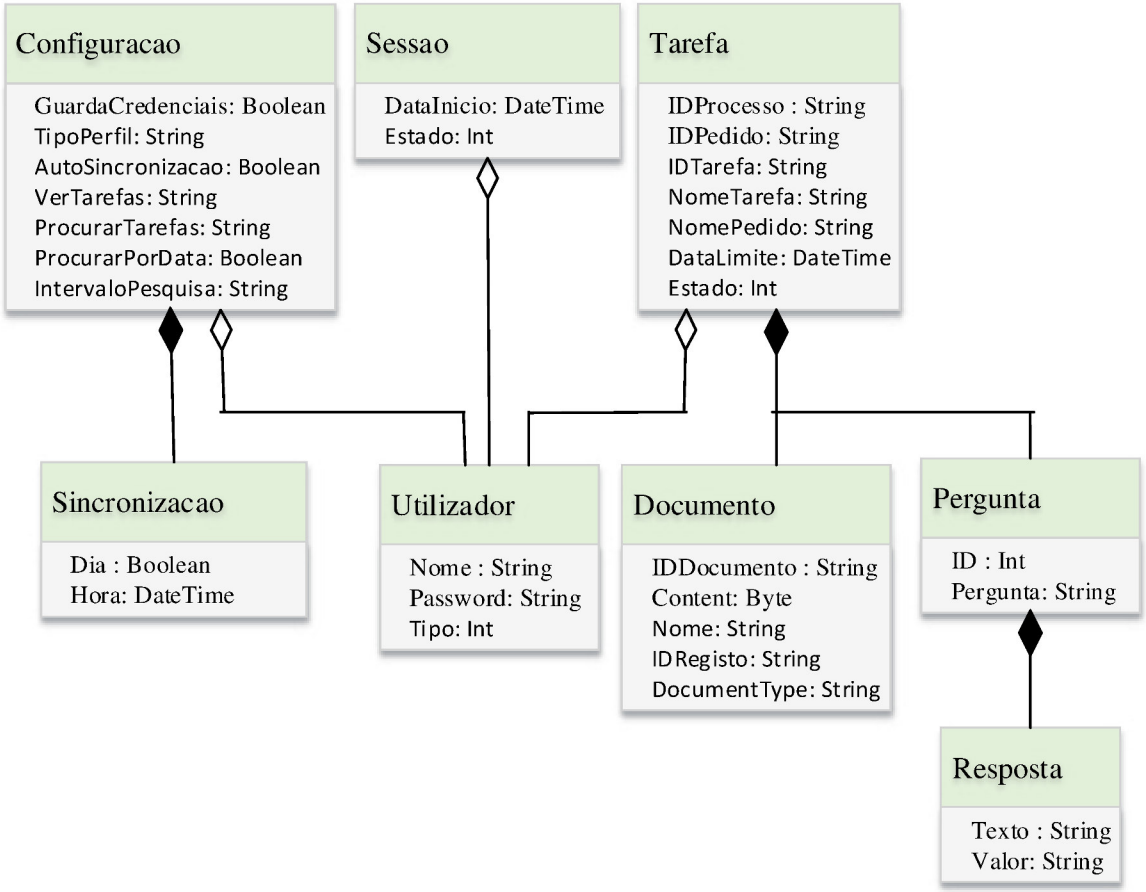


Figura 18 - Diagrama de classes do sistema eUrbanMobile

3.5 Protótipos funcionais

Nesta secção são apresentados os modelos funcionais para os ecrãs da aplicação eUrban Mobile, da versão *Android*. Os protótipos ilustram a disposição dos componentes que permitem efetuar as funcionalidades descritas no diagrama de casos de uso. Eles definem a *interface* gráfica do sistema, o modo de iteração entre o utilizador e o sistema (as entradas e saídas de informação). Os protótipos são essenciais para a fase de desenvolvimento do produto, porque pretendem anular as lacunas existentes entre a perspetiva da equipa de desenvolvimento do produto e a versão que o cliente pretende. Após validados os casos de uso com o cliente, e garantida a consistência dos protótipos com todos os requisitos funcionais e não funcionais, passamos à fase de implementação do sistema.

O sistema irá funcionar em dois tipos de dispositivos: *smartphones* e *tablets*. Como a configuração e resolução das telas são muito diferentes foram criados dois grupos de protótipos adequados a cada configuração (resolução e densidade de pixéis): um grupo de protótipos para a versão *smartphone* (ver figuras 29 - 44); outro para *tablets* (ver figuras 45 - 57).

Todos os protótipos foram validados tendo em conta os requisitos do sistema, com as aptidões e necessidades do cliente.

3.6 Arquitetura

Neste subcapítulo será apresentado o desenho da arquitetura da aplicação móvel. A aplicação eUrban Mobile irá funcionar inicialmente em *smartphones* e *tablets* com sistema operativo *Android*. A arquitetura pode ser definida segundo o modo de comunicação de dados entre os sistemas, ou sobre o ponto de vista de desenvolvimento e tecnologias envolvidas.

3.6.1 Comunicação entre sistemas

A aplicação móvel usa várias formas de comunicação, com ou sem alcance *Wi-Fi* à rede (modo local ou modo remoto). Se a ligação for remota, o sistema deverá utilizar uma ligação *VPN* para comunicar com o servidor. No (s) servidor (es) estão os *webservices* de integração e o sistema de fiscalização urbanística (eUrban), necessários à etapa de sincronização (ver figura 19).

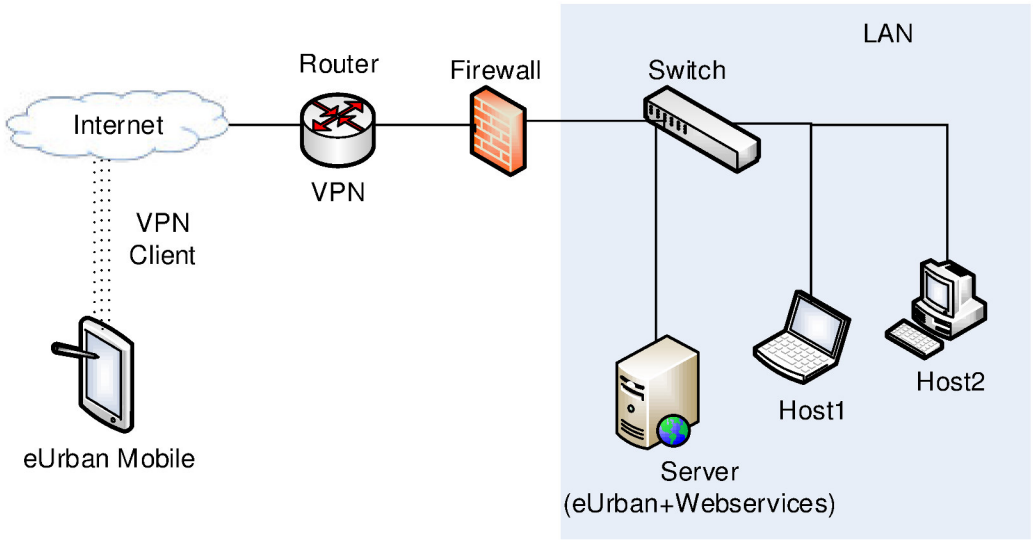


Figura 19 - Integração via VPN+Wi-Fi entre o sistema móvel e fixo

Outra vertente de conexão remota consiste numa ligação por *Wi-Fi* onde os *webservices* são colocados numa *Demilitarized Zone (DMZ)* e o servidor eUrban instalado a parte, dentro da rede interna (ver figura 20).

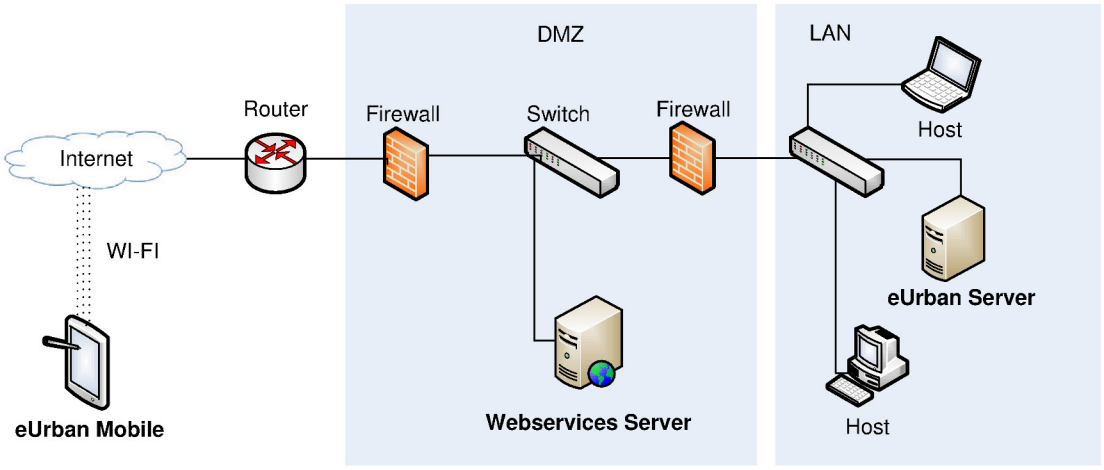


Figura 20 - Integração via Wi-Fi com zona DMZ e LAN interna

Quando a comunicação entre a aplicação e o sistema eUrban é efetuada dentro da rede local, existem duas formas de comunicação: por *Wi-Fi*, nesse caso o dispositivo móvel conecta-se à rede local interna através do sinal *wireless* (ver figura 21), ou por ligação direta entre o dispositivo e um computador ligado à rede interna (ver figura 22).

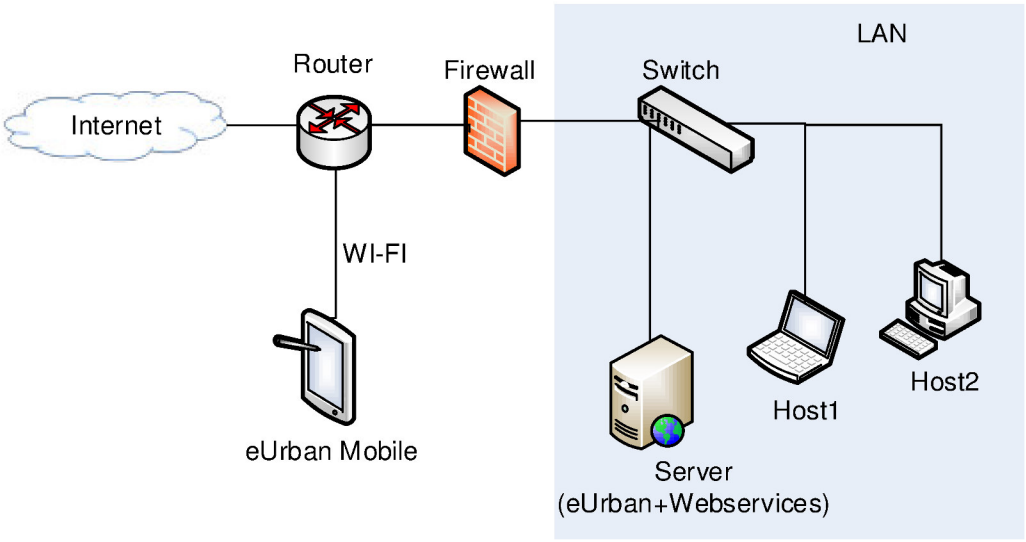


Figura 21 - Integração em rede local por Wi-Fi entre os sistemas

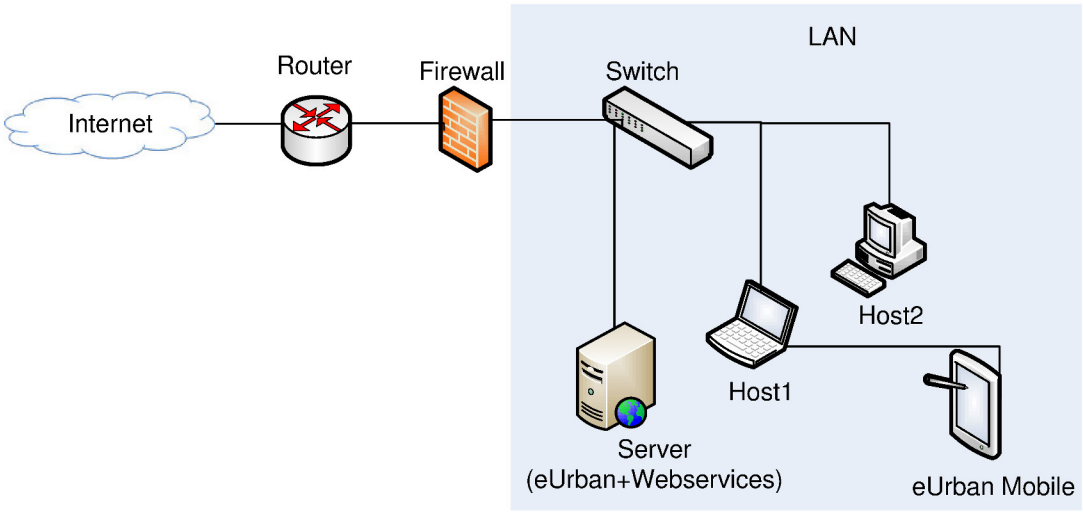


Figura 22 - Integração via ligação de cabo de dados entre os sistemas

3.6.2 Arquitetura Geral

O sistema eUrban Mobile foi desenvolvido usando a plataforma móvel *Xamarin*. Esta plataforma permite, através de um conjunto de elementos, criar aplicações para *iOS*, *Android* e *WP*. O *Xamarin* usa as ferramentas de desenvolvimento (*Android SDK*), a linguagem *C#* do *.NET* e a *API Mono.NET*, no desenvolvimento de aplicações para *Android*. *Xamarin* permite a implementação de aplicações multiplataforma, com a

vantagem do uso de recursos da *Microsoft*, aliado com o comportamento nativo dos componentes *Android*, disponibilizados através do *IDE*.

Do ponto vista geral, o sistema define-se a uma aplicação restrita para uma plataforma (*Android*, *iOS*, *WP*) que comunica com um conjunto de código partilhado genérico. Este por sua vez contém a lógica de negócio do sistema, e uma camada de serviços necessários ao sistema (métodos de sincronização, os serviços de integração da BD, armazenamento de ficheiros,…, etc.). (ver figura 23).

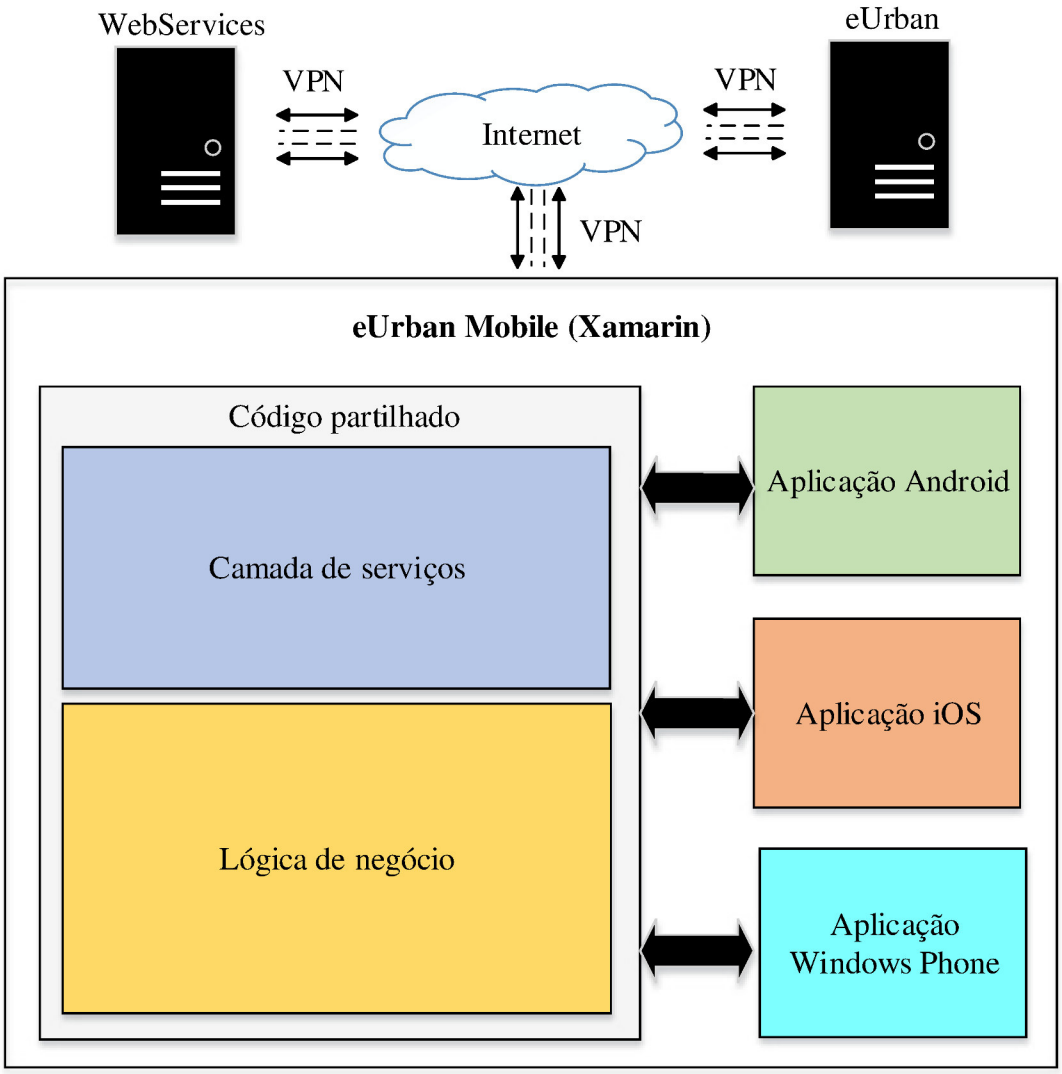


Figura 23 - Arquitetura geral do sistema

3.6.3 Arquitetura do sistema

A arquitetura do sistema a desenvolver, usando *Xamarin*, contém várias camadas com um propósito bem definido. A nível arquitetónico existe uma clara separação de responsabilidades, a utilização de conceitos de encapsulamento e polimorfismo ajudam na modulação, manutenção da aplicação, enquanto permite um desenvolvimento escalável adaptado para outras plataformas.

Cada aplicativo multiplataforma contém dois projetos que interagem e complementam-se entre si (um destinado à plataforma e outro comum a outras plataformas).

O projeto restrito à plataforma divide-se em duas partes: uma **camada de *interface*** que contém o desenho das *interfaces* gráficas para cada "*layout*" do sistema e uma **camada de aplicação** que tem o código próprio da plataforma que não pode ser partilhado.

O projeto com código partilhado no sistema é composto por várias camadas: a ***API Client Webservice*** fornece um conjunto de métodos que interage com os serviços disponibilizados dos *webservices* no servidor e permite a sincronização dos dados no sistema; os **serviços de integração de dados** disponibilizam funções para a utilização dos métodos *Create, Read, Update and Delete* (*CRUD*) ao banco de dados *sqlite*; os **serviços de ficheiros** permitem operações de leitura e escrita na memória física do dispositivo; a **lógica de negócio** contém as regras e serviços de negócio que cumprem os requisitos do sistema e as entidades de negócio (modelo de objetos) que garantem a interoperabilidade e consistência da *BD* com o sistema. (ver figura 24).

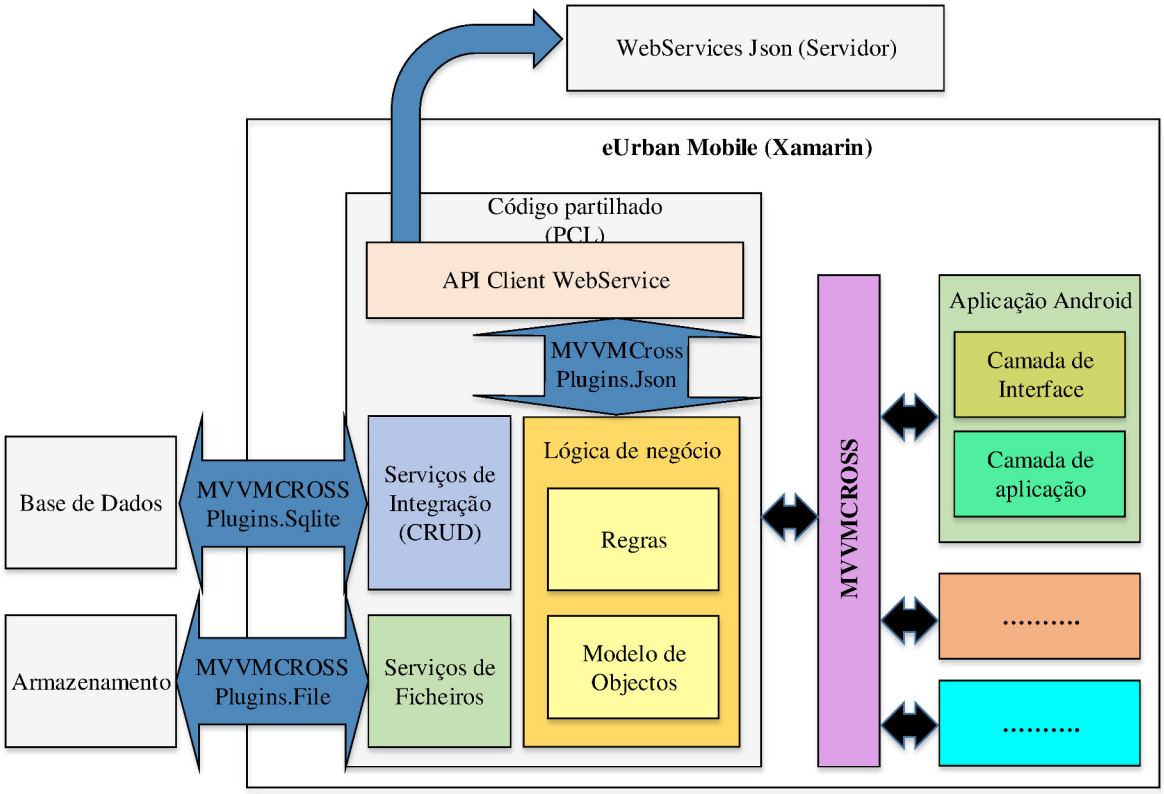
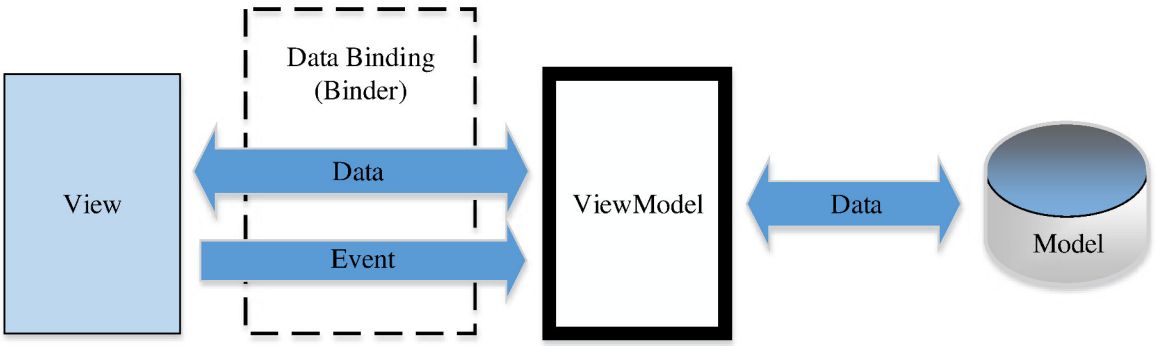


Figura 24 - Arquitetura do sistema

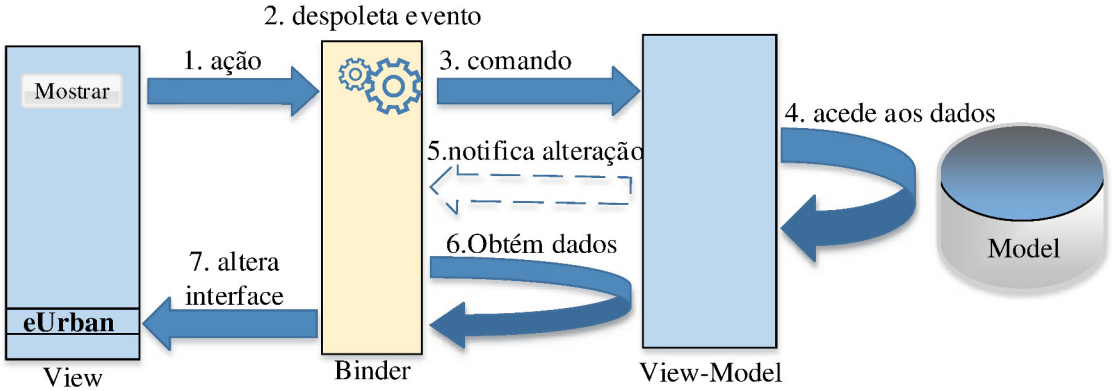
3.6.4 MvvmCross

MvvmCross é uma plataforma que permite desenvolver aplicações para dispositivos com diferentes sistemas operativos usando o padrão *M-V-VM*. Esta plataforma permite criar projetos com código robusto e reutilizável usando ferramentas do *Visual Studio .NET* e das *APIs MonoTouch* e *MonoDroid* do *Xamarin* [50]. Um projeto usando esta arquitetura é composto por pontos de vista (*Views)* que tem ligações às suas representações lógicas (*ViewModels)* através de um mecanismo de conexão (*Binder* ou *Data Binding)*. Sempre que existe uma ação numa *interface* de utilizador é despoletado um evento a partir do *Binder* que chama um comando da sua representação lógica. Este por sua vez acede ao modelo de dados (*Model*), extrai os dados ou informação e notifica o *Binder* que efetua as alterações na *interface* gráfica da *View* (ver figuras 25 e 26).



Fonte: http://www.slideshare.net/fullscreen/Xamarin/mvvmcross-seminar/8

Figura 25 - Padrão M-V-VM do MvvmCross



Fonte: http://www.slideshare.net/fullscreen/Xamarin/mvvmcross-seminar/8

Figura 26 - Padrão M-V-VM detalhado do MvvmCross

O *MvvmCross* é responsável pela iteração entre o projeto *Android* e o projeto que contém o código partilhado. Assim como podemos constatar na figura 24, a *View* e o *Binder* encontram-se na aplicação *Android*, enquanto a *View-Model* e o *Model* pertencem ao bloco de código partilhado. O *MvvmCross* também possui *plugins* para lidar com a persistência de dados no banco *sqlite* (*MvvmCross.Plugins.Sqlite*), efetuar operações de escrita e leitura de ficheiros (*MvvmCross.Plugins.File*) e converter objetos *Json* para tipos reconhecidos pelo sistema (*MvvmCross.Plugins.Json*).

3.7 Ferramentas e tecnologias de desenvolvimento

Na elaboração desta tese e projeto de dissertação utilizamos diversas ferramentas e tecnologias da área computacional:

- ***Project 2010*** – para elaborar a fase de planeamento do projeto;
- ***PowerPoint 2010*** – para criar os slides das apresentações;
- ***Word 2010*** **–** na elaboração deste documento de dissertação e outros ficheiros de rascunho;
- ***Visio 2010/2013*** **–** elaboração do diagrama de casos de uso, diagrama de classes e construção dos esquemas ilustrativos desta dissertação;
- ***Pencil*** **–** para o desenho dos protótipos funcionais do sistema;
- ***Visual Studio .NET 2010*** **–** *IDE* de desenvolvimento do projeto eUrban Mobile;
- ***Xamarin*** **–** para desenvolvimento de aplicações multiplataforma usando a linguagem de programação *C#*. Foi necessário o registo no *site web*, e uma licença de utilização (facultada pela empresa) para que fosse possível testar a aplicação em dispositivos móveis. A instalação do *Xamarin* adiciona ao sistema computacional a *API MonoDroid*, o *Android SDK* e o módulo de integração com o *Visual Studio 2010*;
- ***Portable Library Tools 2*** **–** foi adicionado um suplemento ao *Visual Studio 2010* que permite criar projetos do tipo *Portable Class Library (PCL)*. Este tipo de projetos é a base para construção de aplicações móveis multiplataforma no *Visual Studio 2010*;
- ***MvvmCross*** **–** através da opção *Manage NuGet Packages*, existente no *Visual Studio*, instalou-se o *MvvmCross*. Esta plataforma que tem suporte para projetos *PCL, Android, iOS* e *WP*, possui *plugins* para tratamento do banco de dados *Sqlite* (*MvvmCross.Plugins.Sqlite*), operações sobre

ficheiros (*MvvmCross.Plugins.File*), e manuseamento de objetos *Json* (*MvvmCross.Plugins.Json*);

Após analisarmos toda a documentação, através dos requisitos, desenhos e esquemas do projeto, preparámos o computador e o *IDE* de desenvolvimento com as tecnologias necessárias para a implementação do projeto.

3.8 Implementação do projeto

Esta etapa foi a mais longa do projeto. Aquela onde despendemos mais tempo, pesquisando e aprendendo o funcionamento das tecnologias envolvidas (*Xamarin* para *C#* e *MvvmCross*), adaptando-as às necessidades do eUrban Mobile.

Na construção da aplicação, tivemos em consideração o fator ambiente multiplataforma. Por isso, mais do que as próprias funcionalidades, o essencial foi construir a aplicação dividida em módulos, o mais independentes possível, que permitisse o escalonamento e suporta-se outras plataformas para além do *Android*. Para esse efeito, o mais indicado foi a construção de dois subprojectos:

- ***eUrbanMobile.Droid*** - contém todas as *interfaces* do utilizador, com o ambiente interativo do *Android* e comportamento idêntico às aplicações nativas. Relativamente à tecnologia *MvvmCross,* este projeto é responsável por apresentar as vistas *(Views)* e contém as ligações *(Data Binding)* que fornecem e mantém automatizadas as conexões bidirecionais entre a informação que circula entre cada *View* e *ViewModel* (ver figura 27);

- ***eUrbanMobile.Core*** - contém a lógica do negócio e os módulos que interagem com os webservices do servidor, tratam da persistência dos dados e efetuam operações de escrita e leitura de ficheiros no dispositivo. Quanto ao *MvvmCross*, é neste projeto que estão os modelos das vistas *(ViewModel)* e os modelos dos objetos da aplicação *(Model).* Cada *ViewModel* tem

conexão ao *Data Bindind* do outro projeto e ao modelo de objetos da aplicação, provocando a fusão dos dois projetos (ver figura 27);

3.8.1 eUrbanMobile.Droid

Este projeto é uma aplicação *Android* criada no *Visual Studio 2010* com os recursos das plataformas *Xamarin* (*API MonoDroid*) e *MvvmCross*. A nível estrutural é idêntico a um projeto *Android* em *Java*, à exceção de alguns elementos, de realçar: a pasta *Views,* os ficheiros *Setup.cs* e *SplashScreen.cs* (ver figura 27).

- ***Properties*** – aqui encontra-se o ficheiro de configuração da aplicação (*AndroidManifest.xml*). Foi neste ficheiro que colocámos a versão, nome da aplicação. Também configurámos os tipos de dispositivos com écrans compatíveis (normais, grandes e muito grandes), assim como algumas das permissões que a aplicação necessita do dispositivo para funcionar: aceder à internet e à rede; acesso via *Wi-Fi*; escrever na memória do dispositivo;

- ***References*** – aqui foram adicionadas todas as referências para as bibliotecas que o projeto utiliza;

- ***Resources*** – todos os objetos (ficheiros *xml* e imagens) que as *interfaces* de utilizador carregam estão concentrados dentro desta pasta. Como os écrans dos dispositivos tem resoluções e densidades diferentes as pastas *Drawable-hdpi*, *Drawable-mdpi* e *Drawable-xhdpi* permitem que a aplicação escolha automaticamente, das pastas, ícones e componentes adequados à resolução e densidade do dispositivo alvo. Pelo mesmo motivo existem as pastas *Layout* e *Layout-sw600dp* que contém os écrans de *interface* gráfica para a versão *smartphone* (*Layout*) e versão *tablet* (*Layout-sw600dp*). A pasta *Menu* possui a configuração dos componentes que permitem a navegação entre os écrans (menu de opções) existente na parte superior da aplicação. Na pasta *Values* encontramos as cores usadas na *interface* gráfica, os estilos dos tipos

de letra e de alguns componentes usados, assim como os nomes estáticos dos componentes da *interface* gráfica.

- ***Views*** – acrescentada pelo *MvvmCross*, neste módulo existem os ficheiros responsáveis por carregar as *interfaces* gráficas das pastas *Layout* e *Layout-sw600dp*.

- ***SplashScreen.cs*** – este ficheiro, adicionado pelo *MvvmCross* ao projeto, permite que seja mostrado um ecrã inicial de *Loading,* enquanto o dispositivo se prepara para carregar o ecrã de entrada no sistema (*Login*) do eUrban Mobile;

- ***Setup.cs*** – este foi o ficheiro de configuração inicial adicionado ao projeto pela plataforma *MvvmCross*. Todos os objetos que devem ser inicializados, para uso posterior, serviços e *plugins* que carecem de registo devem ser colocados aqui. Este ficheiro também tem a referência e instância para o projeto *eUrbanMobile.Core*. Assim, o projeto *Android* tem conhecimento da existência do conteúdo do outro projeto, podendo usufruir e usar os métodos e objetos dele.

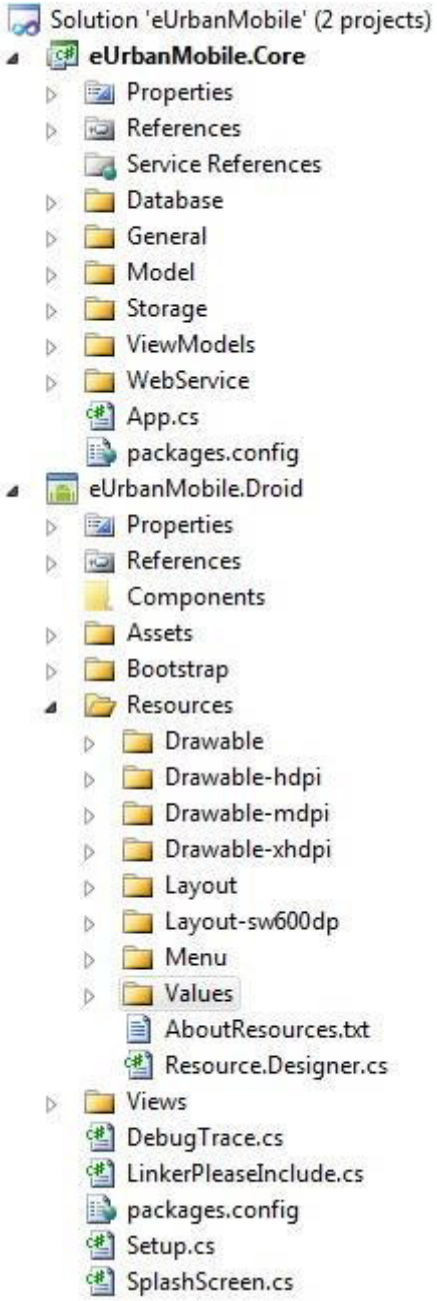


Figura 27 - Estrutura da projeto eUrbanMobile

3.8.2 eUrbanMobile.Core

Neste projeto foi colocado todo o código da aplicação que é independente da plataforma móvel, a lógica de negócio, os métodos dos serviços disponíveis e o modelo de objetos. É um projeto do tipo *Portable Class Library* que suporta desenvolvimento multiplataforma com *.NET Framework 4.5, Silverlight, Windows Phone* [51]. Para o

projeto *PCL* suportar *MonoDroid e MonoTouch* foi necessário criar dois ficheiros *xml*, uma para cada *framework*, na pasta *"C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETPortable\v4.0\Profile\Profile104\SupportedFrameworks"* [52]. Desta forma o nosso projeto *PCL* tem suporte para as *frameworks* alvo (ver figura 28).

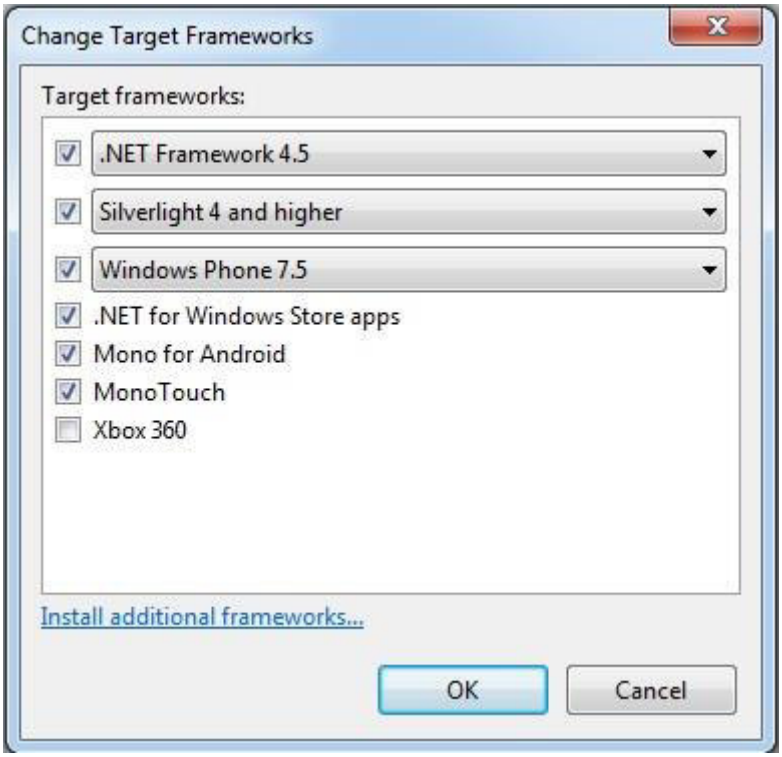


Figura 28 - Frameworks suportadas pelo projeto

O projeto *eUrbanMobile.Core* foi dividido em módulos distintos que facilitam a manutenção da aplicação a médio e longo prazo (ver figura 27).

- ***Properties*** **-** onde podemos alterar as propriedades do projeto e modificar as *frameworks* suportadas;
- ***References*** – aqui foram adicionadas todas as referências para as bibliotecas que o projeto utiliza;
- ***Database*** – o módulo da BD que contém os serviços e as classes de serviço que tratam da persistência dos dados (métodos *CRUD* à *BD*);

- ***General*** – contém as funções e métodos genéricos úteis que não dependem de nenhum módulo;
- ***Model*** – neste módulo estão todas as classes de objetos e tipos de dados dependentes que existem na aplicação (modelo de objetos);
- ***Storage*** – módulo que contém o serviço e classe de serviço das operações de leitura e escrita na memória do dispositivo;
- ***WebService*** – módulo que contém o serviço de comunicação com os *Webservices Json* no servidor (sincronização de dados entre os sistemas).
- ***ViewModels*** – este módulo acrescentado pela plataforma *MvvmCross* contém todos os modelos lógicos dos pontos de vista (*Views*) utilizados pelo padrão *M-V-VM* da aplicação;
- ***App.cs*** – ficheiro de inicialização que permite configurar qual o ecrã de arranque da aplicação;

3.8.3 Registar e usar serviços

Consiste em criar e registar as referências das *interfaces* ou serviços da aplicação. Por exemplo, foram registados alguns serviços na classe *Setup.cs*:

```
1. Mvx.RegisterSingleton(typeof(IFileStore), new FileStore(dbPath));

2. Mvx.RegisterSingleton(typeof(IWebServiceClient),
new WebServiceClient("http://192.168.1.149/eUrbanWebservice.asmx/"));

3. Mvx.RegisterSingleton(typeof(IMvxTaskDataService<Task>),
          new MvxTaskDataService(connection));
```

No terceiro ponto estamos a registar a *interface* `IMvxTaskDataService<Task>` que contem os métodos que permitem lidar com a persistência de dados da tabela *Task* (tabela que guarda os dados das tarefas da aplicação). De seguida podemos ver os métodos públicos que a *interface* implementa.

```
    public interface IMvxTaskDataService<Task>
    {
        Boolean CreateTable();
        Boolean DropTable();
        Boolean Insert(Task objeto);
        Boolean InsertAll(IEnumerable<Task> objeto, Boolean value);
        Boolean Update(Task objeto);
        Boolean Delete(Task objeto);
        void CloseConnection();

        IEnumerable<Task> GetAllTasks();
        Task GetTaskById(string idPedido, string idTarefa);
        IEnumerable<Task> GetALLTasksByState(string user, int taskState);
        IEnumerable<Task> GetTasksByState(string user, int taskState);
        IEnumerable<Task> GetTasksBySearch(string user, int taskState, string
searchType, string search);
        IEnumerable<Task> GetSearchTasks(IEnumerable<Task> query);
        int GetCountItems();
    }
```

Para registar esta *interface* precisámos de instanciar a classe do serviço que recebe como parâmetro a conexão à BD e implementa cada um dos métodos da *interface*;

Se pretendemos inserir os dados de uma tarefa na BD, supondo que a tabela *Task* se encontra criada, devemos então chamar a referência do serviço e o método de inserção:

```
IMvxTaskDataService<Task> _taskDataService =
Mvx.Resolve<IMvxTaskDataService<Task>> ();

_ taskDataService.Insert(item);
```

A mesma filosofia de registo/uso é seguida para os outros serviços da aplicação.

3.8.4 DataBinding

DataBinding é a peça mais importante da tecnologia *M-V-VM*, porque permite vincular cada ponto de vista de um projeto (*View*) ao seu modelo lógico/representação (*ViewModel*) existente em outro projeto [53]. Este tipo de tecnologia necessita obrigatoriamente que as propriedades existentes em um ponto de vista estejam presentes no seu modelo lógico. Qualquer componente de um ponto de vista que possua uma iteração

com o utilizador deverá ter a sua representação lógica no modelo da vista, através de propriedades e comandos se assim for necessário.

Consideremos a funcionalidade de visualizar a informação de uma tarefa escolhida pelo técnico da lista de tarefas da aplicação:

1. O componente gráfico lista (`Mvx.MvxListView`) existente no *Layout* da *interface* gráfica respetiva necessitou de uma ligação para o tipo de dados de destino, onde são guardadas as tarefas a mostrar na View (**`ItemsSource ListTasks`**). A ação de pressionar um item da lista corresponde ao comando que permite invocar o método para visualizar a informação da tarefa escolhida (**`ItemClick ShowTaskCommand`**).

```
<Mvx.MvxListView
  p1:minWidth="25px"
  p1:minHeight="25px"
  p1:layout_width="fill_parent"
  p1:layout_height="fill_parent"
  p1:divider="@color/listdivider"
  p1:dividerHeight="1dp"
  p1:id="@+id/LV_Tasks"
  local:MvxBind="ItemsSource ListTasks; ItemClick ShowTaskCommand"
  local:MvxItemTemplate="@layout/task_item"
 />
```

2. No modelo da *View* (TasksViewModel) foi necessário adicionar o tipo de dados referido anteriormente (**`ListTasks`**), que ficou conectado ao componente anterior (Mvx.MvxListView). Acrescentou-se também o comando (**`ShowTaskCommand`**) que passa por referencia a tarefa selecionada ao método **`DoSelectTask()`**. Este método é responsável por mostrar a informação detalhe da tarefa, porque é invocada outra *ViewModel* (**`ShowViewModel<TaskDetailViewModel>(item);`**), que por sua vez está ligada à *View* que contém a informação requerida.

```
private IEnumerable<Task> _listTasks;
public IEnumerable<Task> ListTasks
{
      get { return _listTasks; }
       set { _listTasks = value; RaisePropertyChanged(() =>
ListTasks); }
}


private MvxCommand<Task> _itemSelectedCommand;
public ICommand ShowTaskCommand
{
      get
      {
            _itemSelectedCommand = _itemSelectedCommand ?? new
            MvxCommand<Task>(DoSelectTask);
            return _itemSelectedCommand;
```

```
        }
}

private void DoSelectTask(Task item)
{
        ShowViewModel<TaskDetailViewModel>(item);
}
```

Neste capítulo foram abordadas as etapas para conceção da aplicação: levantamento dos requisitos do sistema, as funcionalidades existentes na aplicação, os modelos de objetos necessários à BD, os protótipos funcionais, as arquiteturas de comunicação de dados entre o sistema e o servidor, e do próprio sistema sobre vários pontos de vista, as ferramentas e tecnologias usadas nesta tese e como foi implementada a aplicação em termos de programação. No próximo capítulo aborda-se a discussão de resultados tendo em conta os testes efetuados e o estudo de caso investigado.

4. Discussão de Resultados

Neste capítulo serão apresentadas justificações relativas à metodologia e escolhas efetuadas, limitações e dificuldades sentidas na implementação da aplicação, comparação entre o desempenho da aplicação multiplataforma versus plataforma única e a perspetiva de implementação para a plataforma *WP*.

4.1 Metodologia escolhida

Na elaboração deste projeto deu-se enfase aos desafios do projeto. Criar uma plataforma base que sustente uma aplicação compatível / reutilizável em vários sistemas operativos móveis tornou-se um grande desafio. Por um lado devido à quantidade de dispositivos com características muito diversificadas e por outro, as tecnologias existentes que tivemos que analisar para resolver o problema. Para garantir níveis de usabilidade entre a plataforma *Android, iOS* e *Windows Phone* investigamos as soluções existentes para desenvolvimento multiplataforma usando uma única linguagem de programação. Para além, o ideal, conseguir que o desenvolvimento de aplicações futuras para essas plataformas alvo partilhasse uma mesma lógica de negócio, serviços e dados, mantendo o aspeto gráfico e iteração do utilizador própria de cada plataforma.

Foram analisados tipos de aplicações distintas que resolvessem o problema, conforme os seus objetivos, vantagens e desvantagens do seu uso, a aplicabilidade destino e as ferramentas e tecnologias necessárias para implementação de cada tipo de aplicação. As aplicações nativas muito boas sobre o ponto de vista de desempenho foram excluídas

porque não funcionam num ambiente multiplataforma, são construídas usando a linguagem e componentes específicos para cada plataforma. Já as aplicações *web* portáteis são utilizadas usando linguagem *Web* (*HTML5,CSS* e *Java*), mas necessitam de uma ligação quase constante à internet, o que dificulta o uso do utilizador, já que se pretende que a aplicação seja destinada a técnicos que vão fazer a vistoria de obras em locais onde porventura não tem ligação *WiFi*.

A escolha final foi entre as aplicações hibridas multiplataforma e aplicações multiplataforma compiladas. As aplicações hibridas (meio Web, meio Nativas) são implementadas usando a linguagem web (*HTML5, CSS* e *Javascript*), o seu corpo de código é exportado e executado dentro do dispositivo pelos navegadores *web*, juntamente com *APIs* que acedem às funcionalidades dos dispositivos. As aplicações multiplataforma compiladas usam geralmente *C#* ou *C++* como linguagem de programação. Estas utilizam *APIs* nativas que garantem o aspeto nativo da *interface* gráfica de cada plataforma. Optamos por escolher o desenvolvimento de uma aplicação multiplataforma usando *C#* por várias razões: primeiro, porque estávamos mais familiarizados com a linguagem *C#* em vez da linguagem *web*. Além disso programar usando linguagem *web* necessita de uma curva de aprendizagem maior, uma vez que é composta por três linguagens (*HTML5, CSS* e *Javascript*). O acesso ao banco de dados também afetou a nossa escolha, o *SQLite* é mais fácil de usar com objetos *XAML*, acessíveis via *C#*, do que objetos *HTML*, usando linguagem *web*.

Quanto ao padrão de arquitetura da aplicação foi utilizado o *M-V-VM* embutido na *API MvvmCross*. Após investigação e alguns testes efetuados, optamos por usar *MvvmCross* como tecnologia de reutilização/partilha de código. A empresa *Link* patrocinadora/parceira do projeto já tinha avançado com a compra da licença *Xamarin*, portanto, facilitou a nossa escolha para este tipo de tecnologia. Após testarem o desempenho de aplicações desenvolvidas com *MvmCross* e *Xamarin*, passou a usá-la nos seus projetos móveis multiplataforma devido à sua eficiência e fiabilidade.

4.2 Limitações e dificuldades sentidas

No desenvolvimento da aplicação multiplataforma usando *C#* com *Xamarin* e *MvvmCross* foram sentidas algumas dificuldades no uso das tecnologias para implementação do projeto.

A *API MvvmCross* é uma tecnologia recente com uma comunidade pouco extensa. Durante a implementação do projeto foi difícil encontrar informação *online* de suporte sobre o funcionamento do *MvvmCross*, o que dificulta a tarefa para desenvolvedores pouco experientes que pretendem criar aplicações usando esta *API*.

Com a tecnologia *MvvmCross*, o preenchimento de dados nos componentes nativos (*ListViews*, *TextBoxs*, *Spinners*,…etc) e os eventos despoletados através deles usa a programação baseada em conexões entre a vista e o modelo de exibição (*DataBinding*), invés de controladores (*Adapters*) usados nas aplicações nativas para *Android*. Na realidade este facto pode simplificar ou dificultar o desenvolvimento de aplicações multiplataforma. Por exemplo, se não existe uma adaptação do componente *ExpandableListView* (Lista Agrupada) para a tecnologia *MvvmCross*, não pode ser utilizada a sua funcionalidade (expandir/agrupar), através do método de conexão entre a vista e o modelo de exibição.

4.3 Comparação dos resultados obtidos

Com o objetivo de verificar o desempenho da aplicação multiplataforma criou-se outra versão da aplicação no *Visual Studio 2010*, sem reutilização de código e compatível apenas com a plataforma *Android*. Esta versão não utiliza tecnologia *MvvmCross*, a *interface* gráfica da aplicação, a lógica de negócio, os serviços e modelo de dados estão contidos todos no mesmo projeto.

Para ambas as versões foi medido o tempo de exibição do ecrã de sincronização de tarefas, que contem uma lista de dados com 50 itens. Foram usados os mesmos componentes em ambas as versões, os dados a apresentar também são idênticos. Este teste

foi efetuado num emulador *Android*, pelo que os valores apresentados podem diferir consoante o dispositivo de teste. Na versão multiplataforma, as listas e os seus itens usam o método de conexão (*DataBinding*) entre a vista (*View*) e o modelo de exibição (*ViewModel*). Na versão de plataforma única, a visualização dos itens da lista são tratados através de controladores (*Adapters*). Verificamos que a aplicação multiplataforma demora cerca de um segundo e 115 milissegundos a exigir o ecrã de sincronização (ver figura 61 em anexo), enquanto a aplicação que não utiliza *MvvmCross* demora um segundo e 193 milissegundos (ver figura 60 em anexo). Isto significa que a utilização de tecnologia *MvvmCross* usando *DataBinding* não aumentou o tempo de exibição de um ecrã relativamente a outra aplicação que usou controladores de exibição para o mesmo efeito, ou seja, não afeta o desempenho da aplicação.

Para verificar qual o tipo de operações que carecem de mais tempo de processamento efetuou-se um teste usando um dispositivo *Android* de gama baixa, o *Samsung Galaxy Young*[12]. Nos testes efetuados mediu-se o tempo de visualização de dados num écran, ou tempo de execução de operações, conforme o número de tarefas envolvidas. Por cada tarefa inserida está associada uma pergunta, uma resposta e um documento. Portanto, o número de inserções em quatro tabelas e criação de documentos na memória física do dispositivo é proporcional ao número de tarefas. Os valores temporais retirados (ver figura 61) são uma média de cinco ocorrências, pelo que os resultados estão sempre dependentes da capacidade de processamento do dispositivo e da percentagem de processador utilizada no momento. Segundo os valores do gráfico da figura 62, verificamos que as operações de visualização de dados carecem de pouco processamento, menos de meio segundo. Aliás, nos écrans de execução/sincronização de tarefas, o tempo de carregamento, entre mostrar um lista com uma ou 200 tarefas, mantém-se constante. Os écrans de configurações e manuseamento de filtros não estão relacionados com o número de tarefas, pois só verificam e mostram as configurações do utilizador no sistema. Já no teste de inserção de dados e criação de documentos, o desempenho é diferente (ver figura 63). O tempo de execução das operações é proporcional ao número de tarefas envolvidas. São necessários mais de 27 segundos para o dispositivo efetuar 200 inserções em cada uma

[12] Especificações Samsung Galaxy Young - http://www.samsung.com/pt/consumer/mobile-phone/smartphones/android/GT-S6310DBATPH

das quatro tabelas e criar 200 ficheiros de texto na memória física. Tendo em conta os resultados, podemos afirmar que o desempenho da aplicação não é afetado pela quantidade de elementos carregados num ecrã, mas pelo número de operações e verificações à BD e/ou memória física do dispositivo.

4.4 Implementação para outras plataformas

Por forma a validar a arquitetura do sistema e o trabalho efetuado em torno da metodologia escolhida foi implementado um projeto de teste para a plataforma *WP*. O projeto criado contém apenas dois écrans: a entrada no sistema e as configurações (ver figura 59 e 60). A implementação das funcionalidades dos dois módulos para a plataforma *WP7* demorou cerca de cinco horas, muitos menos do que o tempo despendido na criação das mesmas funcionalidades para a plataforma *Android*. Na realidade, isto aconteceu porque toda a lógica de negócio, serviços da aplicação e modelo de objetos já estão implementados no projeto *eUrbanMobile.Core*, e foram apenas reutilizados e invocados. Para criar uma nova implementação do sistema, necessitou-se de criar um novo projeto de *interface* da aplicação. Este projeto contém as vistas (*Views*) com os componentes nativos da plataforma *WP7*. Os componentes são conectados às propriedades dos modelos de vista (*ViewModels*) do projeto *eUrbanMobile.Core*, descrito no ponto 3.6.4 e ilustrado pela figura 29. Verificou-se que com as tecnologias escolhidas, conseguem-se obter *interfaces* de utilizador, de diferentes plataformas móveis, conectadas ao mesmo modelo lógico do sistema.

Neste capítulo abordou-se a escolha da metodologia, as limitações e dificuldades sentidas, comparou-se o desempenho da aplicação e discutiu-se a implementação da aplicação para outra plataforma. No próximo capítulo apresenta-se as conclusões da tese e as referências a trabalho futuro.

Esta página foi intencionalmente deixada em branco

5. Conclusões e trabalho futuro

Neste capítulo apresentam-se conclusões sobre a investigação e trabalho efetuado, no final abordam-se as linhas e perspetivas de implementação futuras.

5.1 Conclusões

A escolha das tecnologias de arquitetura de um sistema é um dos pontos mais importantes para elaboração de uma boa aplicação móvel. Criar aplicações móveis utilizando as tecnologias *Xamarin* e *MvvmCross* é desafiante para quem está habituado a desenvolver aplicações para uma única plataforma. No entanto, ultrapassadas as barreiras iniciais da sintaxe linguística e a dificuldade de encontrar documentação e suporte às *APIs*, estas tecnologias integradas com o *Visual Studio 2010* para desenvolvimento de soluções multiplataforma tem um comportamento muito bom fase ao desempenho das aplicações nativas. Além disso, como estão estruturadas em módulos, facilita o desenvolvimento em equipa caso seja utilizado um controle de versões no manuseamento do projeto. Como estas tecnologias possuem a vantagem de desenvolvimento multiplataforma (*Android*, *iOS* e *WP)* usando uma única só linguagem, o tempo dispensado na fase de aprendizagem e adaptação é compensado. A empresa não tem que recrutar pessoas especializadas em três linguagens e em três plataformas, os desenvolvedores apenas tem de conhecer os serviços que as *APIs* de desenvolvimento multiplataforma disponibilizam e saber estruturar *interfaces* gráficas para cada plataforma.

Segundo os resultados dos testes de desempenho, o número de ligações entre a vista e o modelo da vista, ou seja, a quantidade de elementos visíveis no ecrã a serem carregados, não afeta diretamente a duração do tempo, entre o instante em que o ecrã é criado até à sua visualização. O desempenho é afetado de forma considerável, quando inserimos na BD ou criamos documentos na memória física do dispositivo.

Conclui-se que um sistema otimizado para acesso móvel no terreno permite vários fatores positivos: a flexibilidade de comunicação e documentação porque é independente do tempo e do espaço; aumento da qualidade do produto, evitando a perda de informações e facilita o controlo e manuseamento do projeto; o tempo de espera entre as várias etapas dos processos é minimizado, assim como o tempo de iteração entre as partes envolvidas, o que eleva os padrões de qualidade, aceitação e melhores resultados nas revisões dos projetos; os novos processos que sejam necessários adicionar tem menor riscos e custos de implementação.

5.2 Trabalho futuro

Uma vez que o trabalho foi centralizado no funcionamento da plataforma base para reutilização e partilha de código, ficaram algumas funcionalidades por implementar que enriquecem a aplicação: validar a autenticação do utilizador através das credenciais que existem no servidor; os técnicos poderem agrupar as tarefas a realizar por data limite de execução; criar o módulo de chefia onde estes podem efetuar o despacho das tarefas através dos certificados digitais; abrir os documentos das tarefas no dispositivo ou através da aplicação; completar o módulo de vistoria das tarefas no terreno através de informação mais concreta. Os Testes de sincronização com *webservices* reais que inserem na BD do servidor não foram validados. Para além destas linhas de trabalho futuro, ainda podemos aproveitar o módulo multiplataforma e desenvolver implementações da aplicação para diferentes sistemas (*iOS* e *WP*).

Bibliografia

[1] **Davenport,** T. (1993). *Process innovation: Re-engineering work through information technology*. Cambridge, MA, Harvard Business School Press.

[2] **Hammer**, M. and **Champy** J. (1993). *Re-engineering the corporation: A manifesto for business revolution*. New York, Harper Business.

[3] **Willoch** B.E. 1994. *Business Process Re-engineering*, Docendo Läromedel AB.

[4] **Rummler** G. A. and **Brache** A. P. 1995. *Improving Performance : How to Manage the Whitespace on the Organization Chart,* Second edition, Jossey-Bass Publichers.

[5] **Ould**, M, (2005). *Business Process Management: Uma abordagem rigorosa.* ISBN 1-902505-60-3

[6] **Nilsson**, K., **Isaksson**, L., **Levander**, G., **Olofsson**, T. (2003). Field Force Automation in Construction, *Technical report 2003:10*, Luleå University of Technology, 66 pages, (in Swedish).

[7] **Varshney**, U. (2003). *Mobile and Wireless Information Systems: Applications, Networks, and Research Problems. Communications of AIS*, 2003(12), 155-166.

[8] **Perry**, M., **O'Hara**, K., **Sellen**, A. **Harper**, R. & **Brown**, B.A.T. (2001). *Dealing with mobility: understanding access anytime, anywhere. ACM Transactions on Computer-Human Interaction,* 4(8), 1-25.

[9] **De la Garza**, J.M. & **Howitt**, I. (1998). *Wireless communication and computing at the construction jobsite*: *Automation in Construction* 7(4), 327-347.

[10] **Eisenblatter**, K. (2001). *Investigation and Prototype Development for a Personal Digital Assistant for Document Access from Construction Sites. Research Project Report, Department of Civil Engineering and Environmental Engineering*, Carnegie Mellon University.

[11] **Kimoto**, K, **Endo**, K, **Iwashita**, S & **Fujiwara**, M. (2005). *The application of PDA as mobile computing system on construction management. Automation in Construction*, 14(4), 500-511.

[12] **Loefgren**, A. (2006*),* ICT investment evaluation and mobile computing business support for construction site operations. *Proceedings of the 5th annual Mobility Roundtable Conference*.

[13] Gartner Says 821 Million Smart Devices Will Be Purchased Worldwide in 2012; Sales to Rise to 1.2 Billion in 2013. [consulta em 2012-12-11]. Disponível em: http://www.gartner.com/it/page.jsp?id=2227215.

[14] **Nagesh**, A.; **Caicedo,** C. (2012). Cross-Platform Mobile Application Development, ITERA 2012, The 10th Annual Conference on Telecommunications and Information Technology, 30thMarch-1 April 2012, Indianapolis. Indian. [consulta em 2012-12-20]. Disponível em: http://www.academia.edu/4233075/Cross-Platform_Mobile_Application_Development

[15] **Heitkötter**, H.; **Hanschke**, S.; **Majchrzak**, T. (2012). Comparing cross-platform development approaches for mobile applications, WEBIST 2012 - *8th International Conference on Web Information Systems and Technologies*. [consulta em 2012-12-20]. Disponível em: http://www.wi.uni-muenster.de/pi/veroeff/heitkoetter/Comparing-Cross-Platform-Development-Approaches-for-Mobile-Applications.pdf

[16] MobiWebApp Web site. [consulta em 2012-12-13].Disponível em: http://www.mobiwebapp.eu/.

[17] **Forgue**, M.C.; **Hazaël-Massieux**, D. - Mobile Web Applications: Bringing Mobile Apps and Web Together. W3C/ERCIM, *Sophia Antipolis*. Lyon, France. [consulta em 2012-12-12]. Disponível em: http://www2012.wwwconference.org/proceedings/companion/p255.pdf

[18] Objectives | MobiWebApp. [consulta em 2012-12-13]. Disponível em: http://mobiwebapp.eu/objectives/.

[19] W3C Device APIs Working Group. [consulta em 2012-12-13]. Disponível em: http://www.w3.org/2009/dap/.

[20] MobiWebApp's standardization roadmap (Fifth edition as of 20 February 2012). [consulta em 2012-12-13]. Disponível em: http://www.w3.org/2012/02/mobile-webapp-state/.

[21] "Mobile Web 1: Best Practices" training course. [consulta em 2012-12-13]. Disponível em: http://www.w3devcampus.com/mobile-web-and-applicationbest-practices-training/.

[22] "Mobile Web 2: Applications" training course. [consulta em 2012-12-13]. Disponível em: http://www.w3devcampus.com/writing-great-webapplications-for-mobile/.

[23] MobiWebApp outreach. [consulta em 2012-12-13]. Disponível em: http://mobiwebapp.eu/news-events/; http://mobiwebapp.eu/press-room/; http://mobiwebapp.eu/press-clips/.

[24] **Chipperfield**, R. (2012). An introduction to cross-platform mobile development technologies. [consulta em 2012-12-21]. Disponível em: http://www.codeproject.com/Articles/388811/An-introduction-to-cross-platform-mobile-developme

[25] Store – Xamarin. [consulta em 2013-04-02]. Disponível em: https://store.xamarin.com/

[26] **Longhorn**, A.; White Paper—Native vs Cross Platform Development.Nov. 2005. [consulta em 2012-12-13]. Disponível em: www.istrategyconference.com/app/media/whitepapers/iApps-Native-vs-Cross-Platform-Development-Nov-12.pdf.

[27] Facebook abandons HTML5 on iOS: A shift or exception?. [consulta em 2012-12-10]. Disponível em:http://www.infoworld.com/t/html5/facebook-abandons-html5-ios-shift-or-exception-201604

[28] Zuckerberg admite que as versões HTML5 da aplicação Facebook foram um erro. [consulta em 2012-12-10]. Disponível em: http://www.pcdebolso.com/notVer.asp?ID=9155

[29] An Approach Towards Sustainable Mobile Enterprise Architecture. [consulta em 2012-12-10]. Disponível em: http://sushmobile.co.nz/mobile-application-development-native-hybrid-or-web-optimised-mobile/

[30] **Biel**, B., **Grill**, Thomas., **Gruhn**, V., "Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application", *Journal of Systems and Software (JSS)* 83(11):2031-2044, 2010.

[31] **Krasner**, G., **Pope**, S. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of Object Oriented Programming*, Vol 1, Issue 3, pp. 26–49, 1988.

[32] **Reenskaug**, T., Thing-Model-View-Editor, Xerox Parc, 1979.

[33] **Goldberg**, A., **Robson**, D. Smalltalk-80: The Language and Its Implementation. Reading, MA: Addison-Wesley, 1983.

[34] **Reenskaug**, T. "MVC XEROX PARC 1978-79. [consulta em 2012-12-18].Disponível em: http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html.

[35] **Plakalović** D., **Simić** D. Applying MVC and PAC patterns in mobile applications. *Journal of Computing*, Vol. 2, ISSUE 1, January 2010, ISSN: 2151-9617. [consulta em 2012-12-18]. Disponível em: http://arxiv.org/ftp/arxiv/papers/1001/1001.3489.pdf.

[36] **Gamma**, E.; **Helm**, R.; **Johnson**, R.; **Vlissides**, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. USA. ISBN 0-201-63361-2

[37] **Potel**, M.(1996). *MVP:* Model-View-Presenter: The Taligent Programming Model for C++ and Java. [consulta em 2012-12-18]. Disponível em: http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

[38] Model View Controller, Model View Presenter, and Model View ViewModel Design Patterns. [consulta em 2012-12-20]. Disponível em: http://www.codeproject.com/Articles/42830/Model-View-Controller-Model-View-Presenter-and-Mod.

[39] **Bower**, A.; **McGlashan**, B. "Twisting the Triad: The evolution of the Dolphin Smalltalk MVP application framework", *European Smalltalk User Group* (ESUG), 2000.

[40] **Fowler**, M. GUI Architectures. [consulta em 2012-12-20]. Disponível em: http://martinfowler.com/eaaDev/uiArchs.html

[41] **Gossman**, J. (2005). Introduction to Model/View/ViewModel pattern for building WPF apps. [consulta em 2012-12-20]. Disponível em: http://blogs.msdn.com/johngossman/archive/2005/10/08/478683.aspx

[42] **Smith**, J. (2009). WPF Apps With the Model-View-ViewModel Design Pattern by. [consulta em 2012-12-20]. Disponível em: http://msdn.microsoft.com/en-us/magazine/dd419663.aspx

[43] **Amin,** A.Rasooli; Design patterns for user interface. [consulta em 2012-12-10]. Disponível em: http://www.csl.mtu.edu/cs4760/www/Projects/Graduate/grad6/www/topic_paper_amin_rasooli.pdf

[44] **Moran**, T. (1981) "The Command Language Grammars: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*.

[45] **Nielsen**, J., and **Molich**, R. (1990). Heuristic evaluation of user interfaces, *Proc. ACM CHI'90 Conf*. (Seattle, WA, 1–5 April), 249-256.

[46] *Nielsen*, J., "Enhancing the explanatory power of usability heuristics", *CHI'94 Conference Proceedings*, 1994.

[47] UI Guidelines for mobile and tablet web app design. [consulta em 2012-12-13]. Disponível em: http://www.mobilexweb.com/blog/ui-guidelines-mobile-tablet-design

[48] **Neil**, T. "Mobile Design Pattern Gallery, Color Edition", UI Patterns for iOS, Android, and More,1449336442, 2012.

[49] **Bermudez**, M. (2012). A Mobile Field Diary System for Infrastructure Construction Project Inspections, *10th Latin American and Caribbean Conference for Engineering and Technology*. [consulta em 2012-12-18]. Disponível em: http://www.laccei.org/LACCEI2012-Panama/RefereedPapers/RP286.pdf

[50] MvvmCross | Cirrious. [consulta em 2013-04-15]. Disponível em: http://www.cirrious.com/2012/11/mvvmcross.html

[51] Cross-Platform Development with the .NET Framework. [consulta em 2013-02-13]. Disponível em: http://msdn.microsoft.com/en-us/library/gg597391.aspx

[52] Forwards: Using Portable Library tools for MonoTouch and MonoDroid. [consulta em 2013-04-26]. Disponível em: http://slodge.blogspot.pt/2012/04/using-portable-library-tools-for.html

[53] Databinding. [consulta em 2013-05-11]. Disponível em: · https://github.com/MvvmCross/MvvmCross/wiki/Databinding

Anexos

Anexo 1 – Casos de uso

Nome do caso de uso	**Efetuar login**
Sumário	O utilizador acede ao sistema usando os seus dados de acesso
Ator primário	Utilizador (Técnico; Chefia)
Precondições	
Fluxo Principal	**1.** O sistema verifica as credenciais de autenticação de utilizador guardadas no dispositivo; **2.** O sistema apresenta os campos de nome de utilizador e palavra-chave preenchidos com as credencias guardadas; **3**. O utilizador clica no botão de confirmação para acesso ao sistema "Login"; **4.** O sistema verifica as credenciais de autenticação; **5.** O sistema apresenta o ecrã de lista de tarefas, ou caso haja uma sincronização automática ativada nesse instante, o ecrã de sincronização.
Fluxo Alternativo	**1b.** Não existem credenciais de autenticação guardadas: **1b. 1)** O utilizador preenche os campos do nome de utilizador e a palavra-chave; **1b. 2)** O utilizador é direcionado para o passo 3 do fluxo principal. **4a.** As credencias estão incorretas, ou em branco: **4a. 1)** O sistema apresenta uma mensagem de informação; **4a. 2)** O utilizador preenche os campos do nome de utilizador e a palavra-chave;

	4a. 3) O utilizador é direcionado para o passo 3 do fluxo principal.
Pós-condições	O utilizador está identificado na aplicação do dispositivo móvel.
Regras de negócio	
Requisitos não funcionais	**RNF9, RNF43**

Tabela 1 - Descrição do caso de uso – Efetuar login

Nome do caso de uso	**Agrupar tarefas**
Sumário	O utilizador pode agrupar as tarefas por dia
Ator primário	Utilizador (Técnico; Chefia)
Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador seleciona o parâmetro para agrupar a lista de tarefas por dia; **2.** O sistema apresenta a lista de tarefas ordenada, da mais recente para a mais antiga.
Fluxo Alternativo	
Pós-condições	
Regras de negócio	
Requisitos não funcionais	**RNF10, RNF11**

Tabela 2 - Descrição do caso de uso - Agrupar tarefas

Nome do caso de uso	**Filtrar tarefas**
Sumário	O utilizador pode filtrar a lista das tarefas por qualquer campo à sua escolha existente na lista.
Ator primário	Utilizador (Técnico; Chefia)

Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador seleciona o campo de filtro das tarefas carregadas no dispositivo; 2. O utilizador digita o texto a procurar e confirma a operação; **2.** O sistema apresenta a lista de tarefas relacionadas com o filtro e o campo de pesquisa.
Fluxo Alternativo	
Pós-condições	
Regras de negócio	
Requisitos não funcionais	**RNF10, RNF12**

Tabela 3 - Descrição do caso de uso - Filtrar tarefas

Nome do caso de uso	**Configurar sistema**
Sumário	O utilizador pode alterar algumas configurações do sistema a seu gosto (RF21, RF22,RF23).
Ator primário	Utilizador (Técnico; Chefia)
Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador escolhe a opção de configuração do sistema no menu da aplicação; **2.** O sistema apresenta o ecrã de configuração; **3.** O utilizador altera as configurações; **4.** O sistema grava as alterações efetuadas.
Fluxo Alternativo	
Pós-condições	

Regras de negócio	
Requisitos não funcionais	**RNF14, RNF44**

Tabela 4 - Descrição do caso de uso - Configurar sistema

Nome do caso de uso	**Sincronizar**
Sumário	O utilizador recebe as tarefas marcadas no *eUrban* para execução/despacho no dispositivo móvel e envia as tarefas executadas/com despacho do dispositivo para o *eUrban*.
Ator primário	Utilizador (Técnico; Chefia)
Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal **(sincronização por opção)**	**1.** O utilizador escolhe a opção de sincronização no menu da aplicação; **2.** O sistema apresenta o ecrã de sincronização com as tarefas que estão para receber/enviar do *eUrban*; **3.** O utilizador seleciona as tarefas dos processos e/ou documentos que quer enviar e confirma a sincronização; **4.** O sistema mostra a informação do sincronismo de cada tarefa; **5.** O sistema apresenta um quadro de resumo das operações.
Fluxo Alternativo	**5a.** Erro de sincronização: **5a. 1)** O sistema apresenta a mensagem de erro associado a cada item que não passou na sincronização; **5a. 2)** O sistema permite ao utilizador tentar sincronizar cada item ou todos os itens que falharam; **5a. 3)** O utilizador é direcionado para o passo 3 do fluxo principal. **(sincronização inicial automática)** **1.** A sincronização é iniciada automaticamente pelo sistema, se houver uma configuração para que tal aconteça e haja dados para transferir (O sistema apresenta o ecrã de sincronização. com as tarefas que estão para receber/enviar do *eUrban*).

	2. Continua normalmente no ponto 4 do fluxo principal;
Pós-condições	A lista de tarefas a realizar foi atualizada.
Regras de negócio	
Requisitos não funcionais	**RNF17, RNF25, RNF32, RNF37.**

Tabela 5 - Descrição do caso de uso – Sincronizar

Nome do caso de uso	**Sair da sessão**
Sumário	O utilizador elimina as suas credenciais de autenticação guardadas no sistema (próximo acesso do utilizador ao sistema necessita que coloque os dados de acesso nos campos utilizador e palavra-chave).
Ator primário	Utilizador (Técnico; Chefia)
Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador escolhe a opção de sair da sessão (logout) no menu da aplicação; **2.** O sistema apresenta uma mensagem de decisão; **3.** O utilizador confirma a saída da sessão; **2.** O sistema elimina as credenciais do utilizador.
Fluxo Alternativo	**3a.** O utilizador não confirma a saída da sessão: **3a. 1)** O sistema mostra o ecrã onde o utilizador escolheu a opção para sair da sessão.
Pós-condições	O sistema apresenta ao utilizador o ecrã para efetuar login.
Regras de negócio	
Requisitos não funcionais	**RNF13**

Tabela 6 - Descrição do caso de uso - Sair da sessão

Nome do caso de uso	Escolher tarefa
Sumário	O utilizador pode aceder a toda a informação relacionada com uma tarefa no dispositivo móvel.
Ator primário	Utilizador (Técnico; Chefia)
Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador clica num item da lista, no ecrã da lista de tarefas; **2.** O sistema apresenta o ecrã da tarefa com toda a informação associada ao processo a que pertence.
Fluxo Alternativo	
Pós-condições	O sistema mostra o ecrã com a informação da tarefa.
Regras de negócio	Um utilizador só pode escolher uma tarefa de cada vez.
Requisitos não funcionais	**RNF15**

Tabela 7 - Descrição do caso de uso - Escolher tarefa

Nome do caso de uso	Ver Resumo do Processo
Sumário	O Utilizador pode visualizar a informação do processo à qual pertence a tarefa.
Ator primário	Utilizador (Técnico; Chefia)
Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador seleciona o separador que contém a informação resumo do processo, caso este não esteja ativo; **2.** O sistema mostra a informação resumo do processo à qual pertence a tarefa.
Fluxo Alternativo	
Pós-condições	

Regras de negócio	
Requisitos não funcionais	**RNF15, RNF28,RNF33**
Autor	Rui Ferreira
Data	22/01/2013

Tabela 8 - Descrição do caso de uso - Ver resumo do Processo

Nome do caso de uso	**Visualizar documentos**
Sumário	O Utilizador pode visualizar os documentos relacionados com uma tarefa.
Ator primário	Utilizador (Técnico; Chefia)
Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador seleciona o separador da lista de documentos, caso este não esteja ativo; **2.** O sistema mostra a lista de documentos relacionados com a tarefa; **3.** O utilizador escolhe qual o documento que pretende ver; **4.** O sistema abre o documento no dispositivo.
Fluxo Alternativo	**4a.** Não foi possível abrir o documento, ou tem um formato incompatível: **4a. 1)** O sistema mostra uma mensagem de erro; **4a. 2)** O utilizador é direcionado para o passo 2 do fluxo principal.
Pós-condições	O sistema abre o documento selecionado no dispositivo.
Regras de negócio	
Requisitos não funcionais	**RNF15, RNF28, RNF34,RNF35**

Tabela 9 - Descrição do caso de uso - Visualizar documentos

Nome do caso de uso	**Carregar documentos**
Sumário	O Utilizador pode carregar documentos existentes em memória no dispositivo e juntá-los a uma tarefa do processo, para que possam ser sincronizados posteriormente com o *eUrban*.
Ator primário	Utilizador (Técnico; Chefia)
Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador seleciona o separador da lista de documentos, caso este não esteja ativo; **2.** O sistema mostra a lista de documentos relacionados com a tarefa; **3.** O utilizador pressiona o botão para carregar ficheiros do dispositivo; **4.** O sistema mostra a árvore de diretórios de ficheiros; **5.** O utilizador seleciona o (s) ficheiro (s) pretendido (s); **6.** O sistema relaciona o (s) ficheiro (s) com a tarefa;
Fluxo Alternativo	**6a.** Não foi possível anexar o (s) documento (s): **6a. 1)** O sistema mostra uma mensagem de erro; **6a. 2)** O utilizador é direcionado para o passo 2 do fluxo principal.
Pós-condições	Os documentos são anexados à tarefa.
Regras de negócio	
Requisitos não funcionais	**RNF15, RNF16, RNF35**

Tabela 10 - Descrição do caso de uso - Carregar documentos

Nome do caso de uso	**Executar tarefa**
Sumário	O técnico realiza a execução de uma tarefa preenchendo campos e respondendo a questões referente à vistoria da obra. (a tarefa passa para o estado pendente para envio ao *eUrban*).
Ator primário	Utilizador (Técnico)

Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador seleciona o separador de execução da tarefa, caso este não esteja ativo; **2.** O sistema mostra os campos da tarefa para preenchimento; **3.** O utilizador preenche os campos disponibilizados (informação acerca da tarefa e da vistoria da obra); **4.** O utilizador confirma o preenchimento (pressiona o botão para executar tarefa); **5.** O sistema cria o documento (relatório técnico).
Fluxo Alternativo	**5a.** Não foi possível executar a tarefa no dispositivo: **5a. 1)** O sistema mostra uma mensagem de erro; **5a. 2)** O utilizador é direcionado para o passo 2 do fluxo principal.
Pós-condições	A tarefa é incluída numa lista pendente de envio.
Regras de negócio	Um utilizador só pode executar uma tarefa de cada vez no dispositivo móvel.
Requisitos não funcionais	**RNF15, RNF16**

Tabela 11 - Descrição do caso de uso - Executar tarefa

Nome do caso de uso	**Despachar tarefa**
Sumário	A Chefia depois de verificar a tarefa e o relatório técnico efetuado pelo técnico, difere ou indefere, assinando o despacho da tarefa.
Ator primário	Utilizador (Chefia)
Precondições	O utilizador está identificado na aplicação do dispositivo móvel.
Fluxo Principal	**1.** O utilizador executa a tarefa; **2.** O sistema mostra um despacho, documento em extensão *pdf*; **3.** O utilizador assina digitalmente o documento, usando o certificado digital; **4.** O sistema coloca a tarefa numa lista pendente de envio.

Fluxo Alternativo	
Pós-condições	A tarefa é incluída numa lista pendente de envio. A lista de tarefas a executar pelo utilizador é atualizada.
Regras de negócio	Um utilizador só pode executar uma tarefa de cada vez no dispositivo móvel.
Requisitos não funcionais	

Tabela 12 - Descrição do caso de uso - Despachar tarefa

Anexo 2 – Interfaces

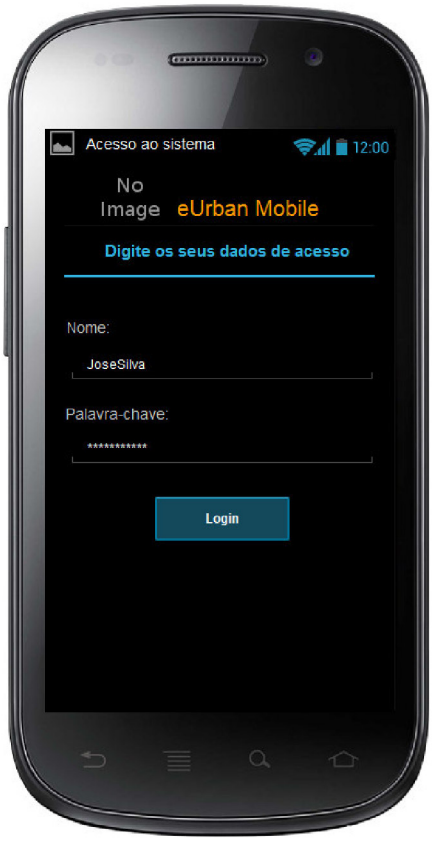


Figura 29 - Efetuar login

Figura 30 - Visualizar tarefas (modo normal)

Figura 31 - Visualizar tarefas (agrupadas por data)

Figura 32 - Menu da aplicação na lista de tarefas

Figura 33 - Configurar filtros de tarefas

Figura 34 - Sair da sessão

Figura 35 - Sincronizar tarefas (enviar e receber)

Figura 36 - Sincronização de tarefas

Figura 37 – Ecrã de sincronização de tarefas

Figura 38 - Resumo do processo

Figura 39 - Visualizar documentos da tarefa

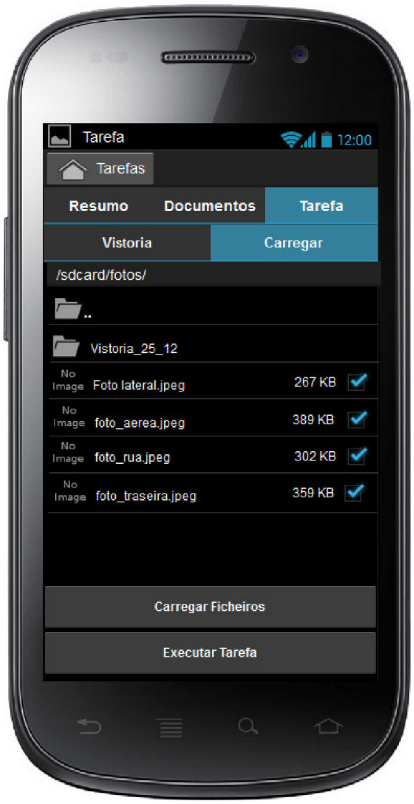


Figura 40 - Adicionar ficheiros a uma tarefa

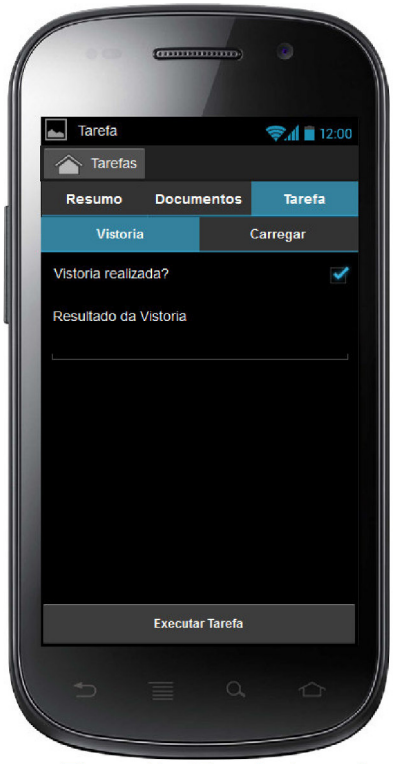


Figura 41 - Vistoria da tarefa

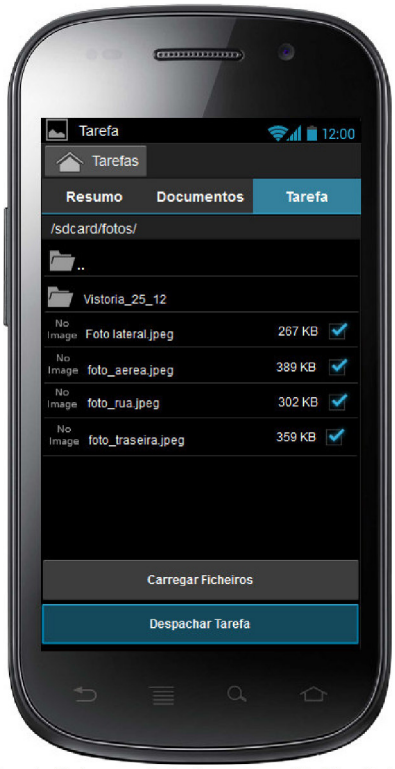


Figura 42 - Adicionar ficheiros (Perfil Chefia)

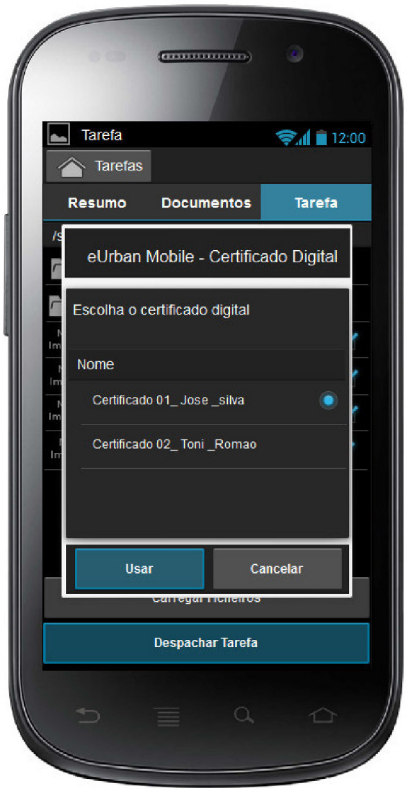


Figura 43 - Despachar uma tarefa (Perfil chefia)

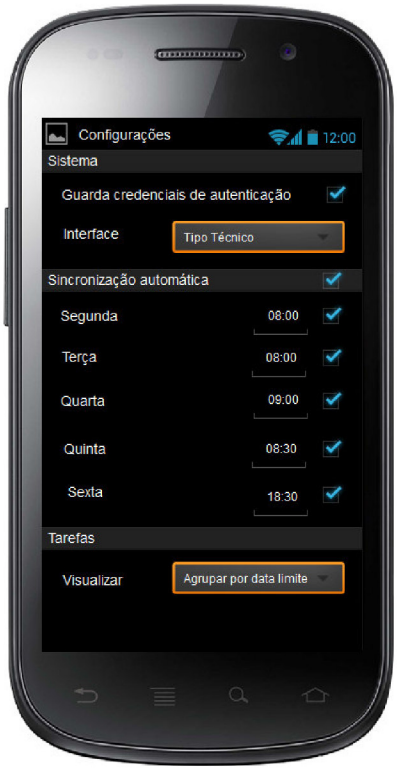


Figura 44 – Ecrã de Configurações

Figura 45 - Efetuar login

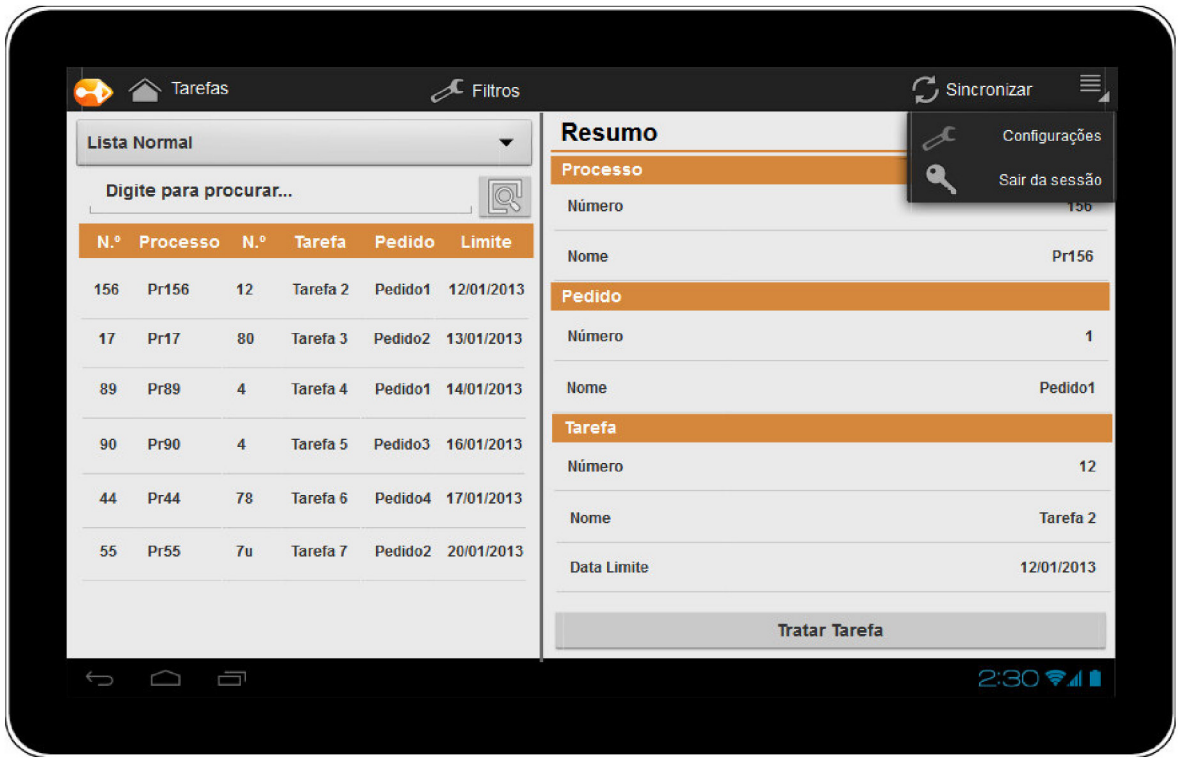


Figura 46 - Visualizar tarefas (modo normal)

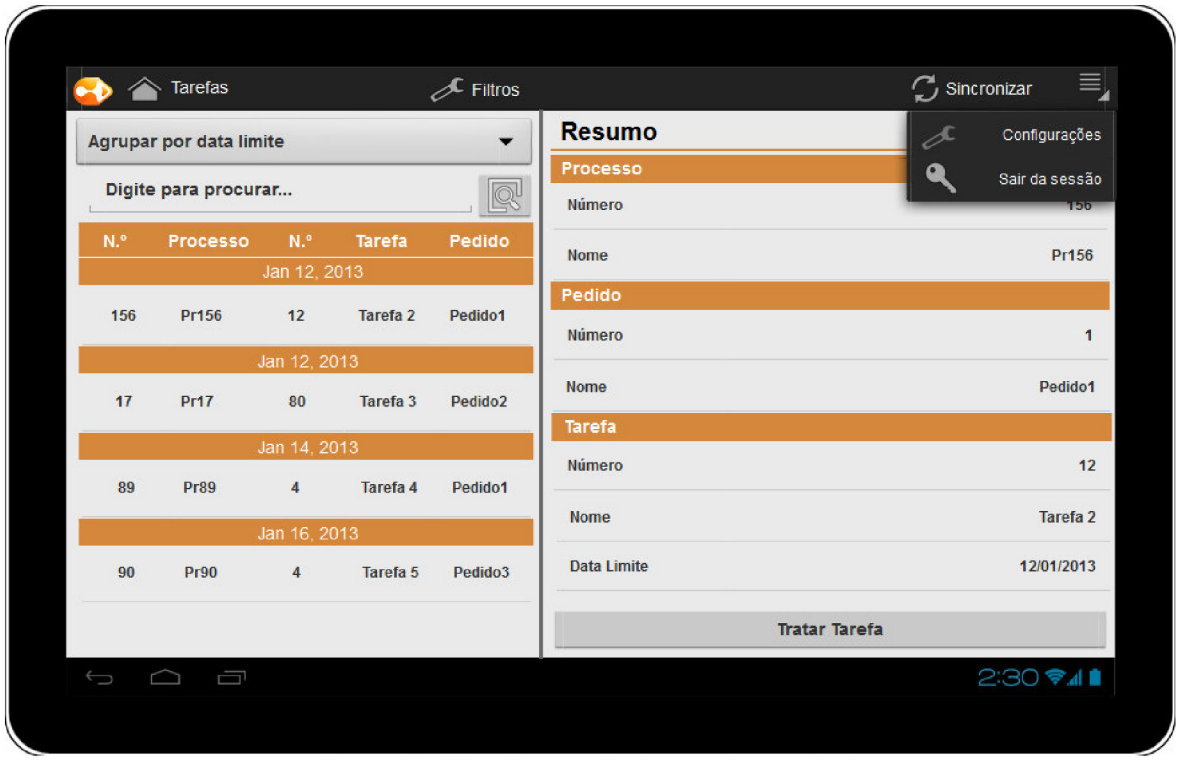


Figura 47 - Visualizar tarefas (agrupadas por data)

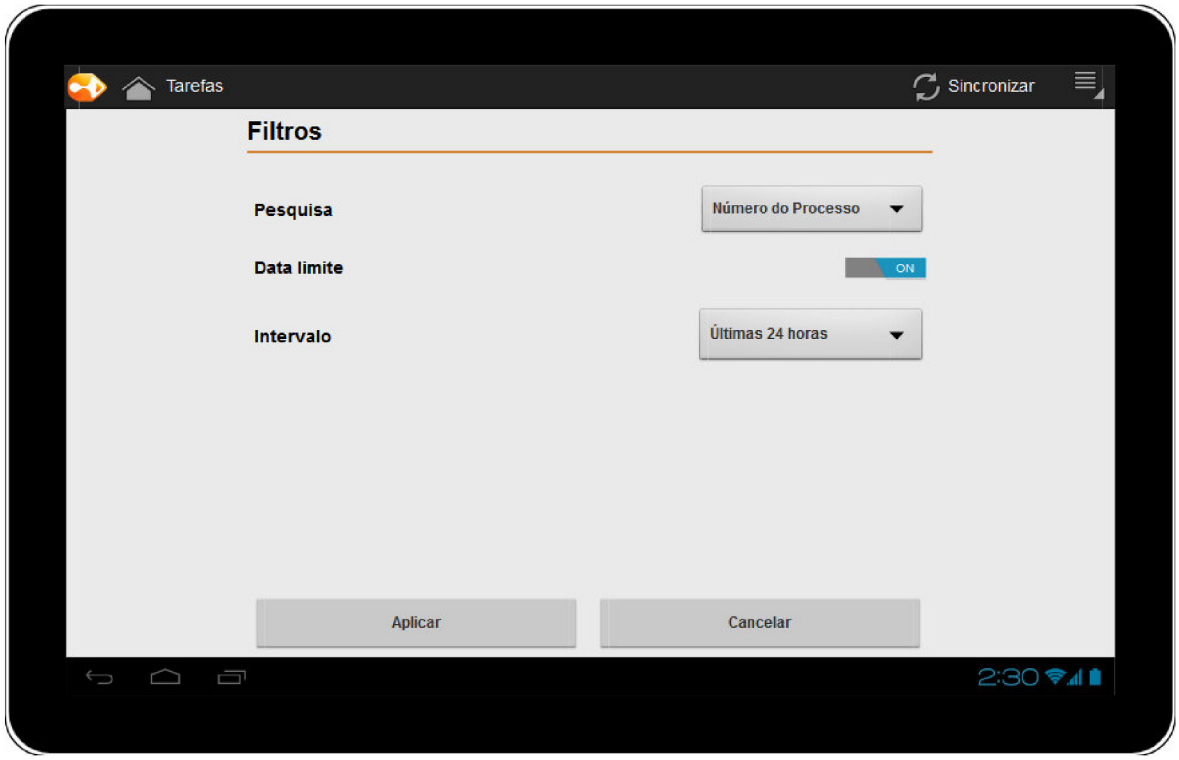


Figura 48 - Configurar filtros de tarefas

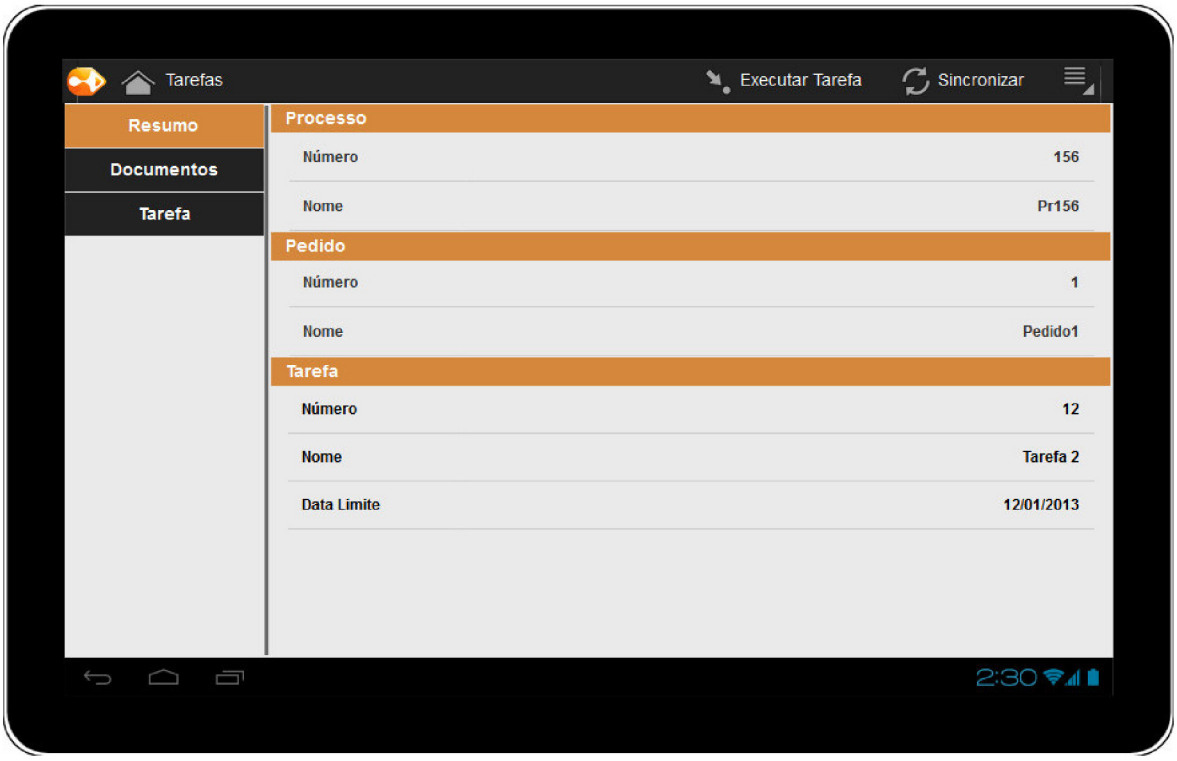


Figura 49 - Resumo do processo

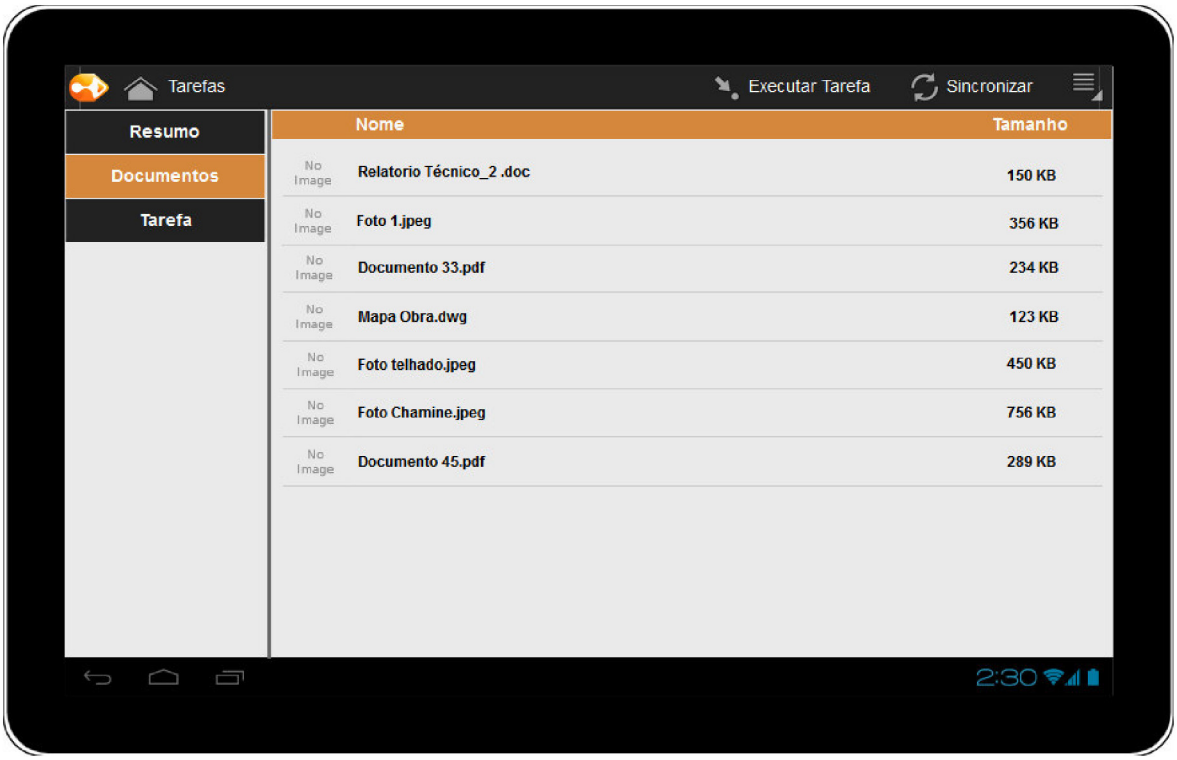


Figura 50 - Visualizar documentos da tarefa

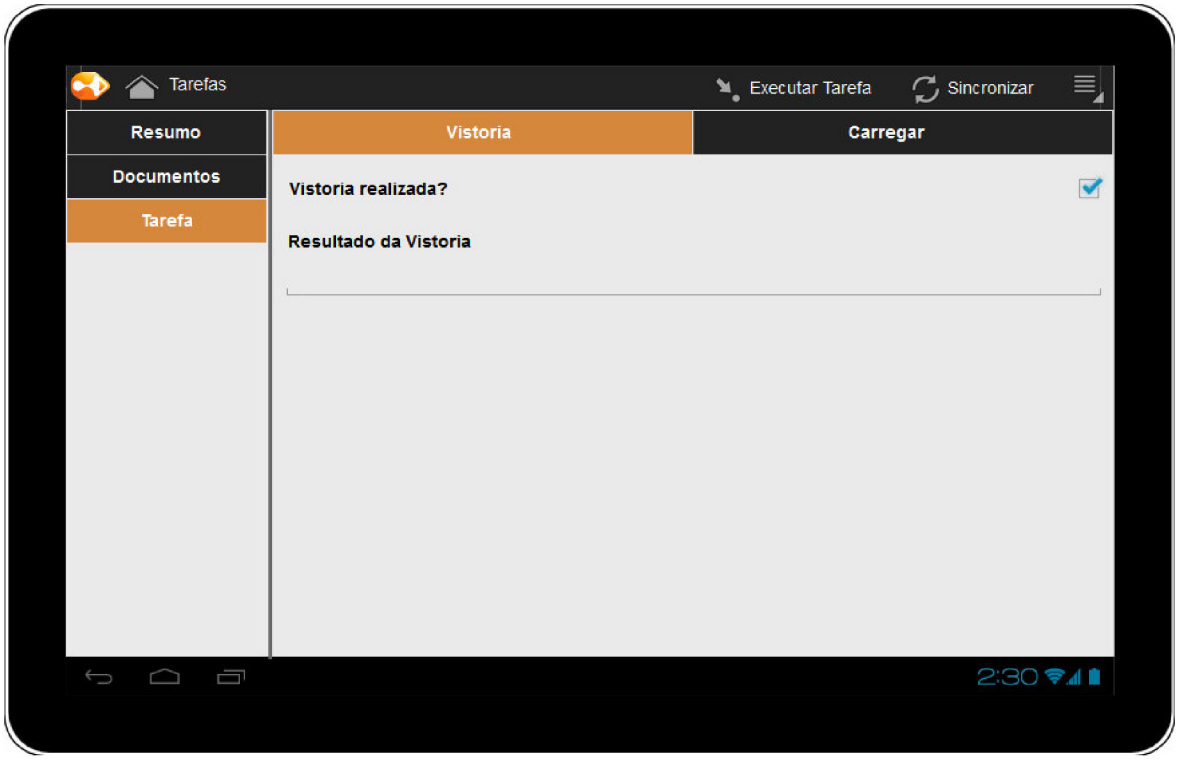


Figura 51 - Vistoria da tarefa

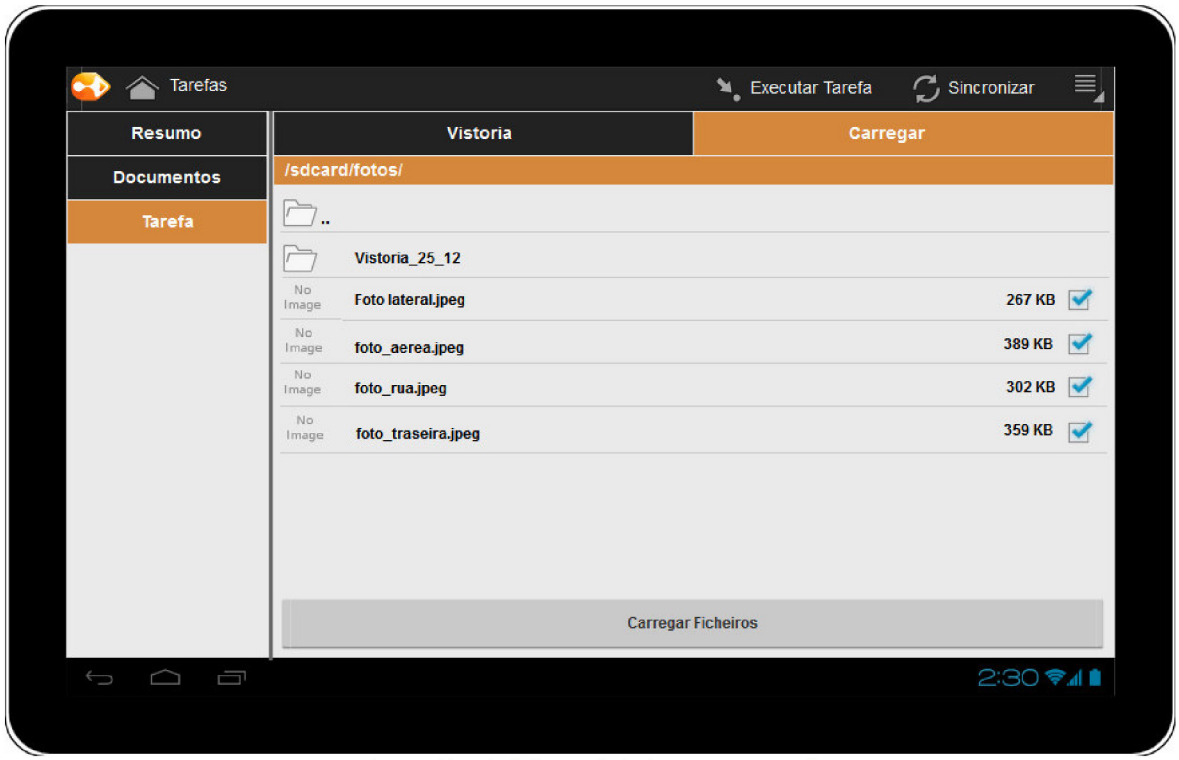


Figura 52 - Adicionar ficheiros a uma tarefa

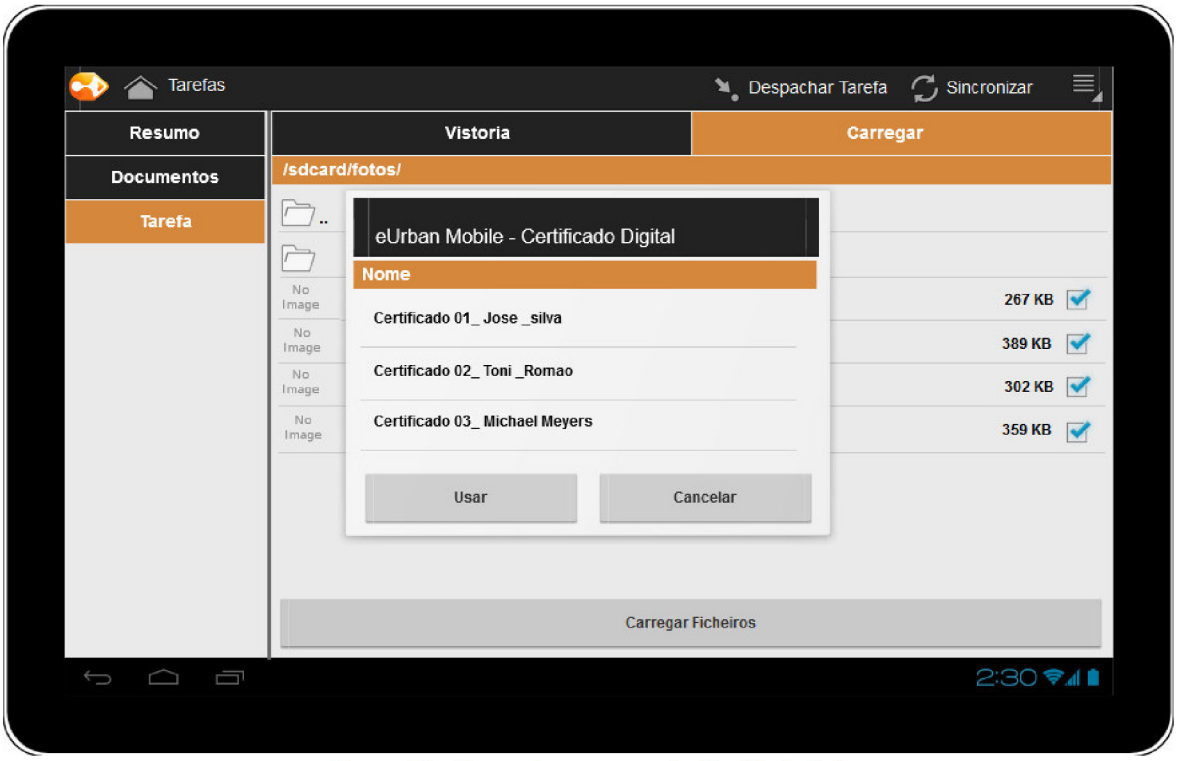


Figura 53 - Despachar uma tarefa (Perfil chefia)

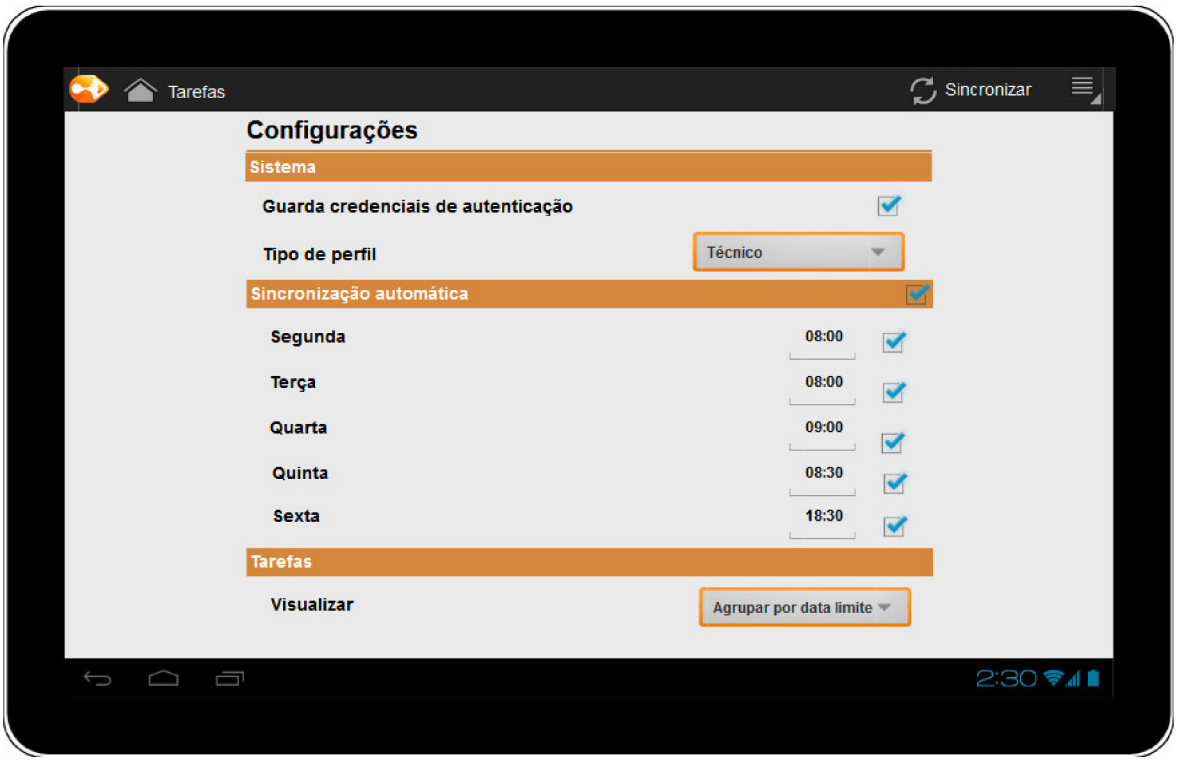


Figura 54 - Configurar sistema

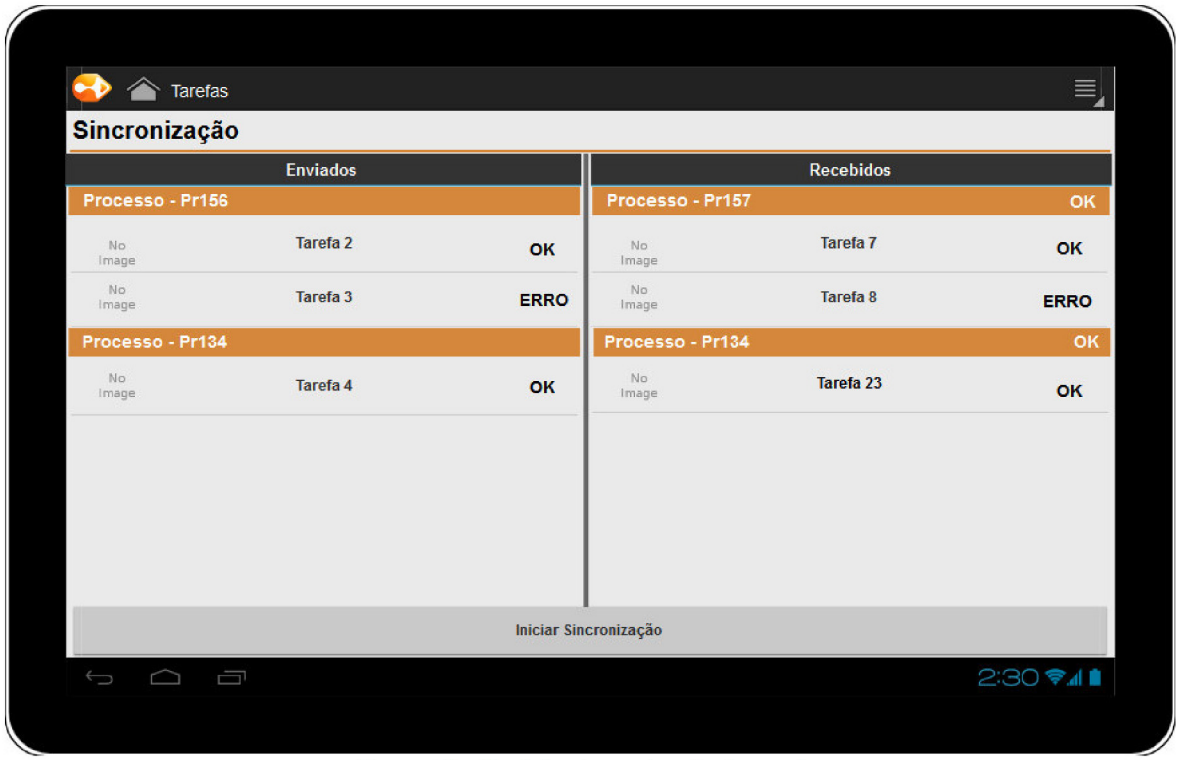


Figura 55 - Ecrã de sincronização de tarefas

Figura 56 -Sincronização de tarefas

Figura 57 - Sair da sessão

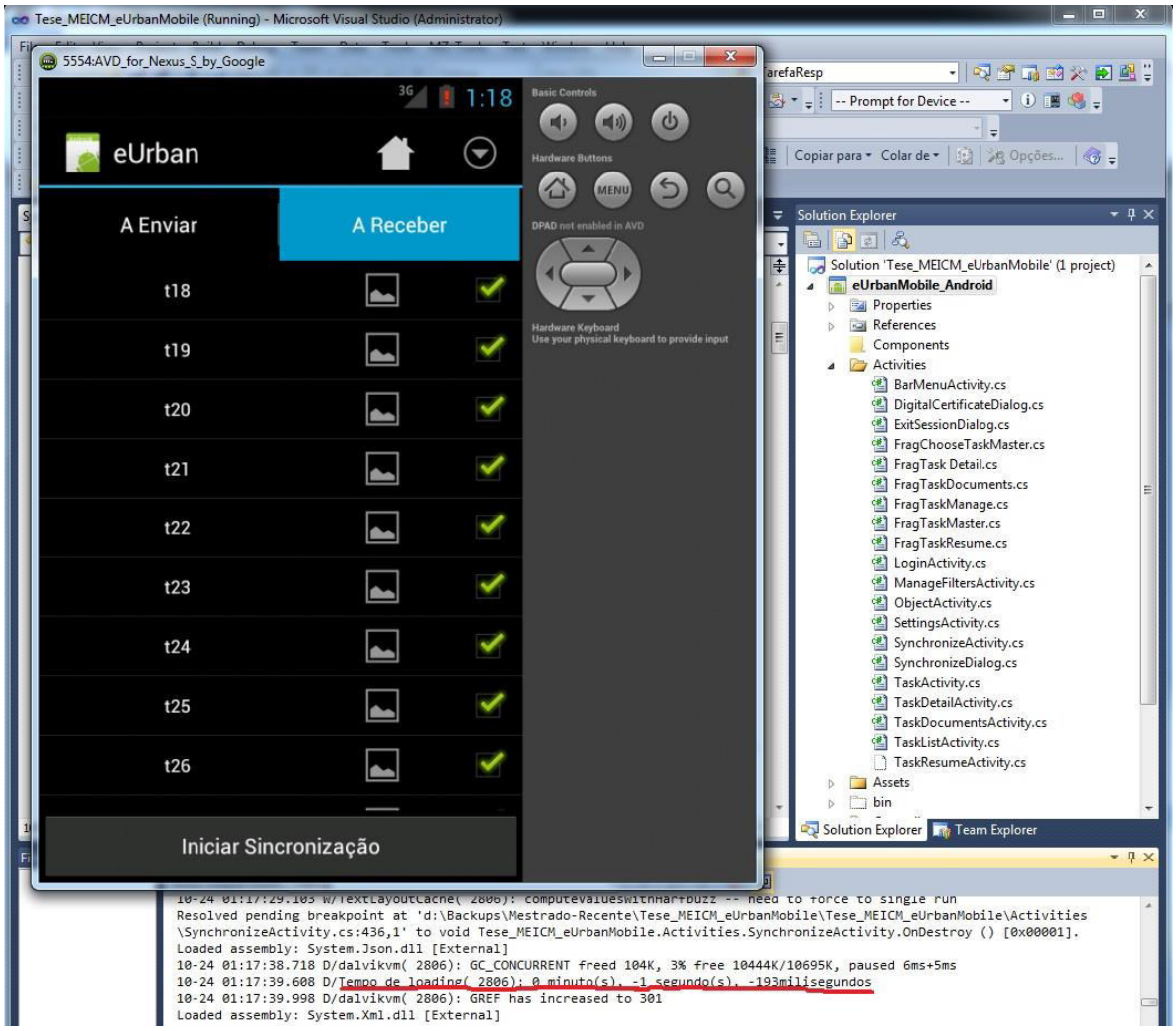


Figura 58 - Versão Android (ecrã de sincronização) com controladores e sem MvvmCross

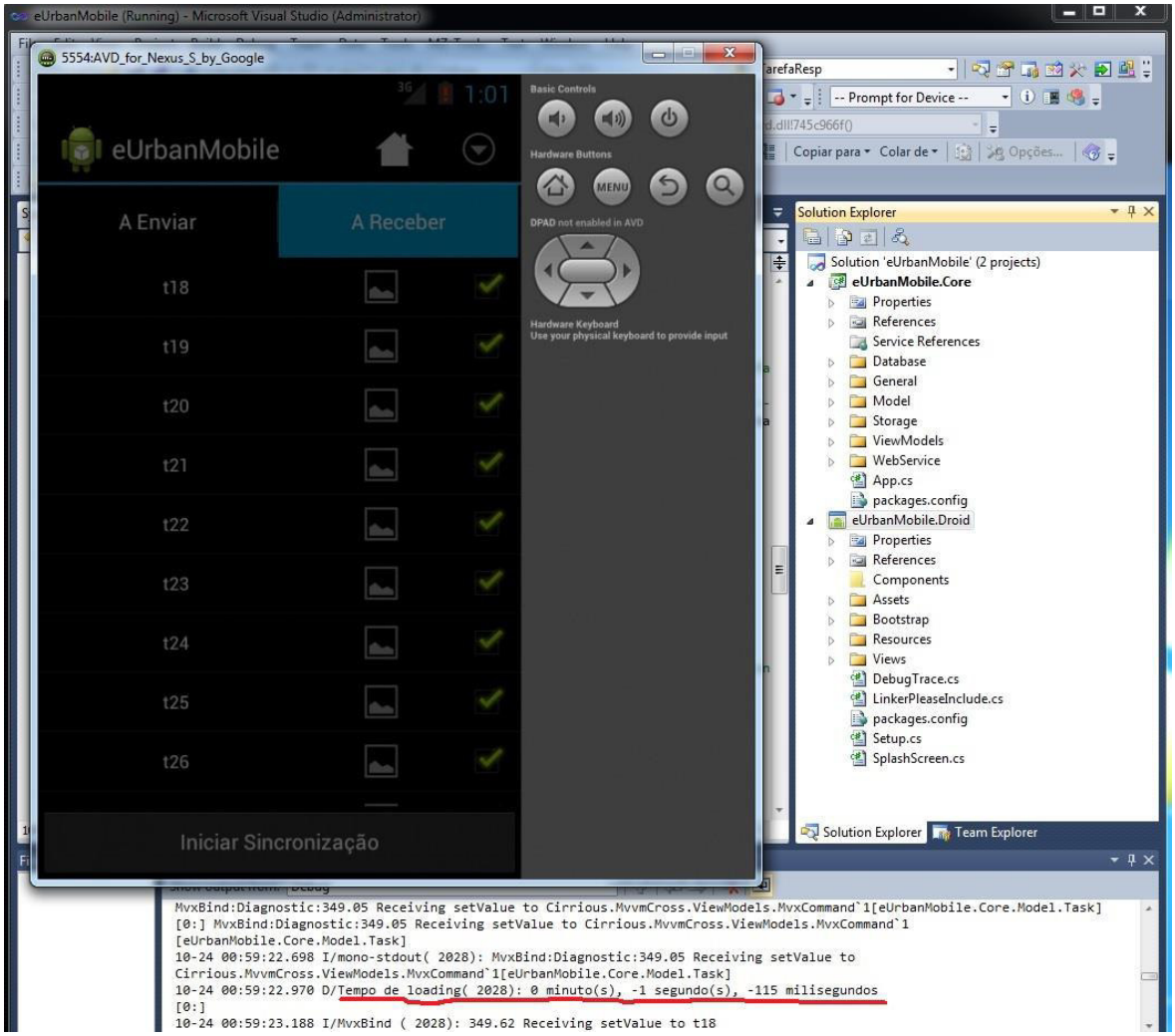


Figura 61 - Versão Android (ecrã de sincronização) com MvvmCross e sem Controladores

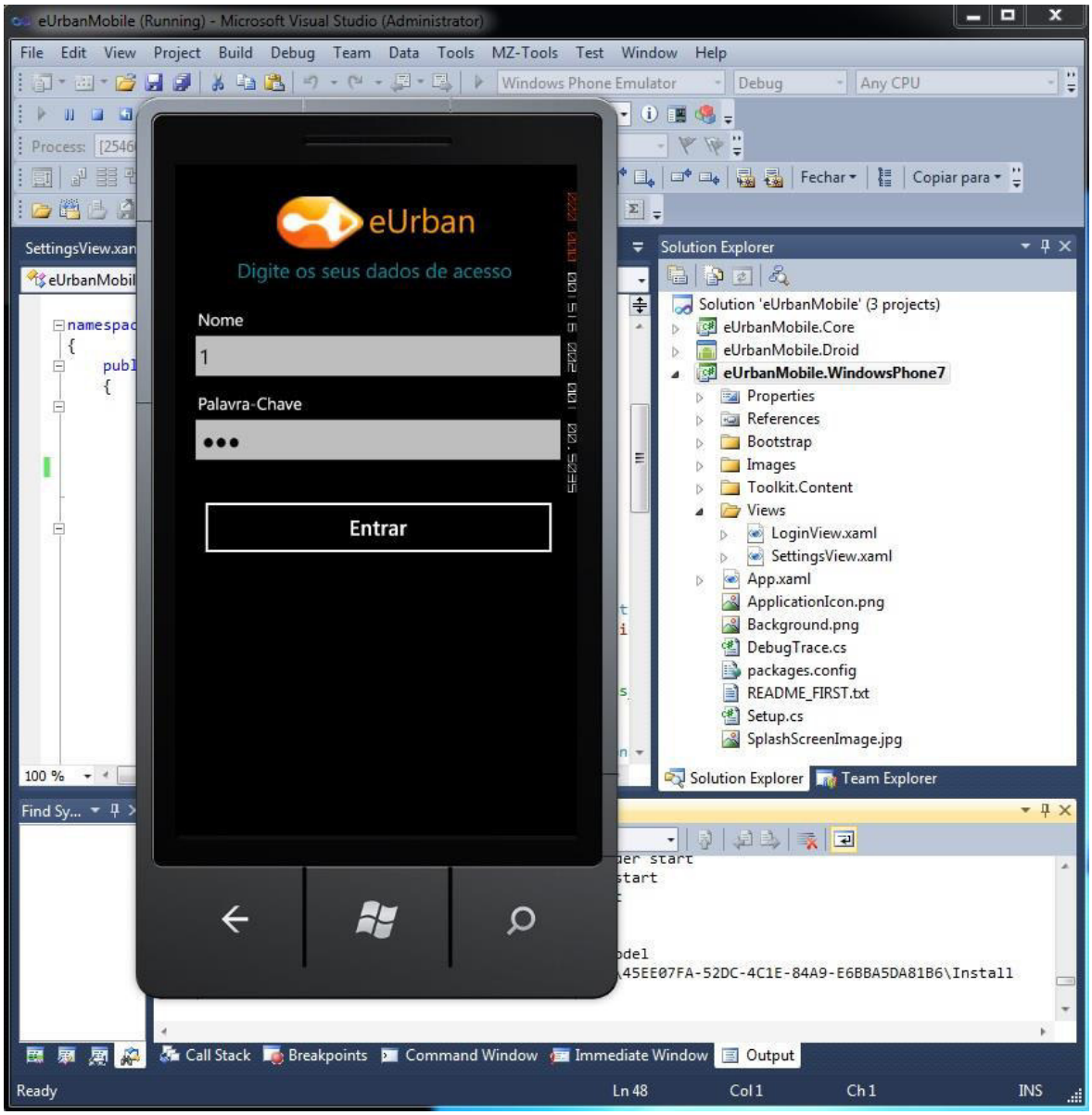


Figura 59 - Ecran de Login - Windows Phone 7

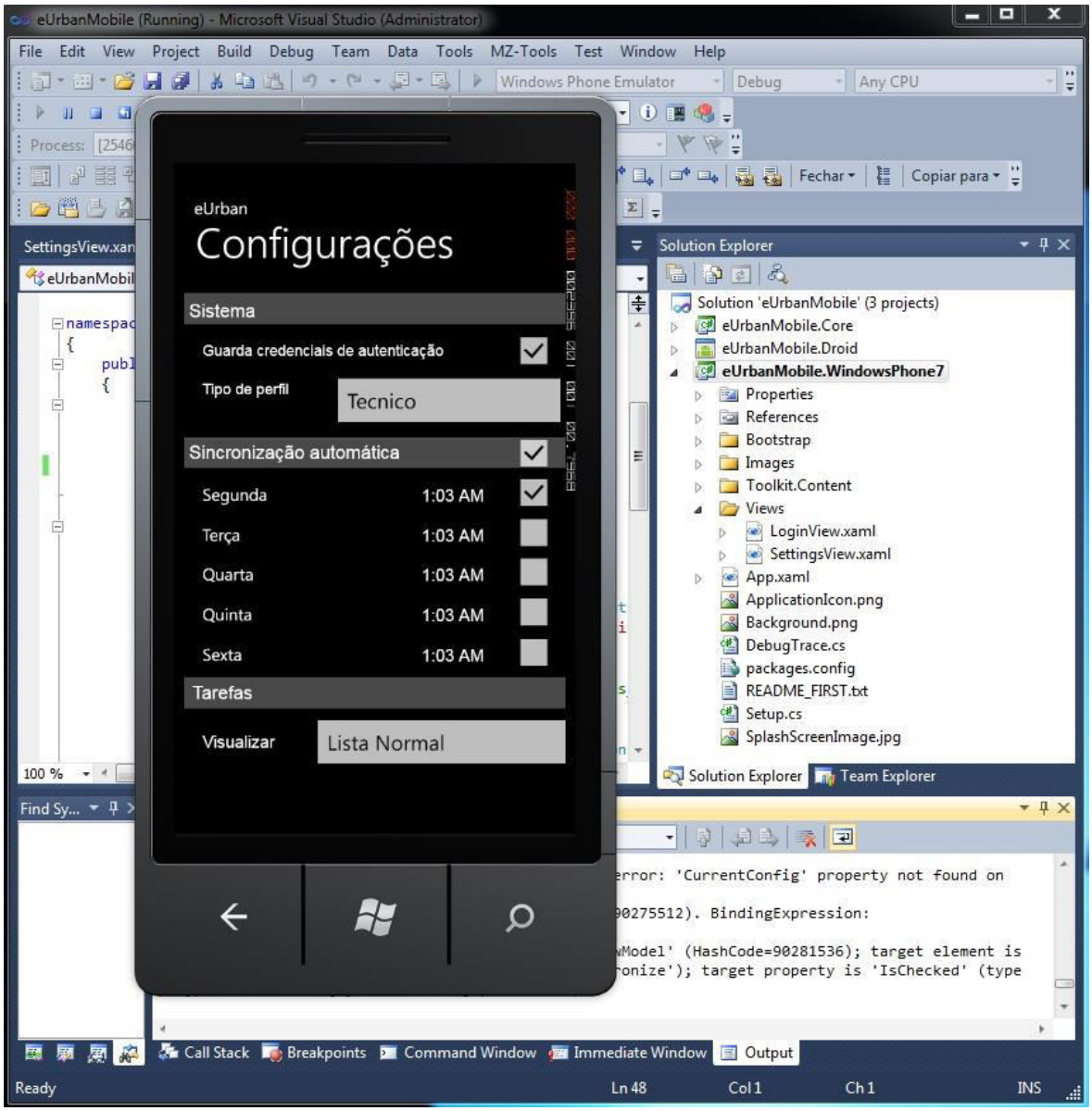


Figura 60 - Ecran de configurações - Windows Phone 7

Anexo 3 – Análise de desempenho

Sem tarefas	
Ecrans	Tempo
Executar tarefas	315 ms
Configurações	600 ms
Filtros	153 ms
Sincronizar Tarefas	185 ms
Inserção e escrita	0 ms

1 tarefa	
Ecrans	Tempo
Executar tarefas	321 ms
Configurações	623 ms
Filtros	106 ms
Sincronizar Tarefas	190 ms
Inserção e escrita	290 ms

10 tarefas	
Ecrans	Tempo
Executar tarefas	325 ms
Configurações	607 ms
Filtros	118 ms
Sincronizar Tarefas	194 ms
Inserção e escrita	1s 410 ms

20 tarefas	
Ecrans	Tempo
Executar tarefas	317 ms
Configurações	561 ms
Filtros	124 ms
Sincronizar Tarefas	196 ms
Inserção e escrita	2s 626 ms

30 tarefas	
Ecrans	Tempo
Executar tarefas	352 ms
Configurações	589 ms
Filtros	139 ms
Sincronizar Tarefas	195 ms
Inserção e escrita	3s 914 ms

40 tarefas	
Ecrans	Tempo
Executar tarefas	3 57 ms
Configurações	596 ms
Filtros	128 ms
Sincronizar Tarefas	194 ms
Inserção e escrita	5s 441 ms

50 tarefas	
Ecrans	Tempo
Executar tarefas	355 ms
Configurações	586 ms
Filtros	112 ms
Sincronizar Tarefas	193 ms
Inserção e escrita	7s 38 ms

100 tarefas	
Ecrans	Tempo
Executar tarefas	360 ms
Configurações	571 ms
Filtros	138 ms
Sincronizar Tarefas	197 ms
Inserção e escrita	13s 325 ms

200 tarefas	
Ecrans	Tempo
Executar tarefas	364 ms
Configurações	568 ms
Filtros	114 ms
Sincronizar Tarefas	197 ms
Inserção e escrita	27s 195 ms

Figura 61 - Tempos médio de algumas funcionalidades da aplicação (s-segundos; ms-milissegundos) consoante o número de tarefas envolvidas

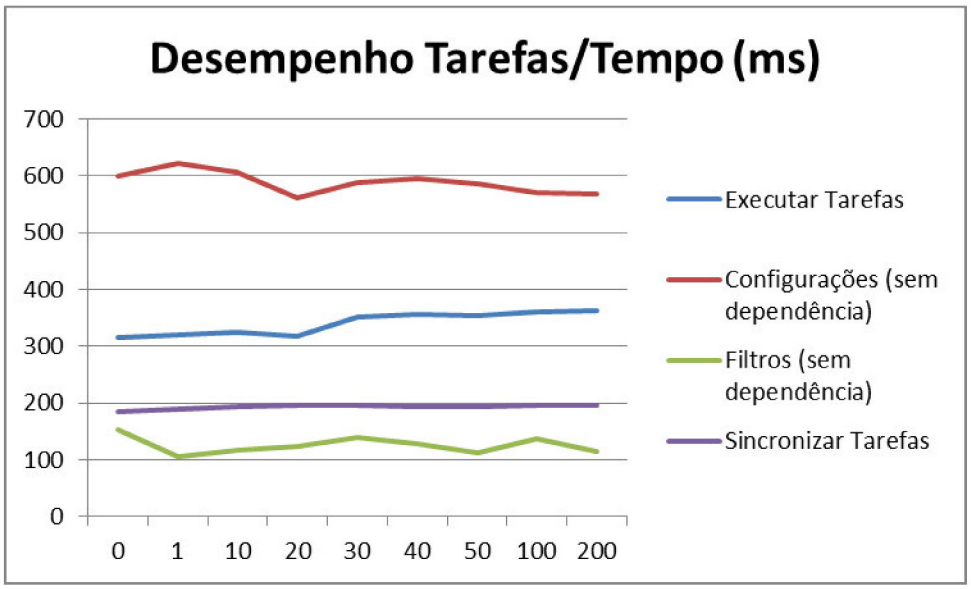


Figura 62 - Tempo médio (milissegundos) de visualização da informação em alguns écrans, consoante o número de tarefas envolvidas

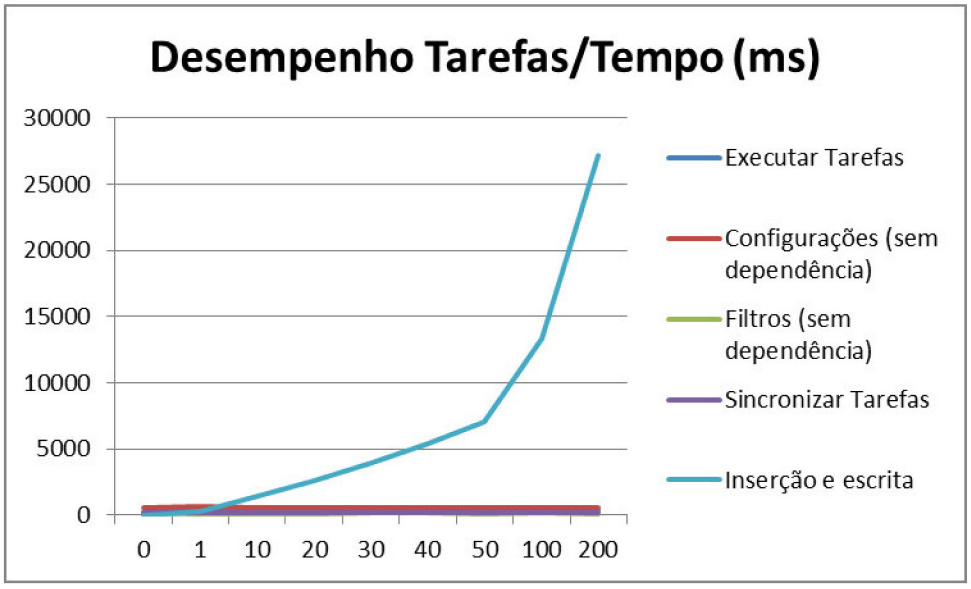


Figura 63 – Comparação do tempo médio (milissegundos) de execução de operações de inserção e escrita com operações de visualização de dados