Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

MESTRADO EM INFORMÁTICA E SISTEMAS

DESENVOLVIMENTO DE SOFTWARE

Relatório Final de Estágio

Desenvolvimento de aplicações móveis para Windows Phone

Mário José Bento Oliveira [a21170292@alunos.isec.pt]

Coimbra, Julho, 2014

Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

MESTRADO EM INFORMÁTICA E SISTEMAS

DESENVOLVIMENTO DE SOFTWARE

Relatório Final de Estágio

Desenvolvimento de aplicações móveis para Windows Phone

Estágio sob orientação dos Exmos.:

Mestre Nuno Martins (ISA)

Doutor João Cunha (ISEC)

Coimbra, Julho, 2014

Agradecimentos

Os resultados atingidos neste estágio foram fruto não só do meu trabalho, mas também do empenho de um conjunto de pessoas que me ajudaram na realização deste estágio.

Mostro assim a minha gratidão, com poucas palavras, mas com grande sentimento, a todos os que tiveram presentes nos momentos de aprendizagem, angústia, ansiedade e satisfação.

Ao meu orientador, Prof. Doutor João Carlos Costa Faria da Cunha, o meu sincero agradecimento pela forma como me orientou, pela disponibilidade que sempre revelou, pelo entusiasmo e motivação.

A todos os elementos da equipa com que trabalhei, expresso o meu profundo agradecimento pela ajuda que me prestaram e a forma como me receberam. O resultado do meu estágio contempla o empenho de todos os elementos da equipa, agradeço ao Mestre Nuno Martins, Engenheiro Luís Carvalho, Mestre Ester Soares, Engenheiro Tiago Correia, Mestre João Lourenço, Mestre Pedro Sá, Engenheiro Ricardo Germano, Mestre Pedro Saraiva e ao Mestre Gonçalo Rodrigues.

À Mestre Andreia Carreiro e Designer Sónia Ferreira, por todo o apoio e ajuda prestada durante o decorrer do estágio.

À minha família, em especial aos meus pais e ao meu irmão, um enorme obrigado pela força, apoio e dedicação que, constantemente me oferecem. Expresso a minha gratidão à minha namorada pela força, carinho prestados nos bons e nos maus momentos.

Resumo

A realização do presente estágio teve como intuito criar competências na empresa ISA (Intelligent Sensing Anywhere) para o desenvolvimento de aplicações móveis para Windows Phone. A ISA é uma empresa de base tecnológica que tem vindo a desenvolver produtos e soluções para o mercado de energia, ambiente, gás e petróleo. A empresa tem vindo a disponibilizar as suas soluções através de terminais móveis, nomeadamente iOS e Android, tendo sentido necessidade de ganhar competências na plataforma Windows Phone devido à sua evolução do *Market Share*, passando assim a oferecer esta solução aos seus clientes de forma mais generalizada.

De forma a que a ISA conseguisse ganhar competências em Windows Phone, o presente estágio teve como objetivos desenvolver uma aplicação Cloogy para esta plataforma e em simultâneo criar uma *framework* para o desenvolvimento em Windows Phone, convenientemente adaptada aos processos da ISA.

Durante o estágio foram desenvolvidas diversas tarefas que fazem parte de todas as etapas de desenvolvimento de *software* através de uma metodologia ágil, o *Scrum*. Foram levantados e especificados requisitos, definida a arquitetura e implementadas e testadas todas as funcionalidades do *software*. De modo a que os utilizadores usufruíssem da solução, a mesma foi publicada na Windows Phone Store.

A experiência adquirida ao longo das várias etapas do desenvolvimento em *Scrum*, essencialmente nas cerimónias *Daily Meeting* e *Sprint Retrospective*, foi uma mais-valia para a conceção da *framework*. Esta experiência permitiu analisar a metodologia usada induzindo assim propostas de melhoria à metodologia seguida na ISA. A *framework* estabelecida irá fazer com que os futuros *developers* possam desenvolver os seus projetos *mobile* e que as presentes equipas possam melhorar o seu método de trabalho. No global, os resultados atingidos excederam as expetativas iniciais para o estágio.

O Cloogy é uma solução em constante evolução, pelo que o seu desenvolvimento para as diversas plataformas móveis continua a ser realizado. A sua visão de *Smart Living* fará com que se continuem a integrar novas funcionalidades tais como dispositivos de segurança e conforto.

Palavras-chave: Cloogy, Windows Phone, processos de desenvolvimento de *software*, aplicação móvel

Abstract

The achievement of this internship had the intention to build skills in order to ISA (Intelligent Sensing Anywhere) developing mobile applications for Windows Phone. The ISA is a technology company which has been developing products and solutions for the energy market, environment, gas and also oil market. This company has been giving their own solutions through mobile platforms such as iOS and Android, having felt the need to gain some skills in Windows Phone platform, due the evolution in Market Share, and so, the company could offer this solution to their clients in a general way.

So that the ISA could gain skills in Windows Phone, the internship aimed the goal of developing a Cloogy application for this platform and simultaneously to create a framework for the Windows Phone development, adapted to the ISA processes.

This internship allows increasing a huge number of tasks which are part of all stages of development software through agile methodology, the Scrum. The requirements were collected and specified, the architecture was defined and all software features were implemented and tested. In order that users could access the Cloogy application, it was published in Windows Phone Store.

The experience gained during the various stages of development in Scrum, essentially in Daily Meeting and Sprint Retrospective ceremonies, was an asset for the design of the framework. This experience allowed to analyze the methodology used thus inducing suggestions for improvements to the methodology followed in the ISA. The framework established will allow the opportunity to all future developers to build mobile projects and that these teams can improve their work methodology. Overall, the results achieved exceeded initial expectations for the internship.

The Cloogy is a solution in constantly evolution, so its development for various mobile platforms continues to be performed. The Cloogy vision of Smart Living will integrate new capabilities, such as security and well-being devices.

Keywords: Cloogy, Windows Phone, software development process, mobile application

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

API – *Application Programming Interface*

BP – *British Petroleum*

C# - *C Sharp*

COM – *Component Object Mod*el

DEIS – Departamento de Engenharia Informática e Sistemas

DLL – *Dynamic-link library*

DSI – Departamento de Sistemas de Informação

EA - *Enterprise Architect*

ERP – *Enterprise Resource Planning*

EUA – Estados Unidos da América

GUI – *Graphical User interface*

HTTP – *Hypertext Transfer Protocol*

IDE – *Integrated Development Environment*

IDI – Investigação Desenvolvimento e Inovação

IP – Impresso Associado

ISA – *Intelligent Sensing Anywhere*

ISEC – Instituto Superior de Engenharia de Coimbra

IT – *Information Technology*

JSON – *JavaScript Object Notation*

LINQ – *Language-Integrated Query*

MI – Manual Interno

MP – Mapa do Processo

MVP - *Model View Presenter*

MVC – *Model View Controller*

MVVM – *Model View ViewModel*

NA – Não aplicado

PAN - *Personal Area Network*

PM – *Product Manager*

PME – Pequena e Média Empresa

PB – *Product Backlog*

PG – Procedimento Geral

PO – Processo Operacional

PO – *Product Owner*

QAPE – *Quality Assurance & Process Enforcement*

REST – *REpresentational State Transfer*

RF – *Radio Frequency*

ROI - *Return on Investment*

SB – *Sprint Backlog*

SM – *Scrum Master*

SGI – Sistema de Gestão Integrado

SQL – *Structured Query Language*

SVN – *Apache Subversion*

UC – *Use Case*

UCP - *Use Case Points*

UI – *User interface*

URL – *Uniform Resource Locator*

WLAN - *Wireless Local Area Network*

WBS – *Work Breakdown Structure* (Estimativa de esforço)

XAML – *eXtensible Application Markup Language*

XAP – *Silverlight Application Package*

XML - *eXtensible Markup Language*

1. Introdução

A ISA (Intelligent Sensing Anywhere) é uma empresa de base tecnológica, com uma experiência de mais de 20 anos em soluções *Machine to Machine* (M2M) "chave na mão", que incluem desde o desenvolvimento de *software* e *hardware*, à prestação de serviços.

No âmbito da unidade curricular de Estágio do Mestrado em Informática e Sistemas, ramo de Desenvolvimento de Software, do Instituto Superior de Engenharia de Coimbra, a realização do presente estágio ocorreu entre o período de Outubro de 2013 a Julho de 2014, nas instalações da ISA sedeadas em Coimbra e no Porto.

Definiu-se o presente trabalho com o intuito de fornecer aos clientes da ISA um maior leque de opções para o acesso e controlo das suas soluções de telemetria e controlo remoto.

A ISA tem vindo a disponibilizar as suas aplicações através de terminais móveis, nomeadamente Android e iPhone, fazendo parte dos seus objetivos de negócio expandir soluções oferecidas também para Windows Phone. É assim objetivo da ISA ganhar competências nesta plataforma, passando a oferecer esta solução aos seus clientes de forma mais generalizada.

1.1. Objetivos do estágio

A realização deste estágio teve como objetivos:

- Desenvolver competências dentro da empresa na Plataforma Windows Phone;
- Criar uma *framework* para o desenvolvimento em Windows Phone;
- Desenvolver uma aplicação Cloogy[1] para Windows Phone.

No desenvolvimento da *framework*, tornou-se ainda fundamental realizar o estudo dos processos relacionados com o desenvolvimento de *software* da ISA para se criar opções de *tailoring*[2] ou mesmo novos processos ou metodologias de trabalho, assim como definir ferramentas e ambientes de desenvolvimento.

[1] Cloogy - solução de gestão energética que permite monitorizar e controlar o consumo de energia de uma casa ou de um escritório. No capítulo 2 encontra-se uma abordagem mais detalhada.

[2] *Tailoring* – Consiste em adaptar um processo, adicionando, eliminando ou modificando elementos e/ou relacionamentos. O processo resultante será o mais apropriado para atingir os objetivos de um projeto.

1.2. Contexto empresarial

A ISA é uma empresa de base tecnológica que tem vindo a desenvolver produtos e soluções para o mercado da energia, ambiente, gás e petróleo. Nos últimos anos, tem apostado fortemente no desenvolvimento de produtos de eficiência energética, nomeadamente sistemas de monitorização e gestão de consumos energéticos para o setor residencial e empresarial. A empresa possui já uma carteira de projetos e clientes nacionais de referência no setor da eficiência energética, destacando-se o BES e ANA Aeroportos, onde os seus sistemas estão instalados. No setor *Oil & Gas* opera igualmente com empresas com um carisma bastante grande, como a Galp, Total, Repsol, BP, Shell, Primagaz e Butagaz.

A ISA tem vindo a afirmar-se como um exemplo de uma PME de sucesso que cresce e cria oportunidades de negócio em novos mercados e novos setores.

Com presença em vários países como Espanha, França, Brasil, EUA e outros no Médio Oriente, a ISA é hoje reconhecida internacionalmente nos setores onde atua. Exemplo disso é a sua recente nomeação pela maior consultora internacional de IT, a Gartner, que distinguiu a empresa na área de aplicações para *Smart Cities*.

O ano de 2012 ficou marcado pela entrada da ISA em bolsa. Em Junho desse ano entrou no Alternext e tornou-se a primeira empresa portuguesa a entrar neste mercado bolsista destinado a PMEs.

A ISA tem apostado na inovação e na excelência como formas de se diferenciar nos mercados. O investimento contínuo na Investigação e Desenvolvimento, aliado à forte qualificação de recursos humanos, representam os principais vetores estratégicos de desenvolvimento desta empresa.

1.3. Calendarização

A proposta inicial do presente estágio incluía um programa de trabalhos assim como a sua calendarização. Esta proposta inicial consistia em sete tarefas, nomeadamente:

- T1 – Estudos preliminares – Análise de aplicações móveis desenvolvidas. Análise dos processos de desenvolvimento da ISA. Estudo de metodologias de desenvolvimento e ferramentas associadas a aplicações móveis e Windows Phone;

- T2 – Definição base de uma *framework* de desenvolvimento para Windows Phone – Criação das linhas orientadoras de desenvolvimento para controlos visuais da aplicação Cloogy em ambiente Windows Phone;
- T3 – Levantamento de Requisitos – Familiarização com a tecnologia Cloogy. Documentação dos requisitos e funcionalidades para uma aplicação móvel Cloogy em ambiente Windows Phone. Elaboração das especificações técnicas dos módulos a desenvolver;
- T4 – Desenvolvimento – Desenvolvimento da aplicação móvel;
- T5 – Testes – Avaliação do desempenho da aplicação móvel em ambiente de laboratório, com telemóveis da empresa. Análise e introdução de possíveis correções ou alterações que melhorem o desempenho da aplicação. Realização de testes, em ambiente real. Comparação com aplicações desenvolvidas para sistemas operativos Android e iOS;
- T6 – Revisão e escrita dos detalhes da *framework* de desenvolvimento de aplicações Windows Phone – Definição de opções de *tailoring* para os processos da ISA. Definição de ferramentas e metodologias; Divulgação e formação interna;
- T7 – Escrita do relatório de estágio.

O escalonamento das tarefas mencionadas anteriormente pode ser visualizado na Tabela 1:

	Meses										
Tarefas		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
T1											
T2											
T3											
T4											
T5											
T6											
T7											

Tabela 1 Escalonamento das tarefas definidas na proposta inicial de estágio

Uma vez que o presente estágio foi executado sob uma metodologia ágil, houve a necessidade de ajustar o planeamento descrito anteriormente, pois a execução de tarefas foi feita em várias

iterações, tipicamente duas iterações por mês. A Tabela 2 apresenta a calendarização efetiva dos trabalhos.

	Meses									
Tarefas	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**
T1										
T2										
T3										
T4										
T5										
T6										
T7										

Tabela 2 Calendarização do trabalho executado no estágio

Todo o trabalho executado ao longo do presente estágio foi sendo continuamente validado. Os relatórios e estudos de processos de desenvolvimento de *software* foram validados ao longo do estágio em reuniões com os orientadores de estágio.

Como apresentado na Tabela 3, as tarefas T3, T4 e T5, correspondentes ao desenvolvimento da aplicação, foram validadas pelas demostrações efetuadas no final de cada iteração.

Final de iteração	**Trabalho demostrado**
5/12/2013	• Login da aplicação
20/12/2013	• Estado atual da aplicação
30/01/2014	• Tomadas inteligentes e seu funcionamento • Eletricidade • Gráficos dos consumos energéticos
20/02/2014	• Gráficos (movimento *flick* e consumos em tempo real)
6/03/2014	• Novo *layout* da aplicação • Novo medidor circular • *Widget* do objetivo mensal

3/04/2014	• Novo *layout* dos ecrãs de login, recuperação de palavra-passe e criação de uma nova conta • *Tooltip* dos gráficos e novas *labels* de comparação • Edição de tomadas inteligentes • Agendamento de tomadas inteligentes
16/04/2014	• Novo ecrã de definições • Funcionalidade de *logout* • *Wonky messages* • Novo *widget* do objetivo mensal • Nova interação da escolha dos dias de repetição na criação de uma regra de atuação • Ajustes na aplicação
6/05/2014	• Novo mecanismo dos consumos em tempo real • Nova funcionalidade inserida na aplicação, permitir ao utilizador iniciar sessão automaticamente. • Gráficos dos consumos em tempo real

Tabela 3 Demonstrações da aplicação no fim de cada iteração

1.4. Estrutura do relatório

O presente relatório encontra-se dividido em oito capítulos principais. O primeiro capítulo tem como objetivo apresentar o estágio e descrever os seus objetivos, descrever a entidade acolhedora do estagiário e apresentar as discrepâncias do planeamento original das tarefas oriundas da proposta inicial de estágio, com o resultado final da sua execução.

O segundo capítulo introduz o Cloogy, descreve o seu modo de funcionamento e apresenta as aplicações móveis existentes.

O terceiro capítulo tem como intuito estudar o desenvolvimento de aplicações na ISA, ou seja, é apresentado um estudo dos processos de desenvolvimento de *software* da ISA e os processos usados no projeto Cloogy.

O quarto capítulo apresenta a aplicação desenvolvida em Windows Phone 8 para o Cloogy. Indica as ferramentas, tecnologias, arquitetura adotada e todas as estratégias necessárias para o seu desenvolvimento.

O quinto capítulo tem como intuito apresentar melhorias à metodologia usada pelas equipas de desenvolvimento *mobile* da ISA.

O sexto capítulo, denominado por *Framework* de desenvolvimento para Windows Phone 8, tem como objetivo estabelecer linhas orientadoras para o desenvolvimento de aplicações para a plataforma Windows Phone 8 na ISA. Neste capítulo é apresentado uma metodologia de trabalho, linhas orientadoras de desenvolvimento, ferramentas a serem usadas e *code conventions*.

As conclusões do trabalho realizado situam-se no capítulo 7.

As referências de todo o material necessário para a realização deste relatório podem ser consultadas no capítulo 8.

2. O Cloogy

O Cloogy é uma solução de gestão energética que permite monitorizar e controlar o consumo de energia de uma casa ou de um escritório. Esta solução consiste num sistema integrado que combina dispositivos de recolha de dados (sensores) com plataformas de visualização e controlo, disponíveis através de um computador, *smartphone*, *tablet* ou Monitor Cloogy.

Desta forma, o Cloogy permite detetar ineficiências numa casa ou escritório, permitindo ao utilizador poupar energia, através da monitorização do consumo dos seus equipamentos elétricos e do controlo do seu horário de funcionamento, através de tomadas inteligentes, permitindo ligá-los e desliga-los à distância.

A solução Cloogy permite também aos seus utilizadores saber os seus consumos globais além dos consumos de cada uma das tomadas inteligentes, permitindo assim o contributo para uma vida mais sustentável, reduzindo os desperdícios e custos desnecessários.

2.1. Modo de funcionamento

Como apresentado na Figura 1, o sensor (pinça amperimétrica) instalado no contador de eletricidade permite recolher os consumos globais de uma casa ou escritório, que posteriormente serão enviados para o Concentrador via um Transmissor. As Tomadas Inteligentes, por sua vez, também enviam informação sobre os equipamentos ligados a estas.

Todos os dados recolhidos são enviados para o Concentrador, que remete para as várias plataformas de monitorização, quer através da rede local dos dispositivos, no caso do Monitor Cloogy, quer através da internet, para as restantes plataformas. A partir de um Concentrador é possível monitorizar e controlar até onze tomadas em simultâneo.

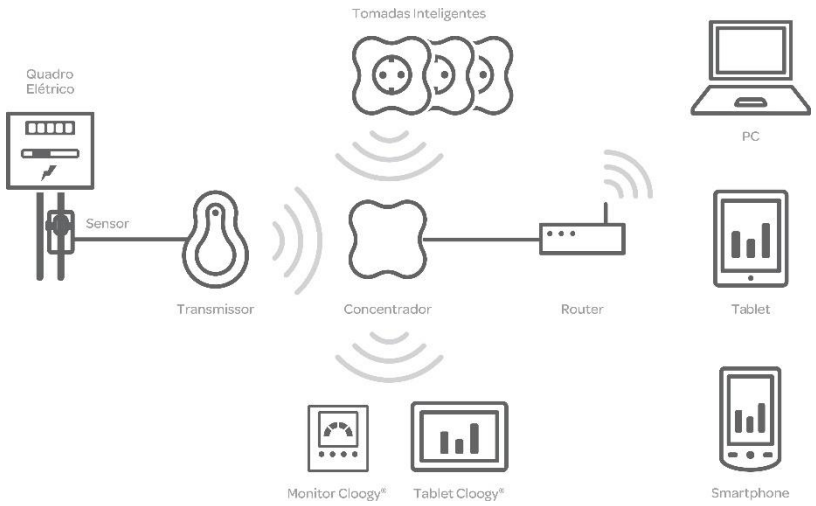


Figura 1 Modo de funcionamento do Cloogy [1]

Os equipamentos comunicam entre si via protocolo RF ZigBee [2]. Este protocolo foi concebido em 1998, sendo baseado no padrão IEEE 802.15. Possui uma relação direta com o padrão IEEE 802.15.4, comparável com as redes Bluetooth [3] baseados no padrão IEEE802.15.1. Ambos os protocolos são usados em PANs (*Personal Area Network*), redes pessoais baseadas no padrão IEEE 802.15, ao contrário da WLAN (*Wireless Local Area Network*) que é baseada no padrão IEEE 802.11.

O protocolo RF ZigBee é usado por aplicações que requerem baixa taxa de dados e consumos. Desta forma, os equipamentos podem ser alimentados por pilhas para estabelecerem a comunicação, evitando a necessidade de serem ligados à corrente elétrica, que é o caso do Transmissor e do Monitor Cloogy.

Através do acesso à internet dos utilizadores, os dados são recolhidos numa plataforma, que disponibiliza além desses mesmos dados, as funcionalidades que os equipamentos possuem. Esta disponibilização é feita através de uma API REST que é utilizada pelas aplicações desenvolvidas para computador (aplicação *web*), *smartphone* e *tablet*. Desta forma as aplicações têm acesso a toda a funcionalidade e dados a qualquer momento.

2.2. Aplicações móveis já existentes

Antes da realização deste projeto, o Cloogy era suportado por duas aplicações para sistemas móveis, nas plataformas iOS e Android.

A aplicação para iOS é compatível com iPhone e iPad, podendo ser descarregadas na iTunes Store. A aplicação para Android está disponível no Google Play Store para qualquer dispositivo com Android superior à versão 4.0.

Independentemente da aplicação escolhida, é necessário uma conta Cloogy para poder interagir com estas aplicações.

Estas aplicações complementam e aumentam a oferta da solução Cloogy, permitindo assim aos utilizadores acederem remotamente aos seus dados de consumo através de um dispositivo móvel, tendo como objetivo disponibilizar um conjunto de funcionalidades tais como:

- Monitorização de consumos (históricos e em tempo real);
- Consulta de indicadores de consumo (desempenho, média diária, previsões baseadas nos padrões de consumo, entre outros);
- Visualização da pegada ecológica;
- Partilha de desempenho através das redes sociais;
- Controlo e agendamento dos períodos de funcionamento dos seus equipamentos elétricos (permitindo ligar e desligar os equipamentos à distância).

O desenvolvimento destas aplicações é da responsabilidade da ISA, desde o *hardware*, *firmware* e *software*, quer do lado do servidor, quer do lado do cliente.

3. O desenvolvimento de aplicações na ISA

A ISA, tal como a generalidade das empresas, necessita de um processo de desenvolvimento de *software* que descreva um conjunto de atividades a executar, de uma forma coordenada, para que o *software* produzido tenha a qualidade requerida.

Uma vez que o presente estágio tem como objetivos a criação de uma *framework* de trabalho e desenvolvimento de uma aplicação em Windows Phone, este capítulo aborda o estado atual dos processos de desenvolvimento de *software* da ISA, bem como os métodos e ferramentas usadas no desenvolvimento de projetos de aplicações móveis, de forma a serem analisados e compreendidos para que seja possível introduzir novas políticas de trabalho.

3.1. Estado dos processos de desenvolvimento de *software*

A ISA possui um Sistema de Gestão Integrado (SGI) de Qualidade e Inovação que tem como finalidade servir de linha orientadora para os seus colaboradores desenvolverem as suas ações colocando sempre o Cliente no centro das preocupações, bem como garantir a melhoria contínua dos seus processos de trabalho e a satisfação dos colaboradores. O SGI está acessível a todos da empresa e pode ser acedido na rede da ISA através da *Intranet*.

O SGI descreve os meios adotados na ISA que permitem assegurar a Qualidade dos Produtos e Serviços fornecidos, tendo como resultado a conquista do nível 2 do CMMI-DEV, um modelo internacionalmente conceituado que reúne as melhores práticas para o desenvolvimento de *software*. De acordo com os referenciais NP EN ISO 9001 (Requisitos para Sistemas de Gestão da Qualidade) e NP 4457 (Requisitos do Sistema de Gestão da IDI (Investigação, Desenvolvimento e Inovação)), o sistema disponibiliza todo o material necessário para o cumprimento dos procedimentos da organização, definindo quatro níveis de documentação, como apresentado na Figura 2.

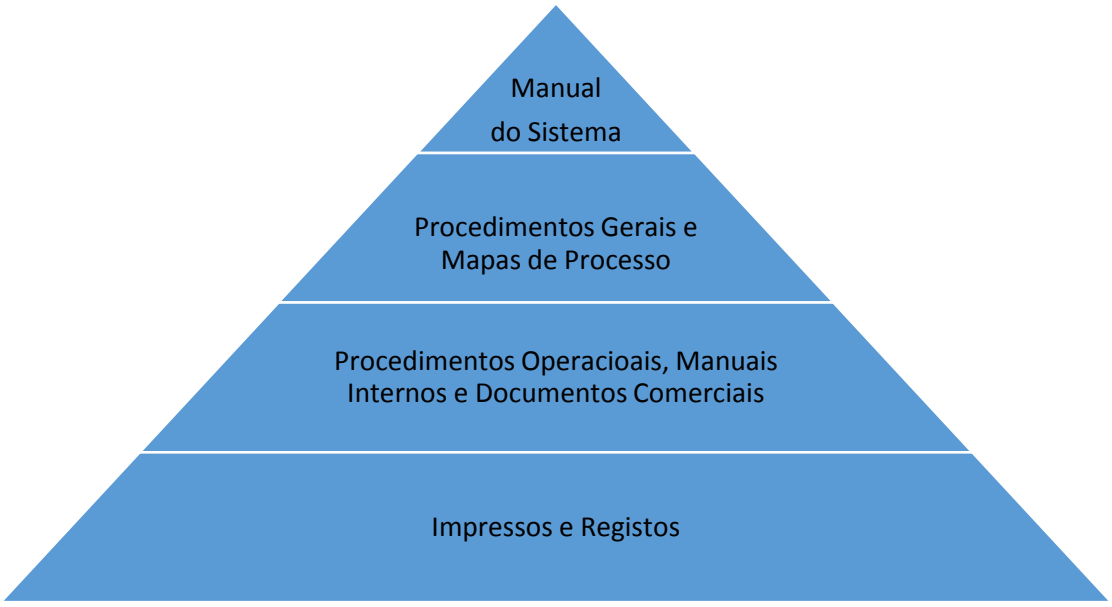


Figura 2 Documentação de suporte do SGI

O topo da hierarquia dos níveis de documentação é constituído pelo Manual do Sistema. É um documento que tem como objetivos apresentar a ISA aos seus colaboradores, indicando a sua visão, valores, estratégias e vantagens competitivas, apresentar a estrutura organizativa da empresa e o SGI.

O nível seguinte é constituído por Procedimentos Gerais e Mapas de Processo. Os Procedimentos Gerais, também tratados como PG, descrevem as atividades necessárias para a execução de tarefas que sejam comuns nas diversas áreas da empresa. Todos os documentos que descrevem os Procedimentos Gerais têm a sigla PG no início do nome do ficheiro correspondente.

Os Mapas de Processo são documentos que descrevem de uma maneira geral os processos usados na empresa, de forma a indicar as atividades associadas a uma determinada área de negócio. Estes Mapas de Processo indicam as entradas e saídas do processo analisado, a relação com outros processos, um organograma do processo e respetivas descrições de regras e atividades. O nome dos ficheiros destes documentos iniciam sempre por MP.

O terceiro nível da hierarquia é constituído por Procedimentos Operacionais, Manuais Internos e Documentos Comerciais. Os Procedimentos Operacionais, também mencionados como PO, são documentos que têm como objetivo fazer uma descrição detalhada de todas as operações necessárias para realizar uma determinada atividade de um processo de uma área de negócio da empresa.

Os Manuais Internos são documentos que descrevem e suportam técnicas usadas para realizar determinadas tarefas. Todos os ficheiros que sejam Manuais Internos têm a sigla MI no início do nome do ficheiro.

Na base do nível hierárquico dos níveis de documentação, encontram-se os Impressos Associados e Registos. Estes documentos são *templates* a serem preenchidos pelas entidades que executam determinadas atividades. Estes impressos também são denominados por IP.

Como apresentado na Figura 3, o SGI é constituído por seis processos principais, nomeadamente:

- Planeamento Estratégico;
- Gestão da Qualidade;
- Desenvolvimento de Negócio;
- Conceção e Desenvolvimento;
- Compras e Subcontratação;
- Inovação.

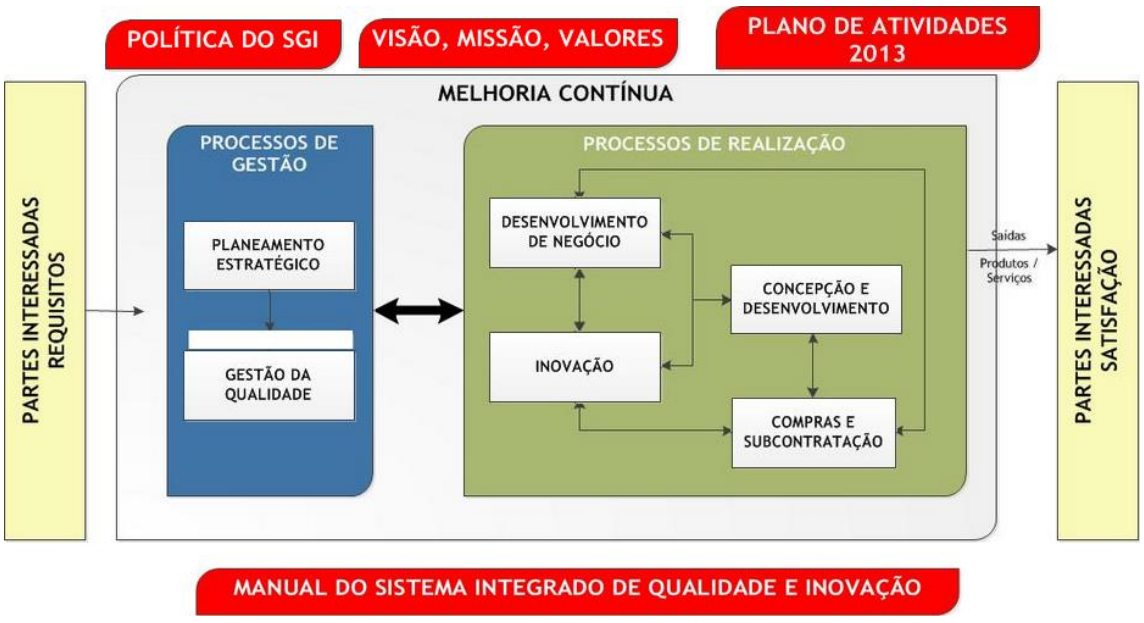


Figura 3 Mapa de processos do Sistema de Gestão Integrado[3]

[3] Figura retirada do Sistema de Gestão Integrado da ISA.

O processo Planeamento Estratégico tem como objetivos definir/rever a política do SGI, elaborar o planeamento anual de manutenção e de formação e efetuar orçamentos previsionais.

O processo denominado por Gestão da Qualidade, tem como objetivos fazer a gestão de documentos, dados e registos, monitorizar os processos existentes na ISA, controlar as Não Conformidades, executar auditorias ao SGI e controlar os equipamentos de medição e de monitorização.

O processo Conceção e Desenvolvimento é o processo responsável por todos os procedimentos necessários para o desenvolvimento de *software* e *hardware*.

O processo Compras e Subcontratação tem como objetivo gerir as compras necessárias para o desenvolvimento do *software* e *hardware*.

O processo denominado por Inovação tem como objetivos desenvolver produtos e serviços inovadores, avaliar resultados e criar oportunidades de melhoria.

A ISA tem implementados um conjunto de procedimentos, tal como apresentado na Tabela 4, para que cada processo consiga descrever todas as ações necessárias à execução das respetivas atividades. Desta forma garante-se que os *outputs* dos processos são assegurados e que são o *input* do processo subsequente, clarificando assim também as formas de interação necessárias:

Processo	**Procedimentos**
Planeamento Estratégico	• Definição/Revisão da política do SGI • Balanço do ano transato • Definição dos objetivos do SGI • Acompanhamento e controlo • Determinação e gestão de recursos
Gestão da Qualidade	• Gestão de documentos, dados e registos • Monitorização dos processos • Controlo das não conformidades e produto não conforme • Gestão das ações • Gestão das auditorias ao SGI • Controlo dos equipamentos de monitorização e medição
Desenvolvimento de Negócio	• Gestão de produto • Comercial

	• Gestão de projetos • Suporte ao cliente
Conceção e Desenvolvimento	• Gestão e Engenharia de Projeto • Gestão de Configurações • Gestão de Alterações de C&D • Gestão de Subcontratação de C&D • Gestão de Indicadores de C&D • Auditorias internas da Qualidade
Compras e Subcontratação	• Seleção e avaliação de fornecedores • Compras • Subcontratação da realização do produto • Inspeção ao produto
Inovação	• Gestão das interfaces • Gestão do conhecimento • Estimular a criatividade • Gestão das ideias • Gestão de projetos • Avaliação de resultados

Tabela 4 Processos do SGI e respetivos procedimentos

Uma vez que se pretende um estudo sobre os processos de desenvolvimento de *software* da ISA, de modo a enquadrar o trabalho a desenvolver durante o estágio, será estudado apenas o processo de Conceção e Desenvolvimento (C&D).

3.2. Processo Conceção e Desenvolvimento (C&D)

O processo de Conceção e desenvolvimento, representado na Figura 4, é um processo genérico da ISA, que tem como objetivo servir de linha orientadora para as equipas de desenvolvimento de *software* e *hardware*. No entanto, como o presente estágio está dirigido ao desenvolvimento de *software*, o estudo do presente processo irá consistir apenas nas linhas orientadoras de *software*.

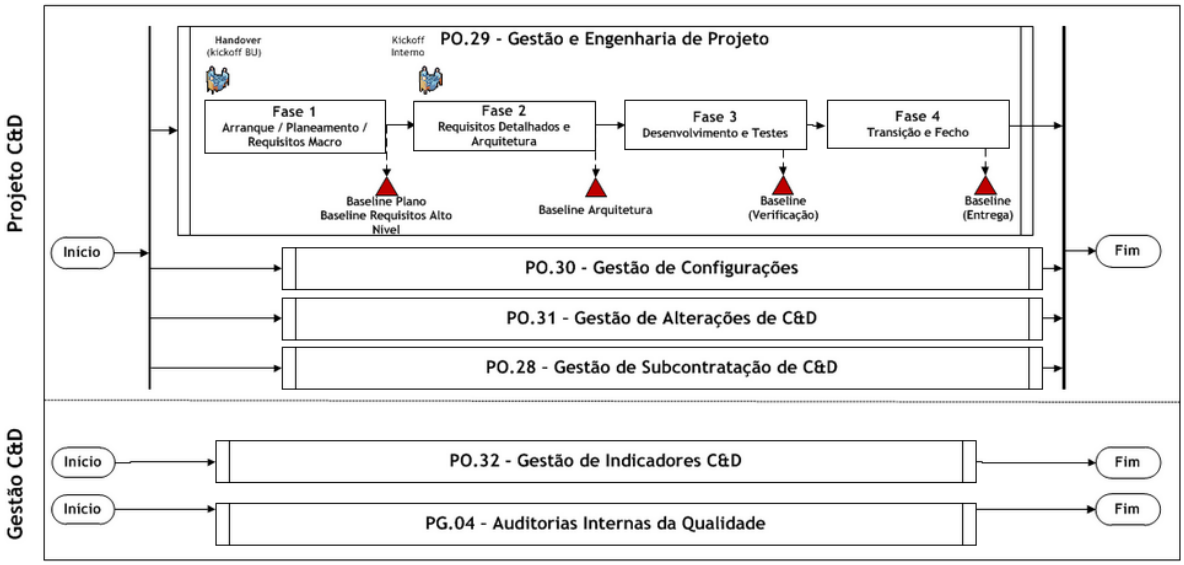


Figura 4 Atividades do processo Conceção & Desenvolvimento[4]

A Tabela 5 retrata as entradas e saídas do processo C&D, assim como os processos relacionados com este:

	Entradas	**Saídas**
Processos	• Planeamento Estratégico • Inovação • Desenvolvimento de Negócio	• Desenvolvimento de Negócio • Inovação • Compras e Subcontratação
Artefactos	• Requisitos do Cliente/Caderno de Encargos (quando aplicável) • Requisitos legais (quando aplicável) • *Roadmap* Tecnológico • Requisitos do Produto • Pedido de alteração • Proposta Comercial Adjudicada	• Produtos desenvolvidos, testados e documentados

Tabela 5 Entradas e saídas do processo C&D

Este processo envolve várias atividades que ocorrem em simultâneo durante o ciclo de vida de um projeto de *software*, sustentadas por um procedimento geral (PG), nomeadamente o procedimento denominado por Auditorias Internas da Qualidade, e cinco procedimentos operacionais (PO), constituídos pelos restantes procedimentos já retratados na Figura 4.

[4] Figura retirada do Sistema de Gestão Integrado da ISA.

No decorrer do processo C&D existem duas fases distintas a decorrer em paralelo, nomeadamente a fase de projeto e a fase de gestão. A fase de projeto é constituída por procedimentos operacionais que descrevem as ações necessárias para a execução das atividades necessárias no desenvolvimento de *software*. A fase de Gestão tem como objetivo interligar o processo de Gestão da Qualidade, para que sejam exercidas atividades de controlo e auditoria, ou seja, controlo de documentos, dados, registos e não conformidades.

De entre os procedimentos operacionais do processo C&D, será de seguida descrito o de Gestão e Engenharia de Projeto, pois é este que se refere propriamente ao desenvolvimento de *software*.

3.2.1. PO.29 Gestão e Engenharia de Projeto

O procedimento operacional Gestão e Engenharia de Projeto é constituído por quatro fases bem definidas, nomeadamente:

- Fase 1 – Arranque / Planeamento / Requisitos Macro;
- Fase 2 – Requisitos Detalhados e Arquitetura;
- Fase 3 – Desenvolvimento e Testes;
- Fase 4 – Transição e Fecho.

Em cada uma das fases deste procedimento operacional, estão estabelecidas *milestones*[5], ou seja, num determinado ponto temporal de cada fase, são reportados determinados artefactos. A Tabela 6 apresenta as várias *milestones* de cada fase:

[5] *Milestone* – Evento ocorrido num projeto, usado para dar visibilidade do progresso de metas atingidas pré-definidas no projeto. O não cumprimento de uma *milestone* indica uma má realização do projeto.

Fase	*Milestones*
1	• *Baseline*[6] Plano • *Baseline* Requisitos Alto Nível
2	• *Baseline* Plano (caso seja preciso replanear) • *Baseline* Arquitetura
3	• *Baseline* Verificação
4	• *Baseline* Entrega

Tabela 6 Milestones das diversas fases do PO Gestão e Engenharia de Projeto

3.2.2. Fase 1 – Arranque / Planeamento / Requisitos Macro

Como apresentado na Figura 5, o procedimento operacional Gestão e Engenharia de Projeto é constituído por 14 atividades, onde dez correspondem a Gestão de Desenvolvimento e as restantes quatro pertencem à Engenharia do projeto.

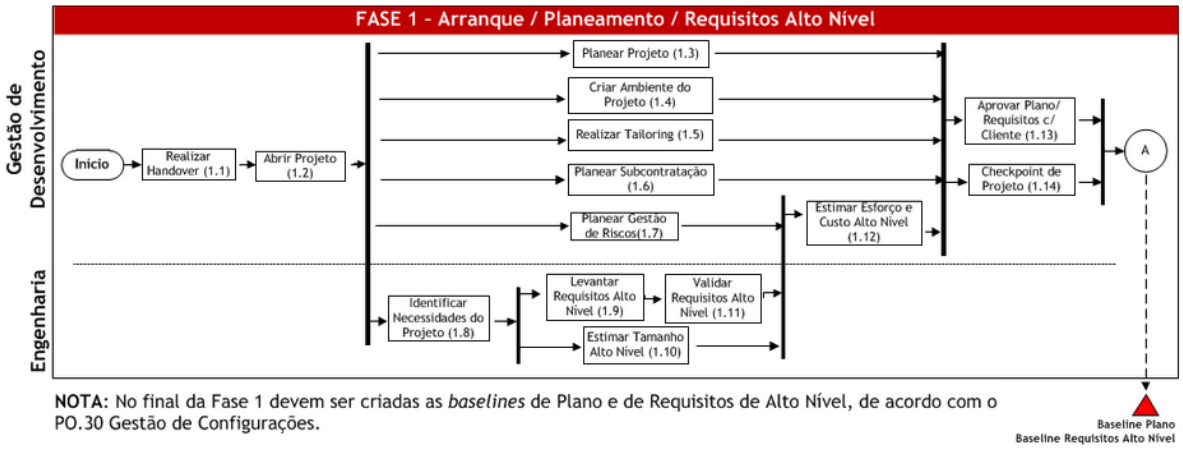


Figura 5 Fluxograma da Fase 1 do PO Gestão e Engenharia de Projeto[7]

As atividades relacionadas com a Gestão de Desenvolvimento encontram-se descritas na Tabela 7:

[6] *Baseline* – Descrição dos atributos de um projeto, num determinado ponto temporal, que serve como base para ser medido e comparado.

[7] Figura retirada do Sistema de Gestão Integrado da ISA.

Atividade	Objetivo
Realizar Handover (1.1)	• Apresentação dos objetivos • Contextualização do projeto e sua utilização pelo cliente • Apresentação do *roadmap* • Apresentação de prazos e *budget* inerentes ao projeto • Apresentação de restrições do projeto
Abrir Projeto (1.2)	• O Departamento de Sistemas de Informação (DSI) deverá criar o repositório e preparar a ferramenta de *Issue Tracking*[8] • Deve ser feita a passagem de informação ao *Quality Assurance & Process Enforcement* (QAPE) a dar conhecimento do arranque do projeto
Planear Projeto (1.3)	• Criar uma primeira versão do planeamento que deve incluir atividades de recolha de necessidades e/ou requisitos do cliente • Descrever o modo de desenvolvimento do projeto, considerando indicadores recolhidos ao longo do mesmo • Planear o projeto com base na *Work Breakdown Structure*[9] (WBS) estabelecida • Definir uma estratégia de gestão de configurações e planear as *baselines* • Enviar o Plano de Projeto C&D ao Departamento de Sistemas de Informação para dar conhecimento dos ambientes de desenvolvimento e para que salvaguardem as cópias de *software* correspondentes
Criar Ambiente do Projeto (1.4)	• Instalar/Configurar as ferramentas necessárias ao projeto • Criar os ambientes auxiliares ao desenvolvimento imprescindíveis ao projeto
Realizar Tailoring (1.5)	• Realizar *tailoring* do processo C&D *standard* da organização
Planear Subcontratação (1.6)	• Planear as subcontratações de acordo com o Procedimento Operacional Gestão de Subcontratações
Planear Gestão de Riscos (1.7)	• Elaborar um plano de gestão de riscos • Realizar uma reunião com os respetivos *stakeholders*[10]
Estimar Esforço e Custo Alto Nível (1.12)	• Estabelecer estimativas de esforço e de custo para cada WBS do projeto

Aprovar Plano/Requisitos com o Cliente (1.13)	• Apresentar o Plano do Projeto ao cliente • Obter aceitação do Plano do Projeto • Aprovar com o Cliente os requisitos do sistema
Checkpoint de Projeto (1.14)	• Analisar e divulgar o Plano do Projeto C&D • Apresentar e discutir os requisitos do sistema • Discutir Riscos do projeto

Tabela 7 Objetivos de cada atividade relacionadas com a Gestão de Desenvolvimento da Fase 1 do PO Gestão e Engenharia de Projeto

As atividades relacionadas com a Engenharia do projeto encontram-se descritas na Tabela 8:

[8] *Issue Tracking* – *software* que permite gerir e monitorizar incidentes (problemas) ocorridos num sistema, reportados por clientes ou colaboradores.

[9] WBS – processo de subdivisão do projeto em vários componentes menores, de forma a facilitar o seu desenvolvimento e a sua gestão.

[10] *Stakeholder* – termo usado para referenciar um indivíduo, equipa, organização ou qualquer outra entidade que tenha interesse no desenvolvimento do projeto.

Atividade	Objetivo
Identificar Necessidades do Projeto (1.8)	• Identificar as fontes dos requisitos, quer seja a equipa ou o próprio cliente • Identificar estratégias e técnicas de forma a levantar as necessidades e definir ações para recolha dos requisitos • Planear, registar e controlar as ações de recolha de necessidades • Compreender o pedido do cliente e perceber quais são as suas necessidades reais
Levantar Requisitos Alto Nível (1.9)	• Planear as atividades para o levantamento de requisitos de alto nível • Executar o levantamento de requisitos de alto nível e documentá-los • Criar a matriz de rastreabilidade entre Necessidades e Requisitos de Alto Nível
Estimar Tamanho Alto Nível (1.10)	• Estimar o tamanho dos requisitos e o tamanho total do projeto • Ajustar os fatores técnicos e ambientais
Validar Requisitos Alto Nível (1.11)	• Validar com os fornecedores de requisitos o levantamento de requisitos de alto nível

Tabela 8 Objetivos de cada atividade relacionadas com Engenharia da Fase 1 do PO Gestão e Engenharia de Projeto

3.2.3. Fase 2 – Requisitos Detalhados e Arquitetura

Como apresentado na Figura 6, a segunda fase do procedimento operacional Gestão e Engenharia de Projeto é constituída por nove atividades, sendo quatro delas comuns nas fases seguintes, nomeadamente as atividades Monitorizar e Controlar Projeto (2.1), Monitorizar Subcontratados (2.2), Gerir Riscos (2.3) e Reportar Progresso e Indicadores (2.4).

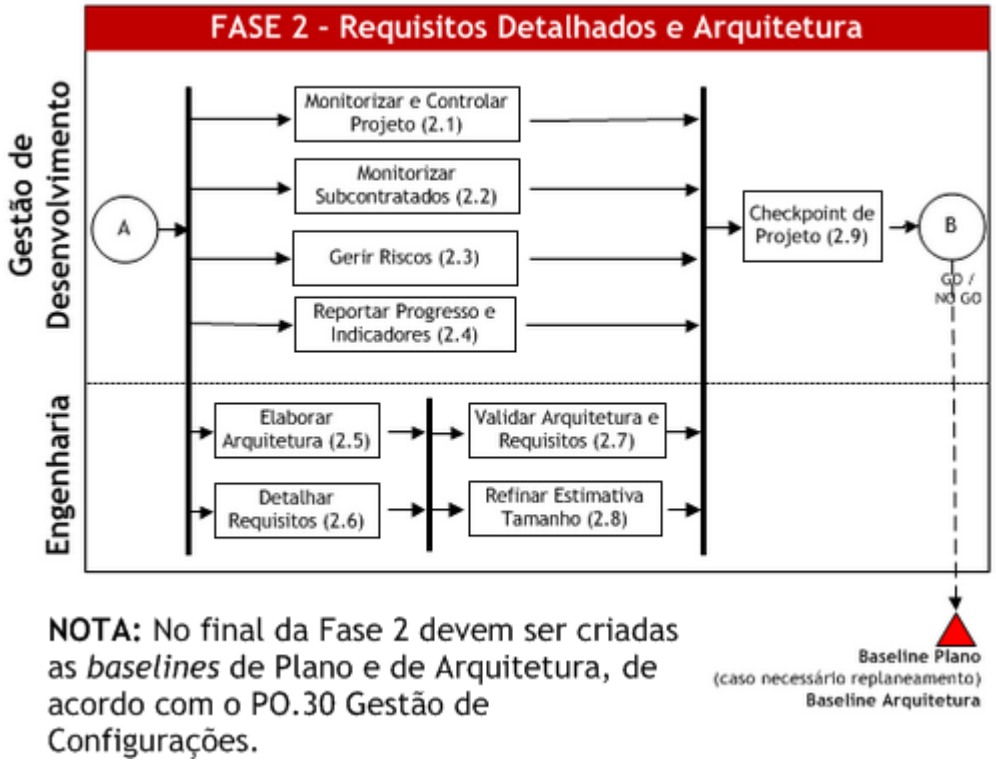


Figura 6 Fluxograma da Fase 2 do PO Gestão e Engenharia de Projeto[11]

As atividades relacionadas com a Gestão de Desenvolvimento encontram-se descritas na Tabela 9:

[11] Figura retirada do Sistema de Gestão Integrado da ISA.

Atividade	Objetivo
Monitorizar e Controlar Projeto (2.1)	• Realizar reuniões mensais de acompanhamento do projeto para discutir e acompanhar o andamento do mesmo • Verificar a execução de tarefas de acordo com o previsto e replanear se necessário • Garantir o cumprimento de *milestones* e entregáveis • Verificar se a equipa e os *Stakeholders* se mantêm alinhados com o plano • Registar e acompanhar até ao fecho as ações originadas a partir das questões críticas/problemas, mantendo o registo periódico das revisões e estado destas ações • Registar as lições aprendidas ao longo do projeto • Garantir o seguimento da estratégia de Gestão de Configurações e as *Baselines* estabelecidas de acordo com o planeado • Garantir a realização de eventuais auditorias previstas no planeamento do projeto • Recolher e analisar os indicadores do projeto, conforme o que foi definido no Plano do Projeto e no documento Caracterização de Indicadores C&D
Monitorizar Subcontratados (2.2)	• Monitorizar os subcontratados, de acordo com o Procedimento Operacional denominado por PO.28 Gestão de Subcontratação de C&D
Gerir Riscos (2.3)	• Identificar a ocorrência de novos riscos • Rever as probabilidades e impactos dos riscos identificados • Manter atualizado o estado dos riscos
Reportar Progresso e Indicadores (2.4)	• Elaborar o relatório de projeto, para que possa eventualmente contribuir para o *Book*[12] da área onde se desenrola o processo • Apresentar e/ou enviar o *Book* à equipa e aos *Stakeholders* do projeto, conforme o descrito no plano de comunicações.

[12] O *Book* é um modelo de reporte interno que suporta a tomada de decisões e que dá a conhecer a situação real da ISA. Cada departamento da ISA contém um *Book*.

Checkpoint de Projeto (2.9)	• Verificar as alterações das estimativas de tamanho do projeto • Rever as estimativas de esforço e custo do projeto • Verificar o impacto das novas estimativas e tomar as ações necessárias para as alinhar com o Planeamento atual • Dar a conhecer à equipa a atualização do plano e avaliar a viabilidade do projeto prosseguir

Tabela 9 Objetivos de cada atividade relacionadas com Gestão de Desenvolvimento da Fase 2 do PO Gestão e Engenharia de Projeto

As atividades relacionadas com a Engenharia do projeto encontram-se descritas na Tabela 10:

Atividade	Objetivo
Elaborar Arquitetura (2.5)	• Construir/Refinar a arquitetura do projeto • Rastrear componentes de arquitetura com requisitos de alto nível • Garantir o alinhamento da elaboração da arquitetura com as atividades de detalhe dos requisitos.
Detalhar Requisitos (2.6)	• Elaborar casos de uso • Elaborar restrições e atributos de qualidade • Rastrear os Requisitos de Alto Nível com os Detalhados • Garantir o alinhamento do detalhe dos requisitos com as atividades de elaboração da arquitetura
Validar Arquitetura e Requisitos (2.7)	• Garantir a validação da arquitetura, assim como a validação dos requisitos detalhados • Recorrer à colaboração das pessoas necessárias de modo a garantir a qualidade das soluções apresentadas
Refinar Estimativa Tamanho (2.8)	• Refinar as estimativas de tamanho dos requisitos e do tamanho total do projeto

Tabela 10 Objetivos de cada atividade relacionadas com Engenharia da Fase 2 do PO Gestão e Engenharia de Projeto

3.2.4. Fase 3 – Desenvolvimento e Testes

Como apresentado na Figura 7, a terceira fase do procedimento operacional Gestão e Engenharia de Projeto é constituída por sete atividades, sendo quatro delas as mesmas

atividades efetuadas na fase anterior, Requisitos Detalhados e Arquitetura, nomeadamente as atividades Monitorizar e Controlar o Projeto (2.1), Monitorizar Subcontratados (2.2), Gerir Riscos (2.3) e Reportar Progresso e Indicadores (2.4), por esta razão não serão referidas nesta terceira fase.

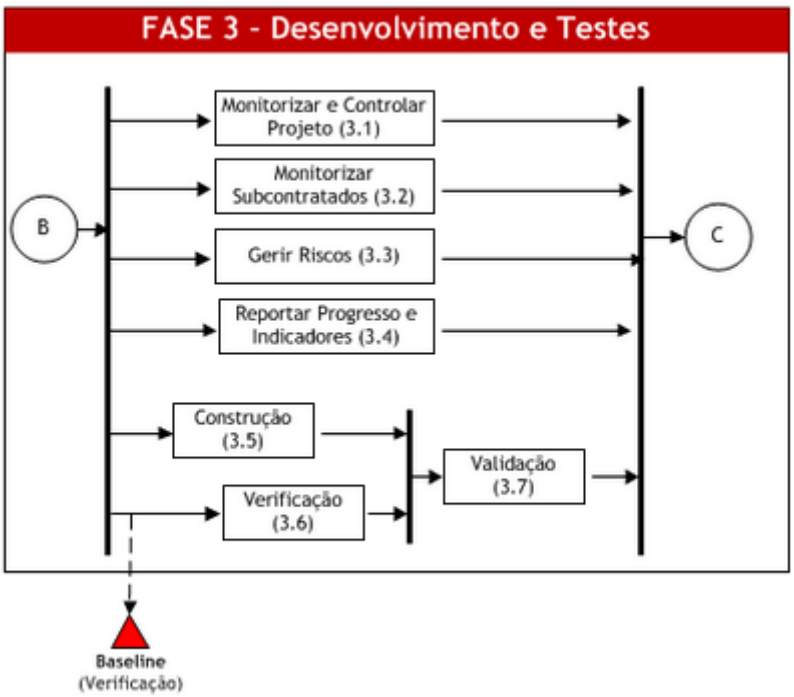


Figura 7 Fluxograma da Fase 3 do PO Gestão e Engenharia de Projeto[13]

As atividades relacionadas com a Engenharia do projeto encontram-se descritas na Tabela 11:

[13] Figura retirada do Sistema de Gestão Integrado da ISA.

Atividade	Objetivo
Construção (3.5)	• Estabelecer *Design* do *software* • Implementar o *software* • Desenvolver protótipos • Elaborar documentação de suporte do *software* • Elaborar plano de testes • Efetuar revisão do projeto
Verificação (3.6)	• Assegurar a criação de uma *baseline* antes de cada execução de testes • Garantir a execução do plano de testes do *software* e o reporte das falhas detetadas na ferramenta de *Issue Tracking* • Controlar e monitorizar a correção das falhas • Garantir a revisão dos manuais elaborados • Verificar todos os artefactos produzidos
Validação (3.7)	• Demostrar os artefactos produzidos ao cliente • Instalar o pacote em ambiente de pré-produção ou simulado • Instalar uma versão piloto em ambiente real, simulado ou Produção com sucesso na fase de testes • Validar produto

Tabela 11 Objetivos de cada atividade relacionadas com Engenharia da Fase 3 do PO Gestão e Engenharia de Projeto

3.2.5. Fase 4 – Transição e Fecho

Como apresentado na Figura 8, tal como na fase anterior, Desenvolvimento e Testes, as atividades, Monitorizar e Controlar o Projeto (2.1), Monitorizar Subcontratados (2.2), Gerir Riscos (2.3) e Reportar Progresso e Indicadores (2.4), não serão referidas na presenta fase denominada por Transição e Fecho.

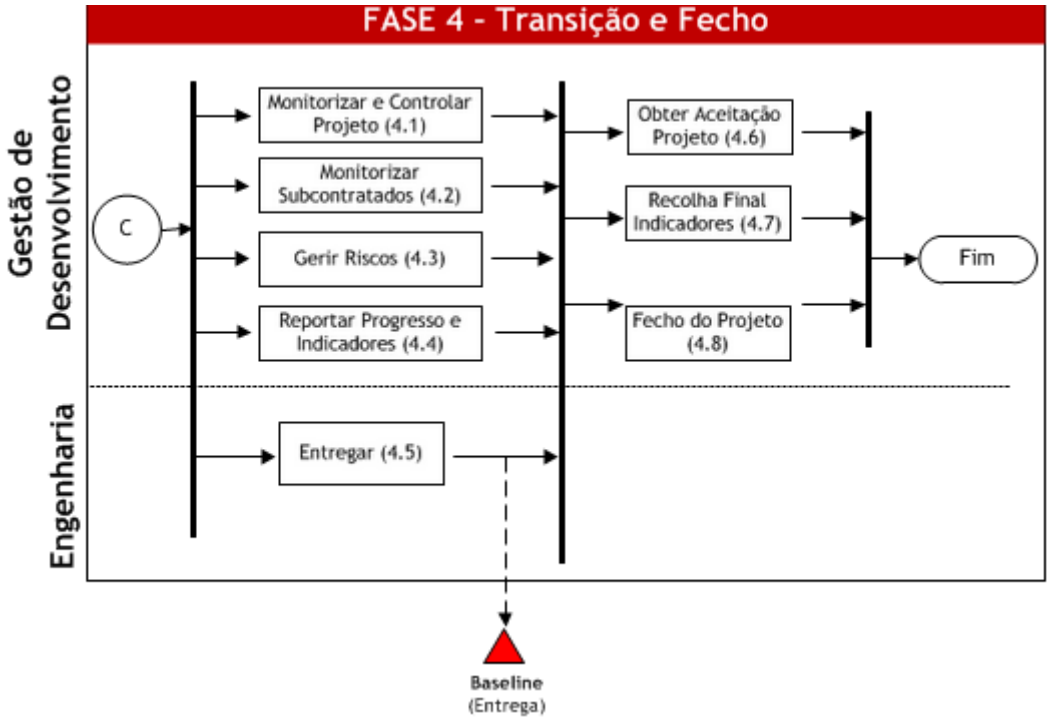


Figura 8 Fluxograma da Fase 4 do PO Gestão e Engenharia de Projeto[14]

A Tabela 12 tem como intuito de apresentar as restantes atividades relacionadas com a Gestão de Desenvolvimento, que não são executadas nas fases anteriores do presente Procedimento Operacional.

[14] Figura retirada do Sistema de Gestão Integrado da ISA.

Atividade	Objetivo
Obter Aceitação Projeto (4.6)	• Efetuar uma reunião com o cliente para aceitação do projeto
Recolha Final Indicadores (4.7)	• Recolher os indicadores finais do projeto • Atualizar documentação de monitorização e rastreabilidade
Fecho do Projeto (4.8)	• Planear a passagem para a manutenção e garantia de acordo com o definido no manual de manutenção do sistema • Garantir que todas as ações resultantes das auditorias foram fechadas • Apresentar o balanço final do projeto, como lições aprendidas e pontos fortes, fracos e a melhorar • Elaborar o relatório de fecho do projeto e disponibilizar à equipa QAPE • Desmobilizar a equipa e os recursos do projeto

Tabela 12 Objetivos de cada atividade relacionadas com Gestão de Desenvolvimento da Fase 4 do PO Gestão e Engenharia de Projeto

A Tabela 13 apresenta a atividade de Engenharia da presente fase, nomeadamente a Entregar (4.5):

Atividade	Objetivo
Entregar (4.5)	• Instalar artefacto no Cliente em ambiente de produção • Disponibilizar o produto com entregáveis definidos • Proceder à formação do Cliente (se aplicável)

Tabela 13 Objetivos da atividade Entregar (4.5) relacionada com Engenharia da Fase 4 do PO Gestão e Engenharia de Projeto

No caso de a entrega coincidir com uma publicação num servidor de produção controlado pelo DSI, esta tem de ser agendada de acordo com o Procedimento Operacional denominado por PO.04 Sistemas de Informação.

3.3. Processos usados no Cloogy

Embora a ISA possua o processo genérico para desenvolvimento de *software*, o processo de Conceção e Desenvolvimento (C&D), a equipa que trabalha no Cloogy estabeleceu os seus próprios métodos de trabalho. Em vez de seguir uma metodologia tradicional – tipicamente baseada em projetos com um planeamento em Cascata, foi optado por trabalhar em *Scrum*, uma metodologia de desenvolvimento ágil que permite iterações curtas e contactos frequentes com os *stakeholders* dos projetos/produtos adaptando-se melhor às suas necessidades. Assim, foi feito um *tailoring* do processo atrás descrito, que teve como consequência o ajuste e eliminação de determinadas atividades do processo C&D, como mostrado na Tabela 14 presente no capítulo 3.3.2.

Desta forma, pretende-se salientar quais as diferenças do método utilizado no decorrer da realização do presente projeto.

3.3.1. O *Scrum* usado no Cloogy

De forma a descrever o método de trabalho da equipa de desenvolvimento do Cloogy, o presente capítulo tem como objetivo apresentar como o *Scrum* é usado nesta equipa.

O *Scrum*, apresentado na Figura 9, é um método iterativo e incremental que permite gerir um projeto. Este método, ao contrário de uma metodologia tradicional que descreve detalhadamente cada processo definido, enfatiza um conjunto de valores e práticas no que toca à gestão de um projeto.

Transparência, inspeção e adaptação, são condições que têm que surgir neste método e é imprescindível que a equipa seja transparente, para que os *stakeholders* estejam ao corrente do sucedido no projeto.

De modo a evitar desvios aos requisitos do projeto, é igualmente importante verificar com alguma regularidade se os resultados obtidos estão de acordo com os requisitos e expetativas, pois neste tipo de projetos é habitual haver vários pedidos de alteração ao projeto, que não sendo nada benéficos, causam uma constante adaptação da equipa.

A equipa que trabalha com este método deve ser auto-organizada e multifuncional. Desta forma, consegue-se minimizar o pedido de ajuda a entidades externas à equipa, pois é ela própria que planeia e executa o seu trabalho.

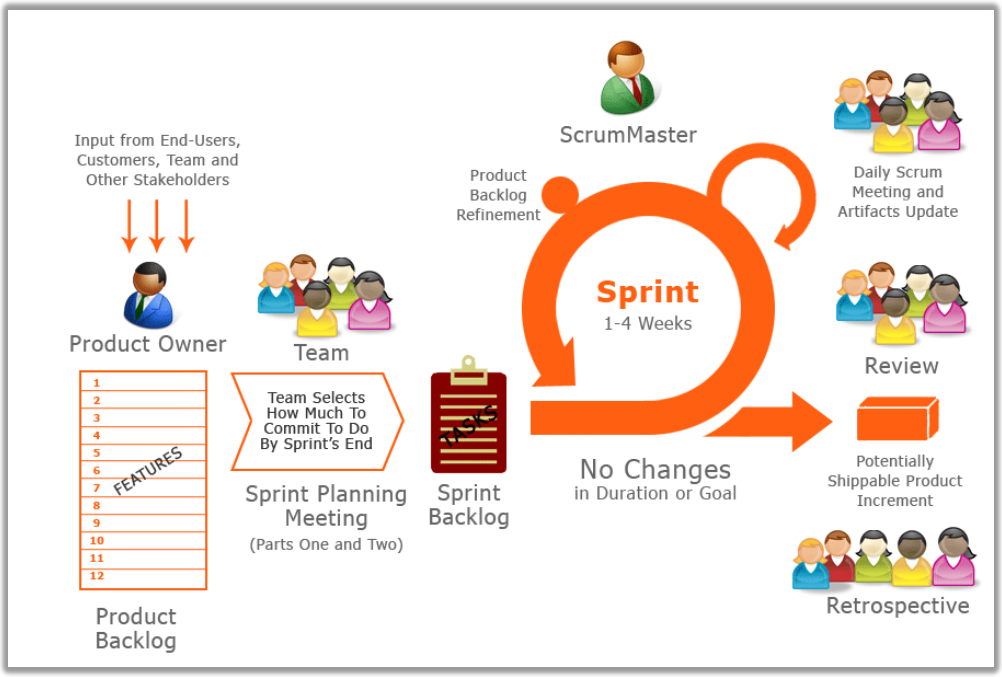


Figura 9 Processo Scrum [4]

O ciclo de vida deste método é constituído em três fases, a *Pre-Game*, *Development* e *Release*.

3.3.1.1. *Pre-Game*

A primeira fase do ciclo de vida do método usado pela equipa de desenvolvimento do Cloogy é a *Pre-Game*. Esta fase permite à equipa estabelecer o seu método de trabalho, identificar a visão do projeto, construir protótipos e o *Product Backlog*[15].

No início do projeto é preenchido um impresso para identificação do projeto, nomeadamente o impresso associado denominado por IP.33 Ficha de Projeto. Este impresso é constituído pelo nome do projeto, área de negócio, cliente, nomes do gestor e arquiteto do projeto, códigos para *Issue Tracking*, *Enterprise Resource Planning*[16] (ERP) e repositório, duração e orçamento previstos. Na sua essência, este impresso é o bilhete de identidade do projeto. Dá-se a conhecer à equipa DSI o início do projeto, com o objetivo de ser criado o repositório para o mesmo. No

[15] *Product Backlog* – lista onde constam todas as funcionalidades do *software* a desenvolver. No capítulo 3.3.1.1.3 é descrito o *Product Backlog* usado pela equipa de desenvolvimento do Cloogy de uma forma mais detalhada.

[16] ERP – Sistema de gestão empresarial. Integra dados e processos de uma organização num único sistema. Usado para sistemas de finanças, contabilidade, recursos humanos, vendas, entre outros.

início do projeto também é criado um documento a mencionar o método de trabalho e os colaboradores da equipa, assim como a definição das versões das aplicações e servidores usados.

Embora no *Scrum* não exista o papel de *Product Manager* (PM), nesta fase a equipa de desenvolvimento do Cloogy dividiu as tarefas do *Product Owner* atribuindo-as ao *Product Manager*. Ficando assim o *Product Manager* responsável pelas atividades referentes ao negócio do projeto. O *Product Manager* do Cloogy substitui claramente o cliente, possuindo assim a visão do projeto e as suas necessidades. Por outro lado, o *Product Owner* executa tarefas orientadas para a *Development Team*, tomando decisões técnicas. Com isto, foram definidos dois papéis e um artefacto imprescindíveis ao projeto, nomeadamente *Product Manager*, *Product Owner* e o *Product Backlog*, a seguir descritos.

3.3.1.1.1. *Product Manager*

O membro da equipa que executa o papel de *Product Manage*r (PM) representa o negócio, os clientes/utilizadores e direciona à equipa a ideia da solução final. Na sua essência o *Product Manager* é *customer-facing*, ou seja, substitui o cliente, traduzindo as suas necessidades em *User Stories*[17], é ele que conhece todas as funcionalidades do *software* a desenvolver, e que esclarece a equipa quando os requisitos não são claros. Possui a visão, o preço, o licenciamento e estabelece o *Product Backlog*. O *Product Manager* do Cloogy, sempre que queira descrever uma nova funcionalidade do produto, a ser inserida no *Product Backlog,* escreve a respetiva *User Story* e coloca-a no JIRA, uma ferramenta de *Issue Tracking*, assinalando ao *Scrum Master*[18] a respetiva *User Story*.

[17] *User Story* – Descrição simples e curta, contada na perspetiva de um indivíduo que deseja uma determinada funcionalidade a ser implementada no sistema. Geralmente é contada na perspetiva de um utilizador ou cliente do sistema.

[18] *Scrum Master* – Papel atribuído a um membro da equipa de desenvolvimento de *software.* Responsável por estabelecer as práticas do *Scrum* e focar a equipa de modo a atingirem os objetivos da *sprint.* No capítulo 3.3.1.2.1 é descrito o papel de *Scrum Master* na equipa de desenvolvimento do Cloogy de uma forma mais detalhada.

3.3.1.1.2. *Product Owner*

Ao contrário do *Product Manager*, o *Product Owner* (PO) da equipa de desenvolvimento do Cloogy é *team-facing*, ou seja, caso seja preciso tomar uma decisão importante de teor mais técnico, é ele que decide qual a melhor tecnologia/forma de resolver o problema, assim como também estabelece os critérios de aceitação das *User Stories* contadas pelo *Product Manager*.

O *Product Owner* juntamente com o *Product Manager*, *Scrum Master* e com um elemento da equipa que desempenha a função de *Business Analyst*[19], na reunião *Backlog Grooming*[20], são responsáveis por atribuir as *User Stories* a entrar no *Sprint Backlog*[21], onde posteriormente são detalhadas em tarefas. Estas tarefas são planeadas pela *Development Team*[22], sendo o *Product Owner* responsável por validar as estimativas efetuadas pela equipa, assim como a prioridade de cada uma delas.

3.3.1.1.3. *Product Backlog*

No início do projeto é estabelecido um *Product Backlog*, ou seja, uma lista onde constam todas as necessidades do cliente para o projeto. Estas necessidades são constituídas pelas *User Stories* contadas pelo *Product Manager* no JIRA, onde posteriormente serão assinaladas ao *Scrum Master*.

---

[19] *Business Analyst* – indivíduo que analisa uma organização e que projeta os seus processos e sistemas, avaliando o modelo de negócio e sua integração com a tecnologia.

[20] *Backlog Grooming* – reunião para preparação da *Planning Meeting*. Tem como objetivo definir as *User Stories* oriundas do *Product Backlog* a entrar em cada *Sprint Backlog*. No capítulo 3.3.1.2.3 encontra-se uma descrição mais detalhada.

[21] *Sprint Backlog* – lista que contém todas as tarefas a serem executadas na *sprint*. No capítulo 3.3.1.2.5 é descrito o *Sprint Backlog* usado pela equipa de desenvolvimento do Cloogy de uma forma mais detalhada.

[22] *Development Tem* – elementos da equipa responsáveis pela entrega do produto. No capítulo 3.3.1.2.2 encontra-se uma descrição mais detalhada da equipa de desenvolvimento do Cloogy.

3.3.1.2. *Development*

Durante a segunda fase do ciclo de vida do método utilizado, são realizadas *sprints*[23], com uma duração típica de duas semanas. Para que o *Scrum* funcione de forma correta, no decorrer destas *sprints*, são desempenhados dois papéis, executadas quatro cerimónias e utilizado um artefacto imprescindíveis ao projeto, nomeadamente *Scrum Master, Development Team*, *Planning Meeting*[24], *Daily Meeting*[25], *Sprint Review*[26], *Sprint Retrospective*[27] e *Sprint Backlog*. Para além das quatro cerimónias mencionadas, parte da equipa é reunida numa reunião chamada *Backlog Grooming*. Segue-se uma explicação de cada um destes.

3.3.1.2.1. *Scrum Master*

Uma vez que os elementos da equipa possuem diferentes competências técnicas e o Cloogy é constituído por vários componentes, por vezes é necessário atribuir tarefas de determinados componentes do projeto a elementos da equipa com as competências técnicas necessárias para tal, de forma a efetuar as tarefas de uma forma rápida e eficiente para que não haja atrasos no desenvolvimento do projeto.

Sempre que sejam atribuídas tarefas aos elementos da equipa, o membro que tem como papel o *Scrum Master* é responsável por assinalar as respetivas tarefas do *Sprint Backlog* pelos vários elementos da equipa na ferramenta JRIA.

Este elemento orienta os restantes membros, sendo responsável por estabelecer as práticas de *Scrum*, focar a equipa, atenuar obstáculos encontrados e responder a necessidades encontradas

---

[23] *Sprint* – unidade básica de desenvolvimento em *Scrum*. Geralmente com duração definida entre duas e quatro semanas, dentro do qual são executadas um conjunto de atividades.

[24] *Planning Meeting* – reunião executada no início de cada *sprint* com o objetivo de priorizar e planear tarefas a serem executadas na *sprint*. As *Planning Meetings* executadas pela equipa de desenvolvimento do Cloogy encontram-me descritas no capítulo 3.3.1.2.4.

[25] *Daily Meeting* – reunião diária de pequena duração com o objetivo de organizar a equipa. No capítulo 3.3.1.2.6 encontra-se uma descrição mais detalhada.

[26] *Sprint Review* – reunião executada no fim de cada *sprint*, com o objetivo de avaliar os requisitos implementados. No capítulo 3.3.1.2.7 encontra-se uma descrição mais detalhada.

[27] *Sprint Retrospective* – última cerimónia de cada *sprint*, tendo como objetivo analisar os aspetos positivos, os aspetos negativos e propor melhorias a fazer na próxima *sprint*. No capítulo 3.3.1.2.8 esta reunião é abordada com maior detalhe.

por parte dos elementos da equipa. O membro que desempenha este papel é responsável pela monitorização e controlo do projeto, atualizando os indicadores no JIRA.

3.3.1.2.2. *Development Team*

A *Development Team* é responsável por entregar um conjunto de funcionalidades do produto no fim de cada *sprint*. Como referenciado anteriormente, a *Development Team* do Cloogy é constituída por elementos com diferentes competências técnicas, por esta razão, as tarefas executadas pela *Development Team* por vezes são tarefas previamente atribuídas aos elementos da equipa, outras vezes são tarefas escolhidas pelos próprios elementos no desenrolar das *sprints*.

3.3.1.2.3. *Backlog Grooming*

Esta reunião é executada no início de cada *sprint*, antes da *Planning Meeting*. Tem como objetivo preparar o *Sprint Backlog* para a *Planning Meeting*, priorizando as *User Stories* do *Product Backlog* a entrar no *Sprint Backlog*.

Estas *User Stories* são priorizadas pelo *Product Owner*, *Product Manager*, *Scrum Master* e por um elemento da equipa que desempenha o papel de *Business Analyst*.

3.3.1.2.4. *Planning Meeting*

No início de cada *sprint*, após a *Backlog Grooming*, é elaborada uma *Planning Meeting*, cujo objetivo é construir o *Sprint Backlog*. Esta reunião tipicamente com duração de quatro horas tem o intuito de planear as tarefas pela *Development Team*, a serem executadas na *sprint*, estando presentes todos os elementos da equipa.

Caso o planeamento são seja concluído no mesmo dia, e como existem vários componentes no mesmo projeto, o planeamento é continuado no dia seguinte por cada grupo responsável pelo desenvolvimento de cada componente. Assim que todas as tarefas do *Sprint Backlog* estejam planeadas, é enviada esta informação ao *Product Owner,* para que este a possa validar.

3.3.1.2.5. *Sprint Backlog*

O *Sprint Backlog* é uma lista que contém todas as tarefas a serem executadas na *sprint*. Estas tarefas são resultantes das *User Stories* priorizadas na reunião *Backlog Grooming*, que são pequenas descrições das funcionalidades descritas na perspetiva do cliente/utilizador, provenientes do *Product Backlog*. Posteriormente ao planeamento das tarefas efetuado pela equipa, o *Product Owner* valida esse planeamento. Caso as tarefas sejam específicas para as competências técnicas de determinados elementos da equipa, o *Scrum Master* assinala essas tarefas aos respetivos elementos da equipa no JIRA, caso contrário, à medida que os elementos vão acabando as suas tarefas vão escolhendo tarefas novas.

3.3.1.2.6. *Daily Meetings*

Outra prática adotada por este método são as *Daily Meetings*, cujo objetivo é reunir a equipa diariamente, ao início do dia, numa reunião de pequena duração, tipicamente de não mais do que quinze minutos. Esta reunião é efetuada presencialmente ou via Skype, no caso dos elementos da equipa não estarem presentes. Esta reunião tem como objetivo organizar a equipa e para isto é necessário que cada um dos membros da equipa refira:

- O que fez no dia anterior;
- O que vai fazer no dia de hoje;
- As dificuldades encontradas.

Esta interação rápida permite uma entreajuda na equipa bem como uma rápida sincronização entre todos os membros que estão a desenvolver, tornando-se assim mais fácil ultrapassar eventuais obstáculos do dia-a-dia, permitindo uma troca de tarefas caso seja necessário.

3.3.1.2.7. *Sprint Review*

O último dia da *sprint* é começado por uma reunião denominada por *Sprint Review*. Nesta reunião, deverão estar presentes todos os *stakeholders* do projeto, para que possam assistir à demonstração do que foi desenvolvido durante a *sprint*, com o objetivo de que todos os requisitos sejam avaliados e a qualidade da sua implementação validada.

Esta reunião permite ao *Product Manager* receber *feedback* de todos os *stakeholders* presentes. Caso sejam propostas melhorias de funcionalidades, estas serão incorporadas no *Product Backlog.*

Caso os requisitos não estejam implementados corretamente, é reaberto o *issue* na ferramenta JIRA. Caso contrário, os requisitos são aprovados, ficando prontos para colocar em produção numa próxima *release*[28].

3.3.1.2.8. *Sprint Retrospective*

A última reunião da *sprint* é de carácter mais pessoal, onde apenas os elementos da equipa estão presentes. Nesta reunião é criado um documento, tipicamente um *power point,* onde são identificados:

- Aspetos positivos;
- Aspetos negativos;
- Melhorias a fazer na próxima *sprint*.

Esta reunião permite reconhecer o que correu mal e o que se pode melhorar para colmatar os problemas ocorridos na *sprint*. Desta forma, são identificados riscos para que possam ser avaliados e monitorizados.

3.3.1.3. *Release*

A terceira fase do ciclo de vida do *Scrum*, a *Release*, serve para fazer o *deployment* do projeto e elaborar a documentação necessária, entre outras atividades identificadas de acordo com cada projeto.

[28] *Release* – publicação de uma nova versão de um *software*, ao qual foram adicionadas novas funcionalidades ou correções.

3.3.2. Adaptação do processo de Conceção e Desenvolvimento

O processo de Conceção e Desenvolvimento segue uma metodologia tradicional de desenvolvimento de *software*, nomeadamente a Cascata. Por sua natureza, este tipo de metodologia é orientada à documentação, tornando o processo de desenvolvimento bem estruturado, sendo constituído por uma sucessão de etapas bem definidas, onde cada uma delas só é iniciada após o término da etapa anterior, resultante de um ou mais documentos.

Tipicamente estas etapas são o levantamento de requisitos, *design*, implementação, verificação e validação e manutenção. Contudo, em projetos de maior duração, por vezes estas fases trazem problemas, pois existem fases muito distintas umas das outras e caso se verifique o atraso de uma etapa, esta irá atrasar todo o projeto.

A falta de *feedback* entre as várias etapas nesta metodologia é outro problema, pois o cliente apenas obtém o produto no final do projeto, o qual pode não atender às suas expectativas.

Outro problema trazido pelo modelo em Cascata é a dificuldade em projetar o *software*, pois as etapas são muito extensas tornando assim mais difícil a previsão de todas as situações no desenvolvimento do projeto.

Sempre que seja necessário proceder a uma alteração de requisitos, após o início do desenvolvimento, esta metodologia revela-se morosa e por vezes muito cara, pois como o *software* é todo planeado e documentado antes da sua implementação, é necessário rever os planos e controlar as alterações.

Como mencionado anteriormente, a equipa de desenvolvimento do Cloogy seguiu o *Scrum* como metodologia de desenvolvimento de *software*. Caracteriza-se por ser um método iterativo e incremental e, ao contrário do modelo em Cascata que descreve cada uma das etapas do projeto, o *Scrum* não descreve o que fazer em cada situação.

Geralmente este método é usado em projetos onde seja muito difícil predizer tudo o que irá acontecer durante o projeto. Neste tipo de metodologias existe uma maior colaboração com o cliente, sendo-lhe entregue o trabalho desenvolvido durante as pequenas iterações do projeto, não recebendo o produto final apenas no término do mesmo.

Isto faz com que a equipa de desenvolvimento se foque mais no projeto e se torne mais auto-organizada e adaptativa às diversas situações encontradas durante o desenvolvimento do produto.

Uma vez que o projeto é desenvolvido no decorrer de pequenas iterações, traz uma maior facilidade à equipa em estimar as tarefas.

Contudo, este tipo de metodologias também tem desvantagens, torna-se difícil determinar o tempo que o projeto irá demorar, devido ao número de iterações necessários no desenvolvimento do mesmo ser uma incógnita e o facto de não se ter o âmbito bem definido no início do projeto.

A alteração do modo de trabalho da equipa levou com que esta fizesse um *tailoring* do processo C&D, ou seja, houve a necessidade de um ajuste das atividades definidas no processo, assim como dos impressos associados.

Um aspeto importante de salientar é que a maior parte das atividades mencionadas no processo C&D continuam a ser realizadas pela equipa que desenvolve o Cloogy, com a diferença que são executadas durante as *sprints* e há um maior acompanhamento entre a equipa de desenvolvimento e o cliente.

A forma como estas atividades são executadas leva com que esta equipa seja mais responsável, transparente e unida. Outro aspeto é o facto de atenuar a criação de documentos e de aumentar o número de reuniões com os *stakeholders*.

Estas alterações conduziram a um desenvolvimento de *software* mais rápido, satisfazendo melhor as necessidades do cliente. Devido à curta duração das *sprints*, a equipa consegue focar-se melhor nas tarefas que tem para fazer, sendo avaliadas sempre pelo cliente no final de cada iteração, evitando à posteriori custos acrescidos com alterações de requisitos. Todas estas alterações levam à maximização do ROI (*Return on Investment*).

Há que salientar que existem algumas atividades que não estavam contempladas no processo C&D e que são executadas pela equipa do Cloogy. Exemplo disto são as reuniões e artefactos típicos do *Scrum,* como *Planning* Meeting, *Daily Meeting, Sprint Review, Sprint Retrospective* e *Sprint Backlog*.

A Figura 10 apresenta um mapeamento entre o *Scrum* e o Processo genérico da ISA denominado por C&D.

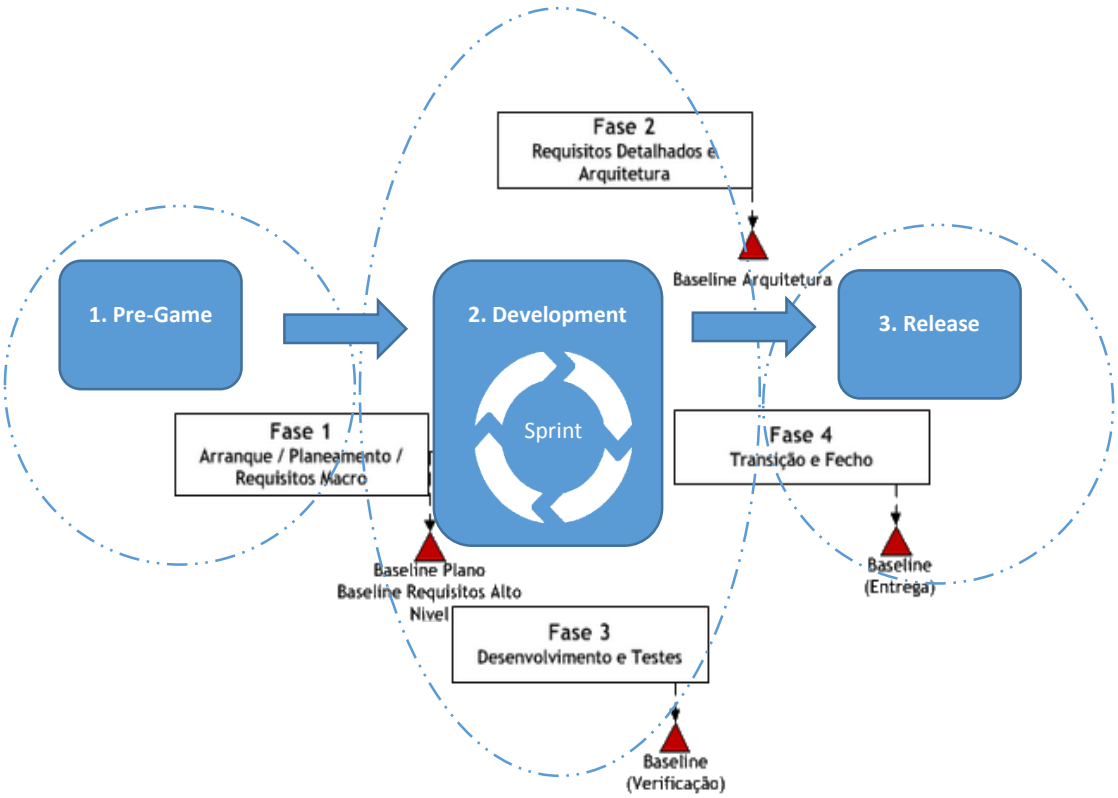


Figura 10 Mapeamento entre o Scrum e as quatro fases do processo C&D

As atividades da Fase 1 do procedimento operacional Gestão e Engenharia de Projeto são executadas durante a primeira fase do *Scrum*, nomeadamente a *Pre-Game*, assim como na segunda fase *Development*.

Na *Pre-Game* são preenchidos impressos, é preparado todo o material e ferramentas necessárias, é identificada a visão do projeto, são construídos *mockups* e o *Product Backlog* e é dado a conhecer à equipa DSI o arranque do projeto.

Por vezes na fase *Development*, é necessário introduzir novas funcionalidades ao *Product Backlog*, o qual leva a necessidade de levantar novos requisitos e estimativas. Nesta primeira fase, é estabelecida uma *baseline* de modo a estabelecer a *Definition of Done*, ou seja, esta *baseline* estabelece quando é que uma funcionalidade está devidamente preparada para ser entregue, o início e fim de uma *sprint* assim como o início da fase *release*.

As atividades referentes à Fase 2 e Fase 3 do procedimento operacional Gestão e Engenharia de Projeto são executadas durante as *sprints* da fase *Development* do *Scrum*, ou seja, todas as *sprints* contêm tarefas para detalhar requisitos, desenvolvimento do *software* e respetivos testes, enquadradas nas respetivas cerimónias e artefactos do *Scrum*.

A *baseline* da arquitetura é definida na primeira *sprint* do desenvolvimento, tipicamente uma *sprint* com maior duração para preparação de toda a arquitetura do sistema.

A *baseline* de verificação é estabelecida no fim de todas as *sprints*, devido à cerimónia *Sprint Review*, para validação das funcionalidades implementadas.

As atividades da Fase 4 são executadas nas fases *Development* e *Release* do *Scrum*. Na fase *Development* são executadas devido à necessidade da entrega contínua do *software* durante as cerimónias denominadas de *Sprint Review*, sendo assim executadas as atividades responsáveis pela entrega do *software*.

Na *Release* é feito o *deployment* do projeto, é elaborada a documentação necessária e procede-se ao fecho do projeto. Devido a esta entrega contínua, a *baseline* responsável é definida em todas as *sprints*.

Como referenciado anteriormente, a equipa de desenvolvimento do Cloogy, executou um *tailoring* do processo C&D, levando ao ajuste das atividades definidas no processo, assim como dos impressos associados.

De forma a perceber o *tailoring* realizado, a Tabela 14 apresenta as alterações efetuadas ao processo C&D:

Nº	**Fase**	**Ação**	***Tailoring* do processo C&D**	**Alternativa ao definido**
1.	Não aplicado	Evidência de Reunião de *Handover*	Não aplicado	A documentação do arranque do projeto é morosa e difícil, com isto, parte dessa documentação não é criada.
2.	Não aplicado	Evidência de aprovação de arranque	Não aplicado	
3.	*Pre-Game*	Preencher IP. 33 Ficha de Projeto	Como definido	
4.	*Pre-Game*	Colocar projeto na ferramenta de *Issue Tracking*	Como definido	Endereço do *SVN* do alojamento do projeto

5.	*Pre-Game*	Estruturar repositório de Projeto	Ajustado	O repositório não tem a estrutura definida, estando ajustada de acordo com as necessidades do projeto
6.	Não aplicado	Criar grupo de email	Não aplicado	
7.	Não aplicado	Enviar email de notificação ao QAPE	Não aplicado	
8.	*Pre-Game*	Preencher IP. 173 Plano do Projeto C&D	Não aplicado	O plano do projeto foi substituído pelo documento denominado por *Working Mode*[29]
9.	*Pre-Game, Development, Release*	Preencher IP. 172 Requisitos, Estimativas e Rastreabilidade	Ajustado	Feito com recurso à ferramenta JIRA. Esta permite consultar todo o projeto, inclusive as *User Stories* e as tarefas a serem executadas
10.	*Pre-Game, Development*	Efetuar planeamento do projeto	Ajustado	No início de cada *sprint* são definidos *os compromissos* com o cliente
11.	*Pre-Game*	Preencher IP. 170 *Tailoring* e Gestão de configurações	Como definido	
12.	*Pre-Game*	Preencher IP. 181 Recursos humanos do projeto	Ajustado	A constituição da equipa é definida no documento *Working Mode*.
13.	*Pre-Game*	Enviar email de notificação ao DSI (Plano de projeto C&D)	Ajustado	Os ambientes necessários para o desenvolvimento do projeto já existiam dos anos transatos.
14.	*Pre-Game*	Evidência da aprovação do *tailoring*	Como definido	
15.	*Development*	Preencher IP. 171 Monitorização do projeto	Não aplicado	Caso existam riscos, estes são identificados na reunião *Sprint Retrospective* que ocorre no fim de cada *sprint*

[29] *Working Mode* – documento que retrata o modo de trabalho da equipa, mencionando a constituição desta, metodologia de trabalho e respetivas cerimónias, endereços do repositório e da ferramenta JIRA.

16.	*Development*	Evidência da recolha inicial de riscos	Ajustado	Feito durante a reunião *Sprint Retrospective* no final de cada *sprint*
17.	*Pre-Game, Development, Release*	Evidência da discussão de necessidades com o cliente	Ajustado	O cliente faz parte da equipa e insere as suas necessidades no JIRA. Quando o detalhe não é suficiente, o *Product Owner* devolve a necessidade ao cliente e este detalha-a melhor
18.	*Pre-Game, Development*	Preencher IP. 160 Especificação dos requisitos	Ajustado	Requisitos especificados no JIRA
19.	*Pre-Game, Development*	Preencher IP. 180 Especificação de requisitos	Não aplicado	Não é usado o EA para especificação de requisitos
20.	*Pre-Game, Development*	Evidência da validação dos requisitos	Ajustado	Validação pelo P*roduct Owner* no JIRA
21.	*Pre-Game*	Preencher IP. 182 Orçamento de projeto	Ajustado	*Roadmap*
22.	*Pre-Game, Development*	Evidência da aprovação do plano e requisitos	Ajustado	Qualquer *issue* que entre para a equipa, é sempre validado pelo *Product Owner*
23.	*Development*	Preencher IP. 174 Checkpoint inicial	Não aplicado	O *checkpoint* inicial são as reuniões no início de cada *sprint*
24.	*Development*	Evidência do acompanhamento do projeto	Ajustado	Reuniões *Planning Meeting* de cada *sprint*
25.	*Development*	Evidência de gestão de riscos	Ajustado	Qualquer risco que seja identificado, será avaliado na *Sprint Retrospective*
26.	*Development*	Preencher IP. 104 Relatório de estado do projeto	Ajustado	Como o cliente faz parte da equipa, o estado do projeto é feito com recurso às demonstrações que fecham o ciclo de *Sprint*, ou seja nas *Sprint Review*
27.	*Development*	Evidência de apresentação do estado de projeto	Ajustado	Igual à anterior

28.	*Development*	Enviar Email ao QAPE (relatório de estado do projeto)	Não aplicado	Igual à anterior
29.	Não aplicado	Preencher IP. 160 Arquitetura	Não aplicado	
30.	Não aplicado	Preencher IP. 184 Arquitetura	Não aplicado	A arquitetura é um legado antigo que apenas vai sendo atualizado quando necessário
31.	Não aplicado	Evidência da validação da arquitetura e requisitos detalhados	Não aplicado	Igual à anterior
32.	*Development*	Evidência do compromisso da equipa	Ajustado	Email de compromisso enviado a todos os *stakeholders*, sendo posteriormente guardado no repositório de gestão do projeto.
33.	*Development*	Preencher IP. 166 Revisão de *Software*	Ajustado	Nem todos os requisitos estão sujeitos a teste.
34.	*Development*	Preencher IP. 93 Revisão/Verificação do projeto	Ajustado	Nas demonstrações, no fim de cada *sprint*, todos os requisitos são avaliados e é verificada a qualidade da sua implementação. Caso algum requisito não esteja implementado corretamente é reaberto o *issue* correspondente ao requisito no JIRA
35.	*Release*	Criação de manuais	Ajustado	Existem manuais para a instalação do equipamento e para a utilização do BackOffice
36.	*Pre-Game, Development*	Estruturar código fonte	Como definido	Tendo em conta a estrutura de repositório definida anteriormente
37.	*Release*	Preparar pacote de instalação	Como definido	
38.	*Release*	Preencher IP. 187 *Release* Notes	Ajustado	Cada *release* é publicada no *website* info.isa.pt.
39.	*Release*	Manual de manutenção do sistema	Como definido	

40.	Não aplicado	Preencher IP. 188 *Change* Log	Não aplicado	Consultar o *website* info.isa.pt
41.	*Development*	Evidência da validação do Cliente	Ajustado	O desenvolvimento é aprovado em demonstração no final de cada *sprint*. Após a aprovação é colocado em Produção. A aprovação é assumida se não houver contestação à demonstração feita. É ainda enviado um email, que deve ser arquivado no repositório, com a comunicação do sucesso da Demonstração.
42.	Não aplicado	Registar a Formação ao Cliente	Não aplicado	Este projeto não contempla formação
43.	*Development*	Evidência da aceitação do projeto	Ajustado	Igual à 41
44.	*Development*	Preencher IP. 48 Auto de Receção	Ajustado	Igual à anterior
45.	*Development, Release*	Enviar email ao QAPE (Requisitos, Estimativas e rastreabilidade)	Ajustado	Apenas os indicadores serão enviados
46.	*Release*	Evidência de Balanço final do projeto	Como definido	
47.	*Release*	Preencher IP.178 Relatório Final de Projeto	Como definido	
48.	*Release*	Enviar email ao QAPE (relatório fecho projeto)	Como definido	

Tabela 14 Resumo do Tailoring adotado

3.3.3. Métodos e ferramentas usadas para desenvolvimento móvel

A equipa de desenvolvimento das aplicações de iOS e de Android para o Cloogy definiu as suas próprias ferramentas para o seu desenvolvimento.

Perante o desenvolvimento do Cloogy para iOS, esta equipa usa o Xcode como ambiente de desenvolvimento e o CocoaPods como *dependency manager* para Objective-C, a linguagem de programação de iOS.

No que toca ao desenvolvimento de aplicações móveis nesta plataforma, existe o conceito de "aplicação universal", ou seja, uma aplicação desenvolvida nesta plataforma, que tanto corre num iPad como num iPhone, sendo ajustadas as vistas para o dispositivo em questão.

Em relação ao desenvolvimento em Android, a linguagem de programação é Java, sendo usado pela equipa do Cloogy, o Android Studio como ambiente de desenvolvimento com o respetivo SDK do Android.

Esta plataforma suporta um comportamento semelhante em relação à plataforma anterior, ou seja, a mesma aplicação desenvolvida para Android, tanto corre num *smartphone* como num *tablet*, sendo igualmente necessário ajustar as vistas da aplicação.

Independentemente da arquitetura usada, também existem métodos e ferramentas semelhantes para o desenvolvimento das aplicações, entre elas o JIRA, Balsamiq Mockups, Enterprise Architect e o uso de emuladores para testar as respetivas aplicações.

3.3.3.1. JIRA

À semelhança de todas as equipas da ISA, independentemente dos projetos que desenvolvam, a equipa do Cloogy usa o JIRA para controlo e monitorização do desenvolvimento das suas aplicações.

A Figura 11 apresenta um exemplo de um ecrã da ferramenta.

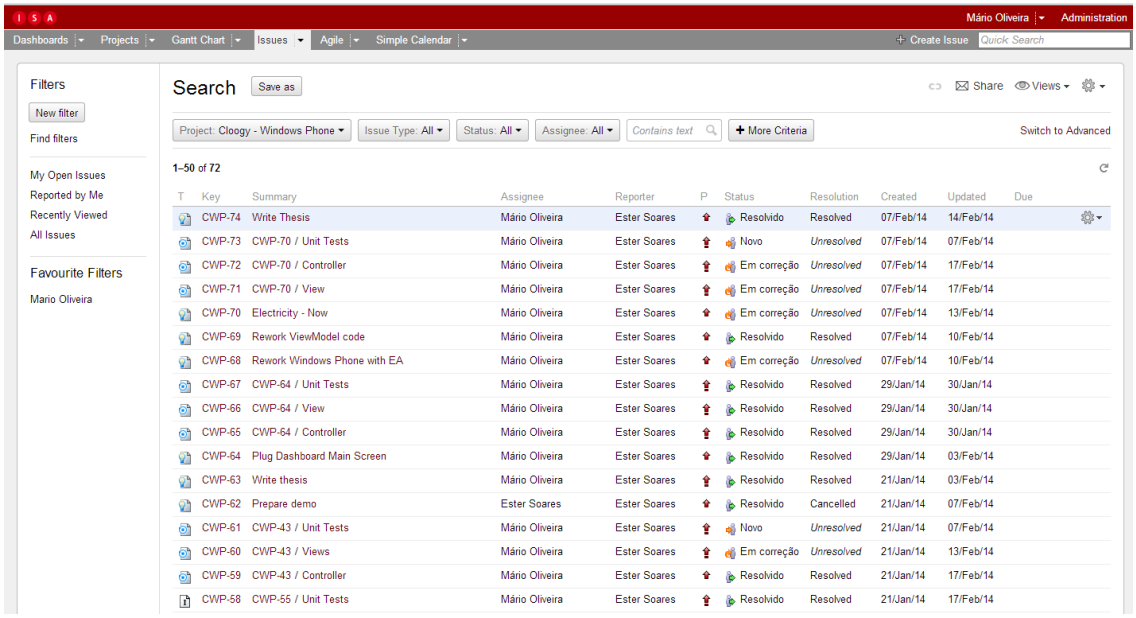


Figura 11 Exemplo de um ecrã da ferramenta JIRA

Esta ferramenta *de Issue Tracking* é usada para monitorização do *Product Backlog*, das *User Stories*, do *Sprint Backlog* e do estado atual dos Requisitos. A ferramenta permite ainda monitorizar o esforço gasto na implementação de cada *issue* em que cada colaborador esteve a desenvolver.

3.3.3.2. Balsamiq Mockups

De forma a ajudar no levantamento de requisitos, a equipa utiliza o Balsamiq Mockups para criar *mokups*. Há que ressalvar que todo o desenvolvimento de aplicações com forte interação com o utilizador são definidas numa abordagem *top-down*, isto é, primeiro definem-se todos os interfaces de utilizador e só posteriormente se inicia a implementação das respetivas funcionalidades, garantido desta forma que as vistas que se implementam serão, de facto, as vistas definitivas utilizadas pelo utilizador final, evitando assim a implementação de vistas que possam ser rejeitadas numa fase seguinte.

A Figura 12 apresenta um exemplo do uso da ferramenta.

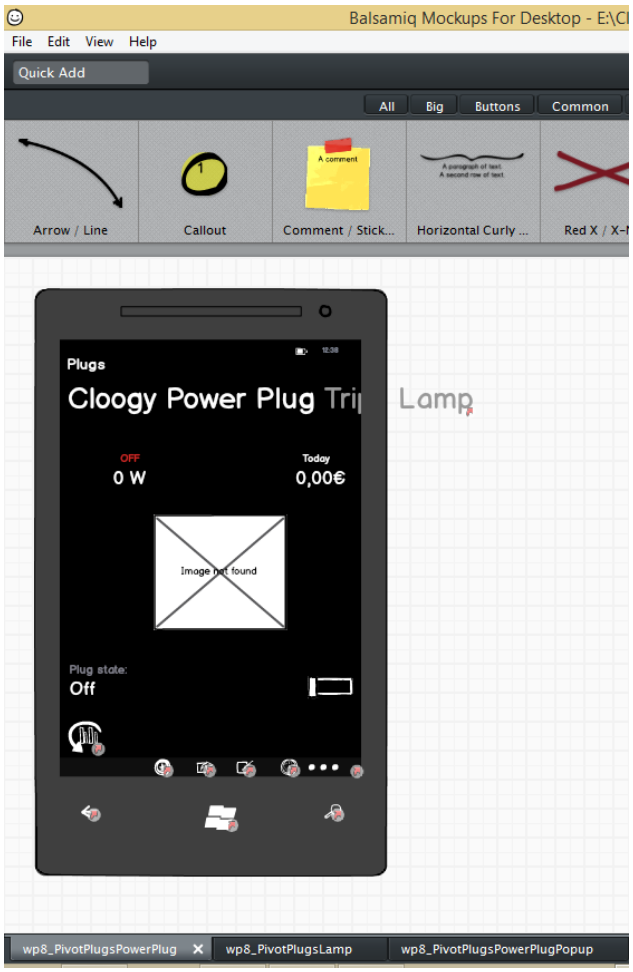


Figura 12 Balsamiq Mockups

3.3.3.3. Enterprise Architect

Outra das ferramentas usadas em ambas as plataformas é o Enterprise Architect (EA), cuja finalidade é ajudar a definir a arquitetura das aplicações.

A Figura 13 apresenta o exemplo do uso desta ferramenta.

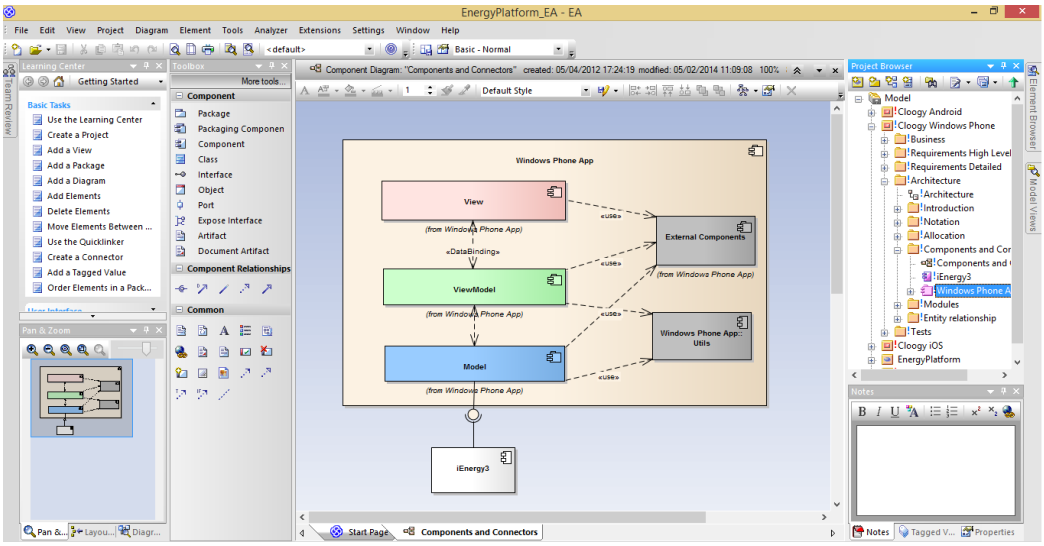


Figura 13 Enterprise Architect

3.3.3.4. Emuladores

Hoje em dia os sistemas operativos correm em múltiplos dispositivos com resoluções diferentes, por isso, testar as aplicações em vários emuladores com resoluções diferentes, é uma mais-valia.

No caso de iOS, o ambiente de desenvolvimento Xcode fornece cinco emuladores para as cinco resoluções diferentes, como apresentado na Tabela 15:

Resolução	Dispositivo
320x480	• iPhone 3G • iPhone 3GS
640x960	• iPhone 4 • iPhone 4S
640x1136	• iPhone 5 • iPhone 5C • iPhone 5S
1024x768	• iPad • iPad 2 • iPad Mini
2048x1536	• iPad Air • iPad Mini Retina

Tabela 15 Resoluções e dispositivos do iOS

A plataforma Android tem uma discrepância enorme nas diversas resoluções existentes dos variadíssimos dispositivos que se encontram no mercado, o que leva a uma maior dificuldade em testar as aplicações em todas as resoluções. Neste caso, a equipa do Cloogy escolhe as resoluções que são mais utlizadas para a realização de testes da interface da aplicação.

De forma a testar as aplicações nas diversas resoluções definidas, a equipa utiliza o emulador Genymotion.

4. Aplicação desenvolvida em Windows Phone 8 para o Cloogy

De forma a apresentar a aplicação desenvolvida para o Cloogy durante a execução deste estágio, o presente capítulo tem como objetivo expor o ambiente da aplicação, ferramentas usadas, arquitetura da aplicação, tecnologias e linguagens de programação usadas, testes efetuados e resultados obtidos.

4.1. Ambiente da aplicação

A plataforma Windows Phone é um sistema operativo para terminais móveis desenvolvido pela Microsoft, sucessora da plataforma Windows Mobile, adotando uma nova interface gráfica, conhecida como *Modern UI* ou Metro.

A aplicação desenvolvida para o Cloogy no presente estágio foi desenvolvida sob a plataforma Windows Phone 8, em particular na terceira geração do sistema operativo Windows Phone para terminais móveis, lançada em 29 de Outubro de 2012 pela Microsoft.

4.2. Ferramentas

Até ao momento a ISA não tinha qualquer experiência de desenvolvimento na plataforma Windows Phone, sendo a aplicação do Cloogy para Windows Phone 8 a primeira.

No seu desenvolvimento foi usado o IDE Visual Studio como ambiente de desenvolvimento recorrendo ao Windows Phone SDK 8.0, de forma a obter os mecanismos necessários para o desenvolvimento do projeto na plataforma Windows Phone 8. Estas ferramentas fornecem diversos controlos necessários ao desenvolvimento do projeto, emuladores para correr a aplicação nas diversas resoluções suportadas pelo sistema operativo e ferramentas que possibilitam testar a aplicação, para ajudar à sua submissão na Windows Phone Store.

À semelhança das aplicações anteriormente desenvolvidas para as plataformas iOS e Android, existem ferramentas de apoio que também foram usadas no desenvolvimento para Windows Phone 8, nomeadamente o JIRA, Balsamiq Mockups, Enterprise Architect e o TortoiseSVN.

O JIRA foi usado para controlo e monitorização do projeto. O Balsamiq Mockups foi usado para criação dos *mockups* da aplicação, o Enterprise Architect suportou a definição da arquitetura da

aplicação desenvolvida e o TortoiseSVN foi usado para alojamento e controlo de versões de todos os documentos necessários à realização do projeto.

4.3. Tecnologias e linguagens de programação usadas

Este capítulo tem como objetivo mencionar as tecnologias e linguagens de programação usadas no desenvolvimento deste projeto.

4.3.1. Padrão MVVM

No início de um projeto, é imprescindível definir uma arquitetura para o *software* a desenvolver, pois as suas funcionalidades tendem a aumentar, o que leva como consequência o aumento das linhas de código. É importante manter o código organizado, de modo a que seja fácil de rever, manter e estender. Como mostrado na Figura 14, existem vários padrões arquiteturais de *software*.

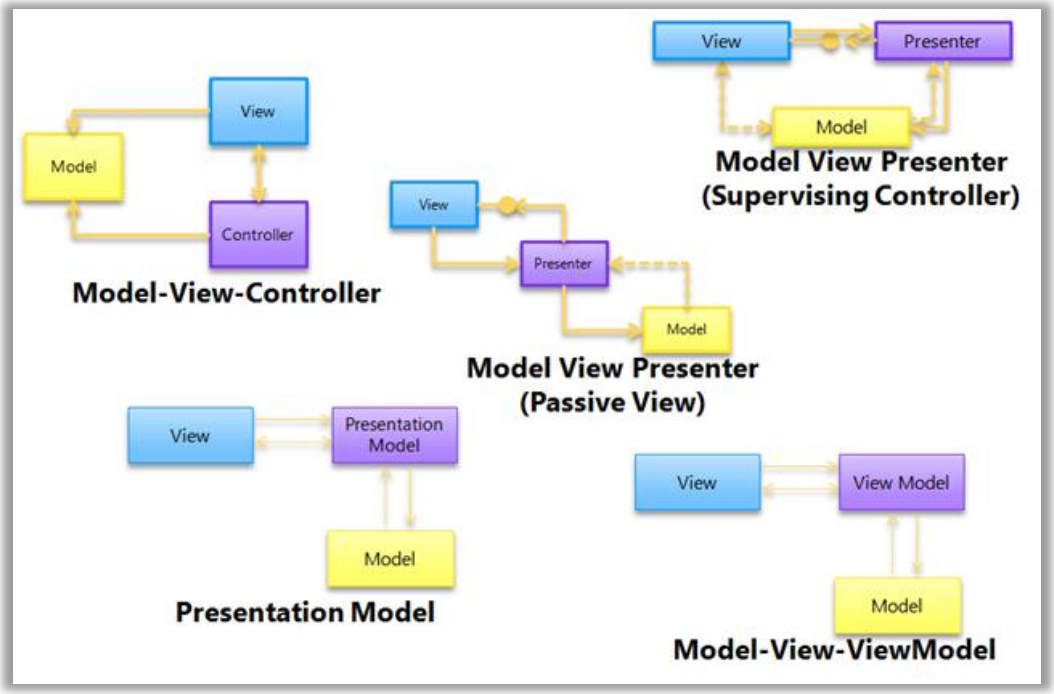


Figura 14 Exemplos de padrões de software [5]

Desde há várias décadas que os programadores utilizam uma derivação do padrão *Model View Controller* (MVC), o padrão arquitetural *Model View Presenter* (MVP).

O *Model* tipicamente é constituído pelos dados da lógica de negócio, residentes numa base de dados ou *web services*, a serem visualizados na *View*.

A *View* deste padrão é constituída pelos controlos da interface do utilizador. No entanto, quando o utilizador interage com estes controlos, a lógica necessária para os tratar é delegada ao *Presenter*.

O *Presenter* é responsável por tratar os eventos disparados da interface do utilizador e por sincronizar a *View* com o *Model,* no entanto o *Presenter* não comunica diretamente com a *View*, necessitando de uma interface para esse efeito.

Em 2004, Martin Fowler publicou um artigo [6] sobre outro padrão, o *Presentation Model*.

Este padrão é similar ao MVP, permitindo separar a vista dos diversos estados e comportamentos. Contudo, o *Presentation Model* facilita a visualização dos dados do *Model* na *View*. Uma vantagem deste padrão em relação ao MVP é a possibilidade do *Presentation Model* guardar informação relacionada com os controlos da interface do utilizador, por exemplo, é capaz de saber qual o item selecionado de uma lista.

Em 2005, John Gossman [7] revelou um novo conceito, o *Model View ViewModel* (MVVM). É um padrão arquitetural usado em engenharia de *software* aparecendo como uma especialização do padrão *Presentation Model*.

Baseado no padrão MVC, este padrão separa claramente o desenvolvimento da interface com o utilizador, do desenvolvimento da lógica de negócio ou *back-end,* como se pode ver na Figura 15.

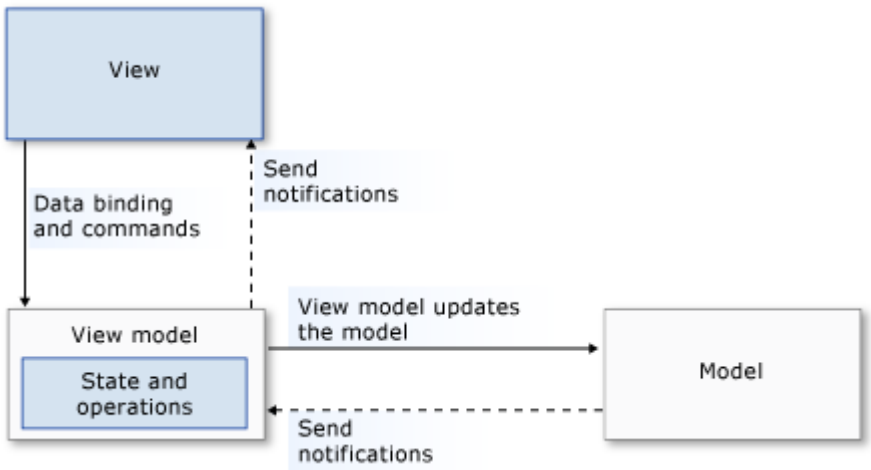


Figura 15 MVVM [8]

- O *Model* representa o estado e as operações dos objetos de negócio manipuladas pela aplicação;
- A *View* contém elementos da interface com o utilizador e inclui código que implementa a experiência do utilizador com a aplicação, ou seja, são definidas estruturas e *layouts*, onde irão conter ecrãs, *grids*, botões, caixas de texto, ou qualquer outro objeto com que o utilizador possa interagir;
- O *ViewModel* encapsula o estado, as ações e as operações da aplicação. Este serve como uma camada de separação entre o *Model* e a *View*, fornecendo os dados num formato que a *View* possa apresentar e atualizando o *Model* para que não haja a necessidade de interação entre o *Model* e a *View*. O *ViewModel* consegue assim responder a comandos e eventos, onde funciona como fonte de dados para os dados mostrados pela *View*.

O padrão MVVM permite assim uma independência clara entre as suas camadas, podendo atenuar o uso do *code-behind*[30], fazendo o uso de mecanismos de *databinding* para comunicação.

4.3.2. XAML

O XAML [9] é a principal linguagem de interfaces gráficas de utilizador (GUI) da Microsoft, sendo uma linguagem de marcação declarativa baseada em XML que é usada para criar interfaces de uma forma rápida e simples.

Ao longo do tempo esta linguagem tem sido designada como sucessora do HTML, que é uma ideia completamente errada.

Aplicada ao modelo de programação da *.Net Framework*, esta linguagem tende a simplificar a criação do interface de utilização de uma aplicação, permitindo criar elementos visíveis na marcação declarativa de XAML e associar um ficheiro *code-behind* separado do ficheiro XAML com capacidade de responder a eventos e manipular os objetos declarados na linguagem XAML. Contudo, a maior preciosidade do uso do XAML com o padrão MVVM, é o facto de ambos fornecerem mecanismos de *databinding*, atenuando assim a necessidade de desenvolver *code-behind* para atualizar diretamente a vista.

[30] *Code-behind* – técnica usada em *web design* (especificamente pela plataforma ASP.NET da Microsoft) onde cujo código fonte da interface gráfica e do *back-end* estão alojados em ficheiros separados, de modo a atenuar dependências entre os *designers* e os *developers*.

4.3.3. C#

A linguagem de programação C# [10] é uma linguagem desenvolvida pela Microsoft que faz parte de um conjunto de ferramentas oferecidas pela *.Net Framework*. É uma linguagem que possui uma sintaxe expressiva, elegante, simples, robusta e totalmente orientada a objetos. É baseada em C++, tendo algumas influências de outras linguagens, tais como Java e Object Pascal.

Esta linguagem de programação permite criar aplicações para o sistema operativo Windows, *web services* baseados em XML, componentes distribuídos, aplicações cliente-servidor, aplicações que acedam uma base de dados, entre outras.

Mesmo sendo baseada em C++, fornece um conjunto de recursos preciosos ao programador, como tipos de valor nulo, enumerações, *delegates*, expressões lambda e acesso direto à memória.

Outro recurso de enorme valor é o facto de esta linguagem suportar expressões LINQ, o que torna as consultas a base de dados mais simples e rápidas.

Como é típico das linguagens orientadas a objetos, esta linguagem suporta também os conceitos de encapsulamento, herança e polimorfismo.

A linguagem C# é fortemente tipada[31], *case-sensitive* e possui suporte a DLL's, COM[32] e COM+.

As suas classes podem implementar várias interfaces e os seus métodos e tipos de dados não necessitam de ser declarados por ordem.

Esta linguagem tem como objetivo facilitar o desenvolvimento, possuindo um ótimo conjunto de recursos que tendem em proporcionar uma boa produtividade.

Uma vez que esta linguagem está aliada à *.Net Framework*, os programadores podem implementar vários tipos de aplicações sem terem que se preocupar com a monitorização de recursos, tais como memória, uma vez que a *framework* se encarrega de monitorizar esses recursos, gerindo a sua alocação e libertação. Contudo, esta linguagem apenas corre em ambiente Windows.

[31] Linguagem tipada – É uma linguagem cuja declaração do tipo de variáveis é obrigatória.

[32] COM (*Component Object Model*) – plataforma da Microsoft para componentes de *software*. Permite a comunicação entre processos e a criação dinâmica de objetos.

4.3.4. JSON

Criado por Douglas Crockford, JavaScript Object Notation mais conhecido por JSON [11], é um formato leve para troca de dados computacionais baseado na linguagem de programação JavaScript. JSON é um formato de texto simples, fácil de interpretar e gerar. Tem vindo a difundir-se, tornando-se uma alternativa ao XML, devido à sua simplicidade e ao uso de convenções que são familiares em múltiplas linguagens de programação, como C, C++, C#, Java, JavaScript entre outras.

O exemplo retratado na Figura 16 mostra um *array* com três objetos, nomeadamente os objetos com nomes "João", "Maria" e "Pedro", com as respetivas notas.

```
{ "Alunos" : [
        { "nome": "João", "notas": [ 8, 9, 7 ]  },
        { "nome": "Maria", "notas": [ 8, 10, 7 ] },
        { "nome": "Pedro", "notas": [ 10, 10, 9 ] }
    ]
}
```

Figura 16 Exemplo de um array de Alunos em JSON

4.3.5. LINQ

LINQ (*Language Integrated Query*) [12] é um componente introduzido em 2007 no Visual Studio a partir da *.Net Framework* 3.5, que adiciona funcionalidades de consulta às linguagens de programação C# e Visual Basic.

A sua sintaxe é inspirada em SQL, permitindo assim realizar consultas diretas a coleções de dados, como bases de dados, documentos XML, estruturas de dados e coleções de objetos.

Este componente torna possível inicializar diretamente uma variável com os resultados de uma consulta.

Como mostrado na Figura 17, todas as consultas que são efetuadas a uma coleção de dados, resumem-se a três operações:

- Obter fonte de dados;

- Criar consulta;
- Executar consulta.

```
class IntroToLINQ
{
    static void Main()
    {
        // The Three Parts of a LINQ Query:
        //  1. Data source.
        int[] numbers = new int[7] { 0, 1, 2, 3, 4, 5, 6 };

        // 2. Query creation.
        // numQuery is an IEnumerable<int>
        var numQuery =
            from num in numbers
            where (num % 2) == 0
            select num;

        // 3. Query execution.
        foreach (int num in numQuery)
        {
            Console.Write("{0,1} ", num);
        }
    }
}
```

Figura 17 Exemplo do uso de LINQ [13]

4.3.6. REST API

Para a comunicação entre a aplicação e os dispositivos físicos, foi usada uma *Application Programming Interface* (API) fornecida pela entidade acolhedora. Esta API segue os princípios de um serviço RESTful, ou seja, usa uma API REST e o protocolo HTTP para comunicação entre os clientes e o servidor.

Para proceder a esta comunicação é necessário indicar o endereço da API, o tipo de dados recebidos e um conjunto de operações que sejam necessárias para obter os resultados obtidos.

Estas operações são tipicamente métodos HTTP, como GET, PUT, POST ou DELETE, sendo que estes métodos servem para informar o servidor o que fazer com os URLs acompanhados. O pedido pode opcionalmente anexar informação adicional, por exemplo, para proceder à autenticação de uma conta, é necessário enviar o nome de utilizador e respetiva palavra-passe.

Estes dados são inseridos numa *string* no formato desejado, tipicamente JSON ou XML.

O método GET é o mais simples, que apenas informa o servidor para enviar dados resultantes do URL para o cliente. Neste método, os dados nunca devem ser modificados do lado do servidor como resultado de um GET *request*, ou seja, deverá ser sempre *read-only*.

O contrário acontece no lado do cliente, o resultado proveniente do servidor, poderá sofrer qualquer tipo de operações, pois não afeta os dados residentes no servidor.

O método DELETE é utilizado no caso de o cliente querer eliminar os dados enviados através do URL.

Tipicamente o método POST serve para criar dados, enquanto o método PUT serve para atualizar dados identificados no URL, no entanto estes métodos podem ser usados para ambas as funções, dependendo de como o servidor for implementado. No caso de um pedido para atualizar um utilizador, o cliente apenas envia os dados necessários para tal, não necessitando de dizer como é que o utilizador deve ser atualizado, facilitando assim as operações do lado do cliente.

4.3.7. Telerik

Um dos requisitos primórdios da aplicação a desenvolver é a visualização de consumos energéticos graficamente. De forma a facilitar o desenvolvimento dos gráficos e uma vez que a ISA já conhecia os seus controlos, foram utilizados componentes externos fornecidos pelo Telerik, para construção desses gráficos.

4.3.8. Windows Phone Toolkit

O Windows Phone Toolkit [14] fornece aos programadores de aplicações para Windows Phone novos componentes, novas funcionalidades e melhores formas que ajudam ao desenvolvimento de *software* nesta plataforma. Este fornece um conjunto de controlos como:

- *AutoCompleteBox*;
- *ContextMenu*;
- *DateTimePickers*;
- *ToggleSwitch*;

- *Navigation Transitions*;
- Entre outros.

No presente projeto, o Windows Phone Toolkit foi utilizado para se poder usufruir dos controlos *DateTimePickers*, *NavigationTransition* e *ContextMenu*.

4.3.9. Bibliotecas externas usadas

De forma a fazer pedidos à API para receber/enviar dados, foram utilizadas duas bibliotecas externas ao sistema para responder a estas necessidades:

- RestSharp;
- Newtonsoft JSON.

A biblioteca RestSharp foi usada para realizar operações HTTP como GET, POST, PUT e DELETE, de modo a comunicar com a API.

Sempre que a aplicação recebe dados vindos da API, estes são recebidos no formato JSON.

De modo a colocar esta informação em objetos, a biblioteca Newtonsoft JSON foi usada para fazer a transformação dessa informação. Desta forma a informação recebida transforma-se em objetos podendo assim ser tratada.

4.4. Desenvolvimento do sistema

A mesma aplicação, quer para iOS, quer para Android pode correr tanto num *smartphone* como num *tablet*.

Ao contrário destas plataformas (iOS e Android), as aplicações para *smartphone* e *tablet* no Windows têm que ser diferentes, ou seja, uma aplicação para Windows Phone 8 apenas corre num *smartphone* e uma aplicação para Windows RT apenas corre num *tablet*.

Sendo o único objetivo desenvolver o Cloogy para *smartphones* Windows, a aplicação foi desenvolvida para Windows Phone 8.

De seguida apresenta-se como foram recolhidos os requisitos da aplicação, como foi definida a sua arquitetura e estratégias adotadas, como foi definida a estrutura da aplicação, como foi feita

a utilização de *Resources* e os *User Controls* que foram necessários implementar de raiz. Apresenta-se também os testes que foram definidos e implementados e os resultados obtidos.

4.4.1. Elaboração de *user stories* e *mockups*

Na primeira fase do desenvolvimento deste projeto começou por se definir os ecrãs da aplicação. Desta forma, as necessidades dos *stakeholders* foram traduzidas em *User Stories*, cujas histórias foram utilizadas para desenhar os vários *mockups* da aplicação na ferramenta Balsamiq Mockups. Tipicamente as *User Stories* são contadas pelo *Product Manager* da equipa de desenvolvimento do Cloogy, no entanto, para que obtivesse experiência em todas as fases do desenvolvimento do projeto, a elaboração das *User Stories* e os *mockups* do projeto fizeram parte do âmbito do desenvolvimento. À medida que estes *mockups* foram desenhados, foram levantadas várias questões relacionadas com os seguintes pontos:

- Usabilidade;
- Experiência do utilizador;
- Linhas orientadoras concebidas pela Microsoft;
- *Design* da aplicação.

Os quatro pontos referidos são essenciais para o sucesso da aplicação.

A usabilidade é um aspeto importante da aplicação, pois é extremamente importante que um utilizador, ao usar a aplicação, consiga atingir o objetivo pelo qual a iniciou, de uma forma eficiente e eficaz, caso contrário irá achar o seu uso complicado, o que irá atenuar o uso da aplicação.

A experiência de utilização da aplicação também é importante, pois o utilizador até pode atingir o seu objetivo inicial, mas se a experiência no seu global não for a melhor, a aplicação também não terá sucesso.

Outro aspeto importantíssimo é conjugar as linhas orientadoras de *design* concebidas pela Microsoft, com o *design* da aplicação. Caso o *design* da aplicação não siga estas linhas orientadoras, a aplicação não será aceite na Windows Phone Store.

Para que os pontos referidos atrás fossem atingidos e que fosse possível chegar a uma uniformização dos vários *mockups* desenhados e do *design* implementado, estes foram discutidos com os vários *stakeholders*, dentro e fora da equipa do Cloogy.

4.4.2. Arquitetura do sistema e estratégias adotadas

Como referenciado anteriormente, a solução Cloogy está disponível aos seus utilizadores sob as plataformas *web* e *mobile*.

A Figura 18 retrata os módulos da solução Cloogy, em que cada cliente, ou seja, a aplicação a executar em cada dispositivo móvel ou computador, comunica através da API da ISA para troca de dados e execução de funcionalidades disponíveis nos equipamentos da solução Cloogy.

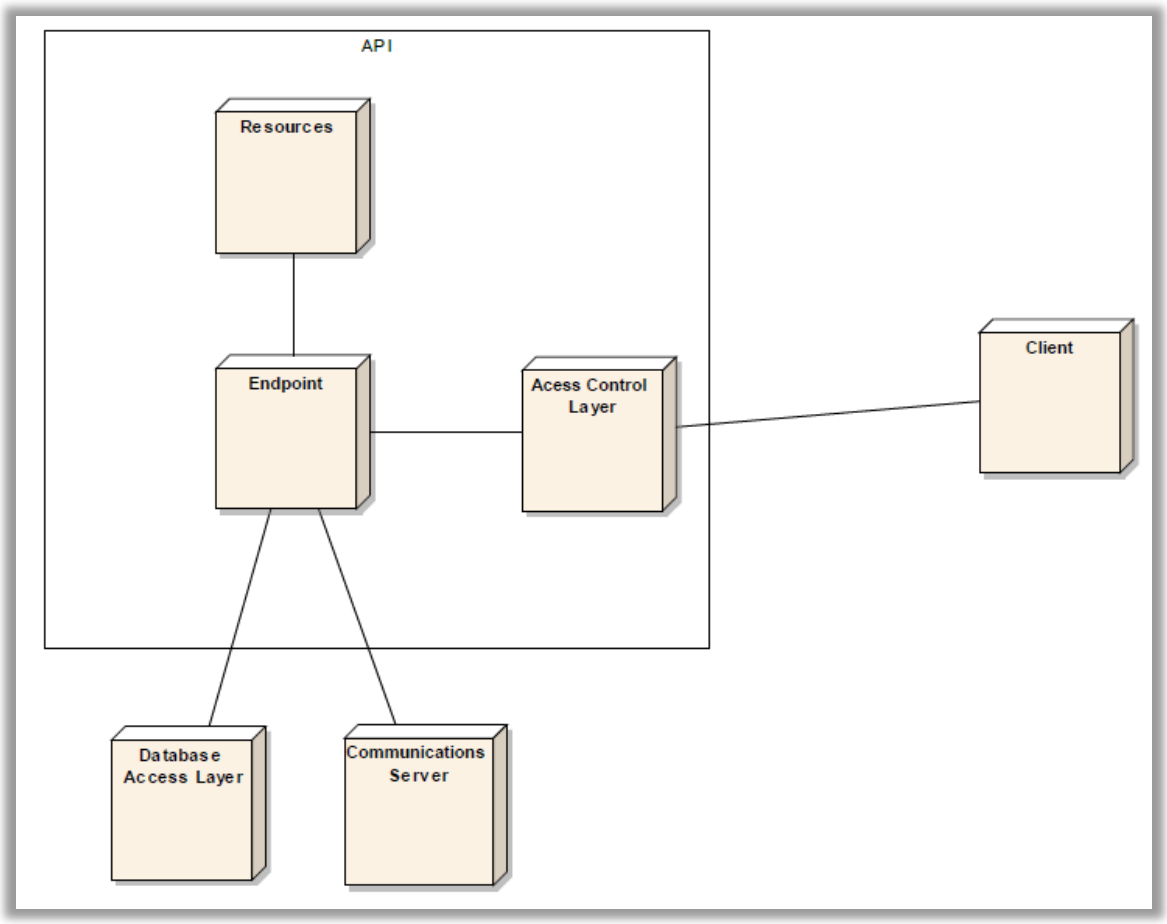


Figura 18 Arquitetura da solução Cloogy

A API abstrai os dados (por exemplo os dados de consumo) que estão presentes através da camada de acesso a dados – "*Database Access Layer*" e as funcionalidades (por exemplo o controlo de uma tomada inteligente) que estão disponíveis através do servidor de comunicações – "*Comunications Server*".

Para validar o acesso, quer aos dados, quer às funcionalidades, através de recursos - "*Resources*" (por exemplo um dispositivo) da aplicação, é utilizada uma camada de controlo de acesso -

"*Access Control Layer*" para que, dependendo do perfil que está a aceder, os recursos sejam disponibilizados.

A API, tal como já referenciado, contém um sistema de mensagens do tipo *request-response*, expresso no formato JSON, acessível através de métodos HTTP.

A Figura 19 retrata a decisão optada da arquitetura da aplicação cliente desenvolvida, passando por ser usado o padrão arquitetural MVVM, de modo a facilitar a separação do desenvolvimento das vistas da aplicação e do desenvolvimento da lógica de negócio.

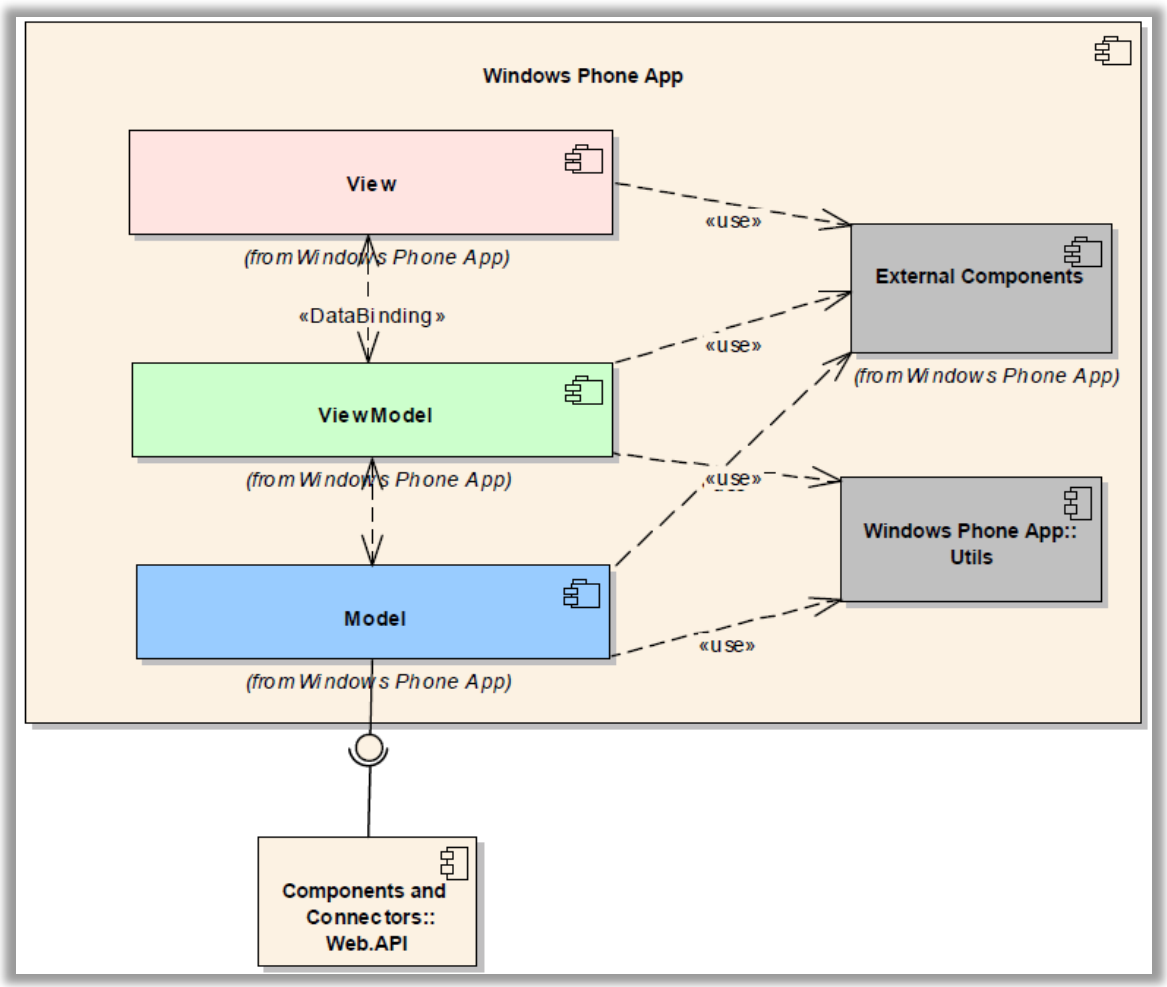


Figura 19 Arquitetura da aplicação Cloogy para Windows Phone 8

Tendo já sido explicados os papéis da *View*, *ViewModel* e *Model*, no capítulo 4.3.1, foram também definidos um conjunto de métodos utilitários (*Utils*) e escolhido um conjunto de componentes (*External Components*) necessários à aplicação.

Representado na Figura 20, o *Model* funciona como um *proxy*, realizando pedidos à API, sendo claramente independente do resto do programa.

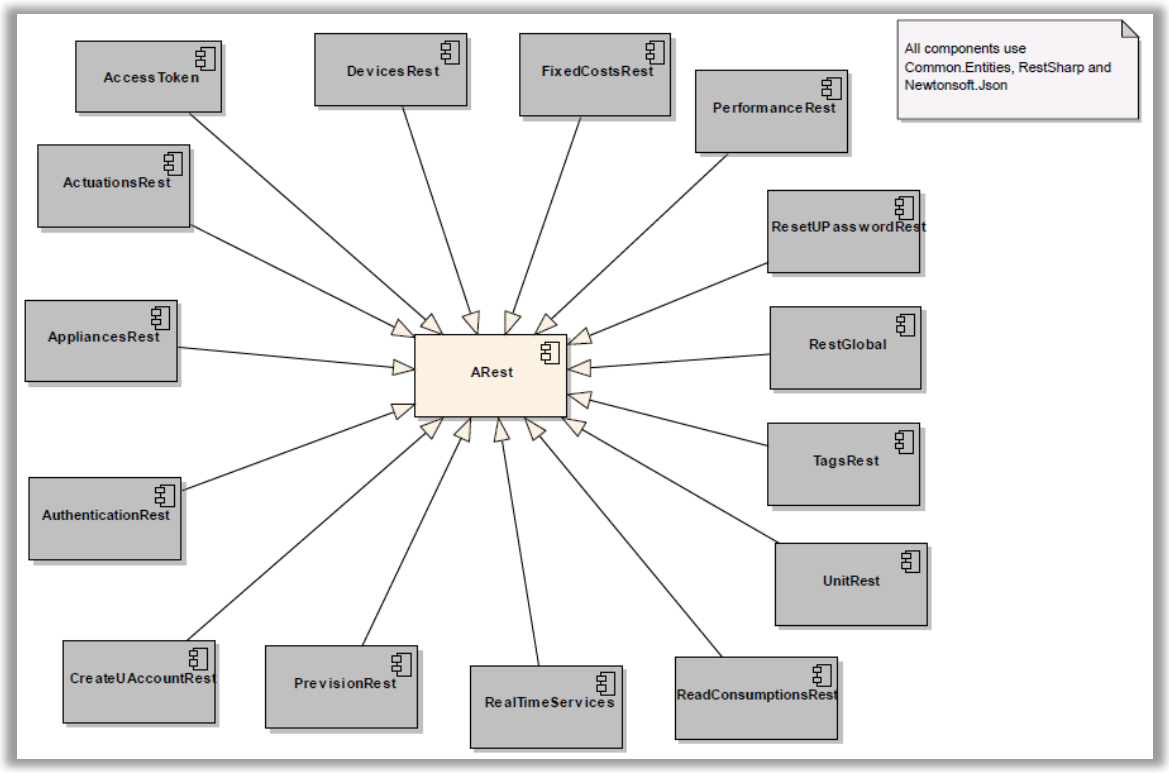


Figura 20 Detalhe do Model definido

Pode-se ver cada uma das classes que foram definidas e que modelaram a camada de negócio da aplicação.

Foi definido o interface *INotifyPropertyChanged*, como apresentado na Figura 21, para que, utilizando o evento *NotifyPropertyChanged*, as alterações aos dados do *Model* sejam refletidas na *View*, através de mecanismos de *databinding*.

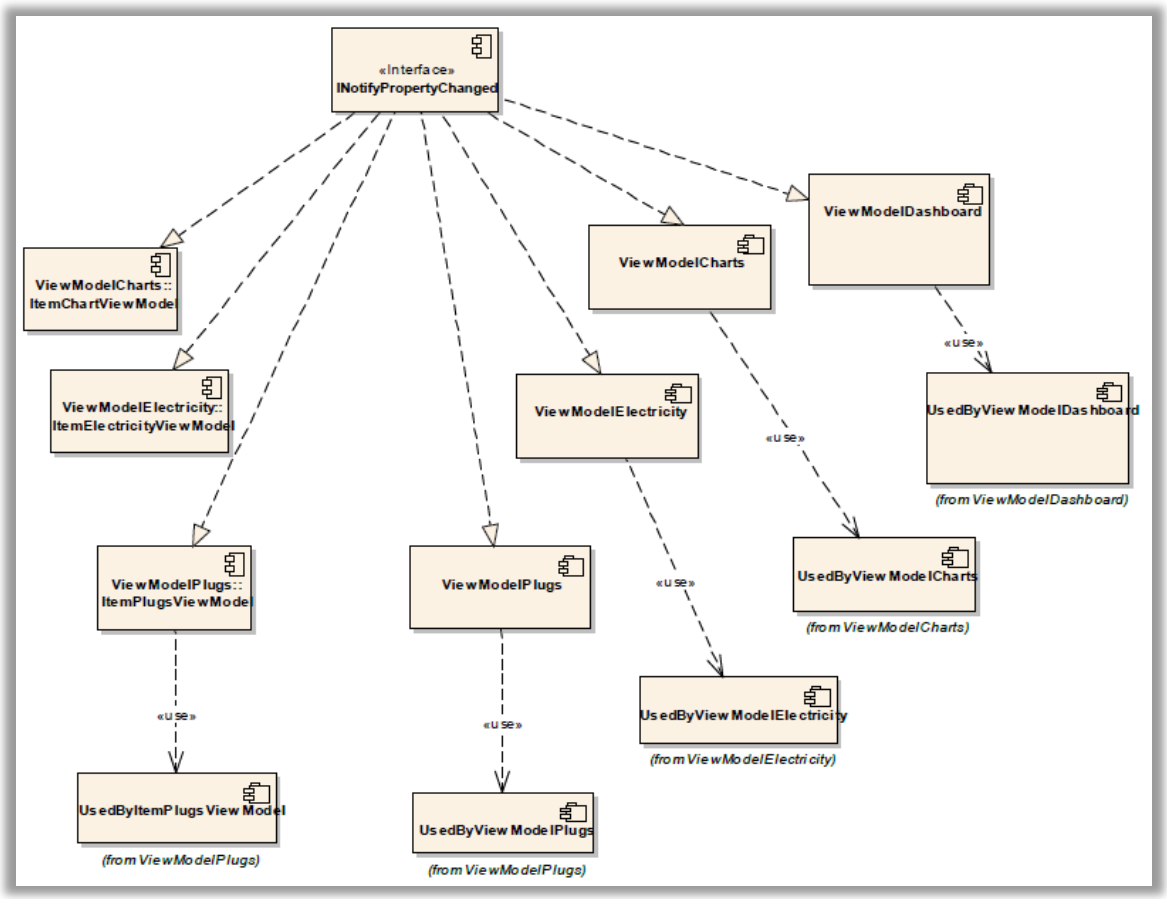


Figura 21 Interface INotifyPropertyChanged implementada nas diversas classes

Sempre que sejam executadas ações na *View* que se traduzem em pedidos à API, o MVVM permite com que a *View* não comunique diretamente com o *Model*, passando esta responsabilidade ao *ViewModel*.

Foi definida a criação de um *ViewModel* para cada *View*. Contudo, a aplicação contém várias vistas que partilham uma grande parte de informação, de modo a não duplicar código.

Este padrão ajudou neste facto, pois a partir de mecanismos de *databinding*, foi possível com que várias vistas acedessem ao mesmo *ViewModel*, ou seja um ponto central dessa informação partilhada.

Uma vez que um dos requisitos da aplicação consistia em iniciar a sessão e registar uma nova conta a partir da rede social Facebook, como apresentado na Figura 22, definiu-se que o código responsável por esta funcionalidade faz parte dos *External Components*.

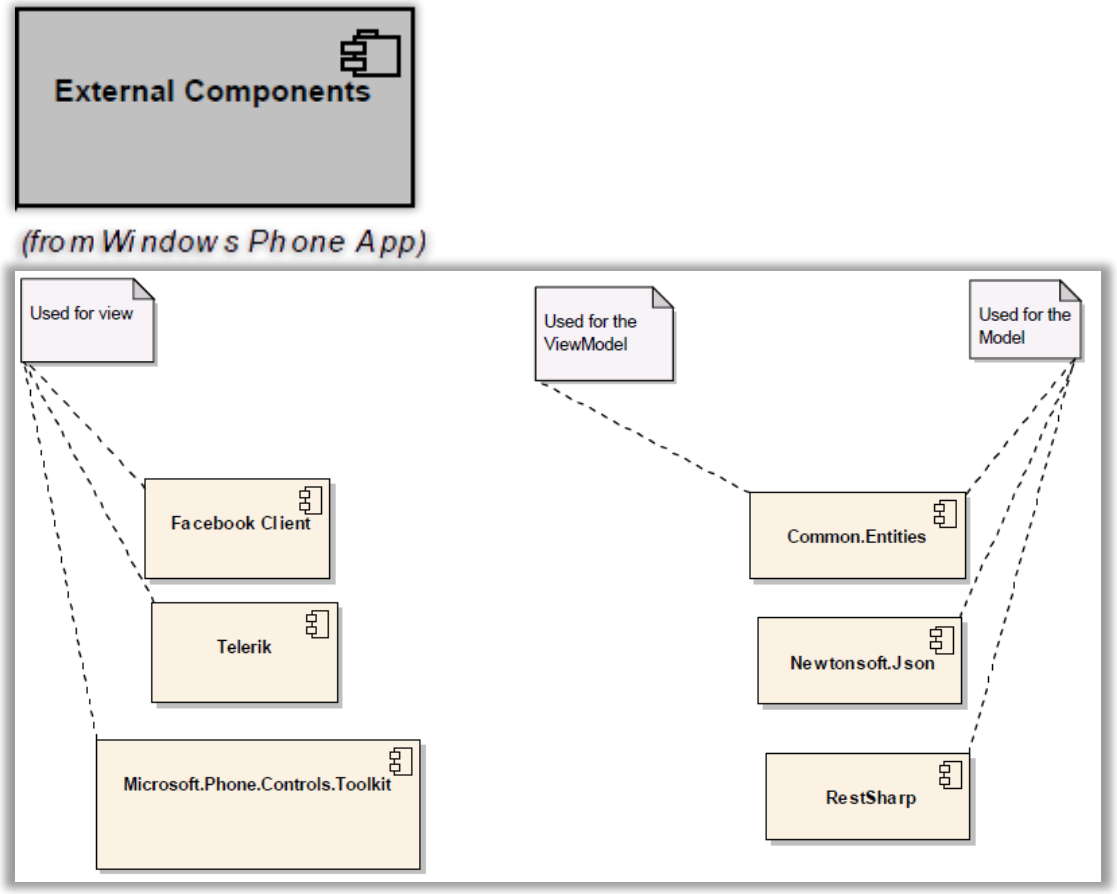


Figura 22 External Components definidos

Além desta funcionalidade, todos os componentes e funcionalidades fornecidas por entidades externas à aplicação, como o Telerik, o Windows Phone Toolkit e as bibliotecas RestSharp e Newtonsoft JSON foram também agrupadas nos *External Components*.

Durante o desenvolvimento da aplicação, levantou-se a necessidade de possuir alguns métodos que são necessários de forma repetida em várias zonas do código. Exemplos disso são converter datas em milissegundo, verificar o estado da ligação à internet, entre outros.

De forma a reutilizar o código responsável por estas funcionalidades, foi definida a classe denominada *Utils* que é responsável por conter todas essas funções.

4.4.2.1. Estratégia adotada para receber consumos energéticos em tempo real

Uma das funcionalidades existentes na aplicação desenvolvida é a visualização de consumos de energia em tempo real.

Esta funcionalidade passa por duas operações, ou seja, dois pedidos à API. O primeiro pedido serve para abrir um canal de comunicação e segundo serve para ir recebendo os consumos de energia.

Inicialmente, para a aplicação receber esses dados, executava-se um primeiro pedido à API para cada dispositivo, mantendo assim tantos canais de comunicação abertos quanto o número de dispositivos que queriam receber dados oriundos da API.

Isto fazia com que houvesse um maior tráfego de comunicação e uma menor autonomia do *smartphone*.

De modo a melhorar o desempenho da aplicação, foi introduzido um novo mecanismo para pedir os consumos de energia em tempo real. Este mecanismo passou por ser criado um ponto central que fizesse este pedido à API, de modo a abrir um único canal de comunicação, indicando previamente todos os dispositivos dos quais se pretende receber consumos.

Depois de receber o sucesso desse pedido vindo da API, é possível receber consumos de energia em tempo real de vários dispositivos, a partir de um único canal de comunicação, melhorando assim o desempenho da aplicação.

4.4.3. Estrutura da aplicação

Antes de elaborar a estrutura da aplicação, foi necessário perceber alguns dos aspetos a desenvolver, nomeadamente quando e onde é usada, o que faz, quais são os seus utilizadores e qual o tipo de conteúdo que deve ser apresentado.

É muito importante conhecer previamente o tipo de conteúdo a apresentar, pois é necessário conhecer a estrutura e modelos de navegação, modos de interação e linhas orientadoras para o uso de controlos na interface do utilizador.

De modo a atingir estes objetivos, foi elaborado um estudo prévio. Este estudo poderá ser consultado no Anexo 2 Windows Phone 8 *guidelines*.

Os três modelos de navegação mais comuns na plataforma Windows Phone 8 consistem no uso de:

- *Panorama Control*;
- Do *Pivot Control;*
- Listas com detalhes *drilldown*.

Desta forma, para que o utilizador possa navegar pelas diferentes áreas da aplicação, depois de ter iniciado a sessão, a informação visualizada é apresentada ao utilizador através dos modelos de navegação anteriormente mencionados.

O *Panorama Control*, apresentado na Figura 23, é usado tipicamente para apresentar múltiplos tipos de informação, não sendo porém recomendado conter mais do que cinco secções diferentes.

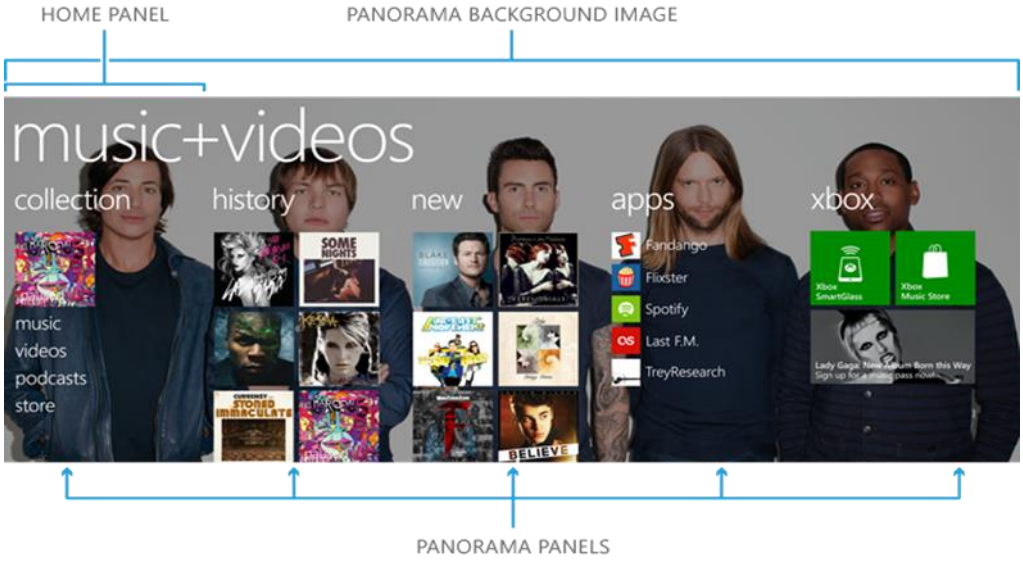


Figura 23 Panorama Control [15]

O *Pivot Control*, apresentado na Figura 24, é parecido com o controlo anterior, no entanto é usado para situações divergentes, pois é usado para conter vários itens, cuja informação seja do mesmo tipo.

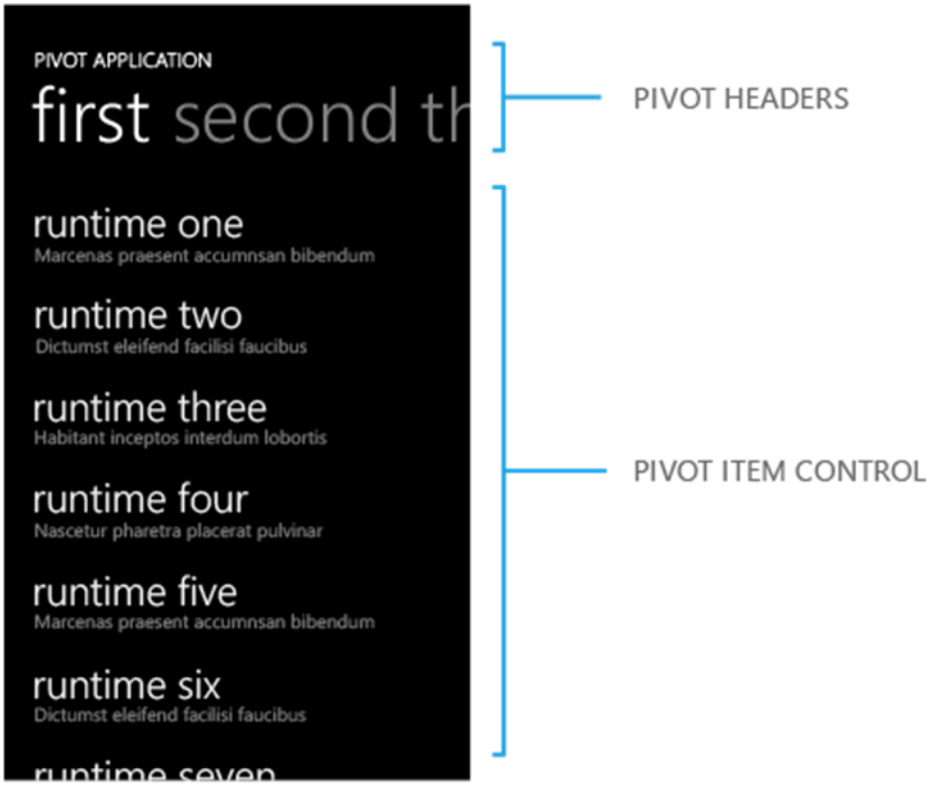


Figura 24 Pivot Control [16]

É preciso ter em atenção que o uso do botão *back* do dispositivo nos controlos anteriormente descritos, não funciona para voltar para a secção anterior do *Panorama Control*, ou do item anterior do *Pivot Control*, mas sim para voltar para uma área diferente da aplicação ou mesmo para o seu término.

Nas Listas com detalhes *drilldown*, apresentado na Figura 25, cada entrada na lista redireciona para um ecrã de detalhes.

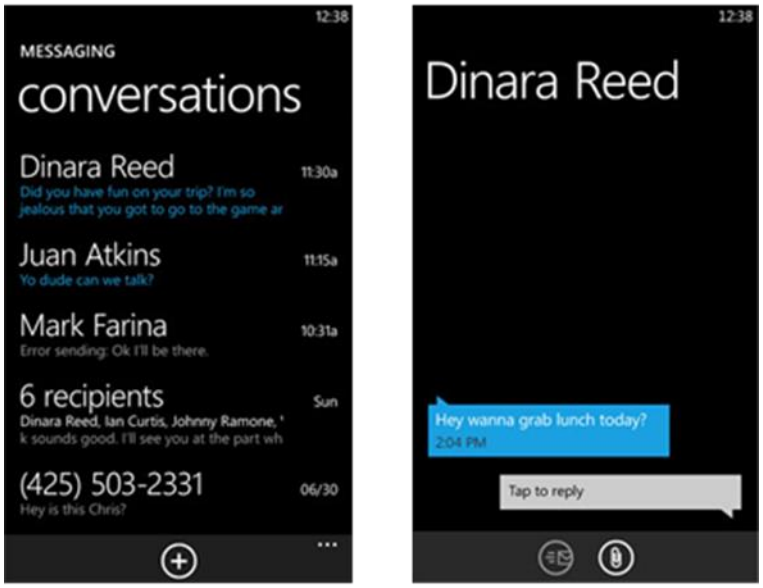


Figura 25 Lista com detalhes em drilldown [17]

Durante a navegação da aplicação, são visualizadas várias transições, ou seja, animações da navegação entre os diferentes ecrãs.

Em qualquer altura, o utilizador poderá sair da aplicação, colocando-a em suspensão. Para isto deverá pressionar a tecla *Windows.*

A Figura 26 retrata a estrutura da solução final da aplicação.

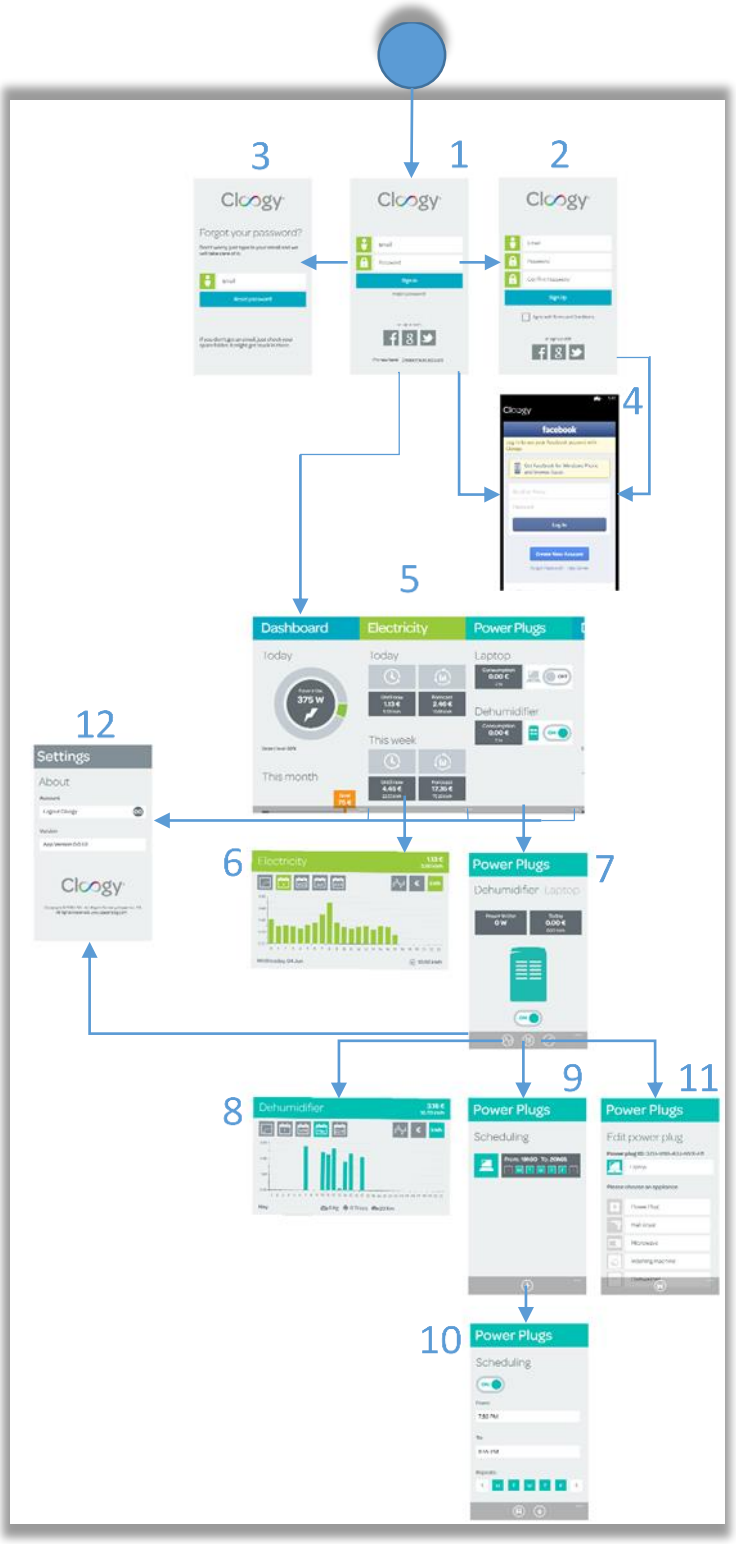


Figura 26 Estrutura da aplicação (5 – Panorama Control; 7 – Pivot Control; 9 – Lista com detalhes drilldown)

De seguida apresenta-se o detalhe de cada um dos ecrãs.

4.4.3.1. Ecrã de login (1)

O ecrã de login, apresentada na Figura 27, permite ao utilizador iniciar a sessão na aplicação.

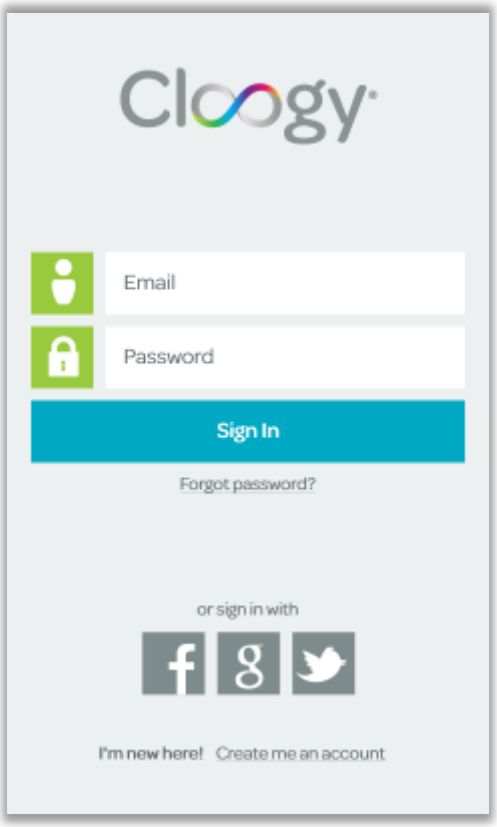


Figura 27 ecrã de login

Este ecrã é apresentado sempre que o utilizador, ao entrar na aplicação, tenha acesso à internet e não tenha sessão iniciada.

Sempre que as credenciais de início de sessão não estejam corretas ou em caso de indisponibilidade do lado do servidor, o utilizador é notificado com uma *Message Box* da respetiva mensagem.

Deste ecrã, o utilizador pode navegar para os ecrãs de:

- Recuperação da palavra-passe;
- Criação de uma nova conta;
- Iniciar sessão através do Facebook.

Apesar de existirem disponíveis no ecrã a utilização do Twitter e do Google+, esta funcionalidade não se encontra desenvolvida do lado do servidor, aparecendo uma mensagem de indisponibilidade da funcionalidade.

Por vezes é necessário mudar o endereço do servidor da aplicação para usufruir e testar novas funcionalidades. Para isto, foi adicionado um controlo ao logotipo do Cloogy, nomeadamente um *ContextMenu*.

Este controlo permite ao utilizador escolher o respetivo servidor, pressionando durante dois segundos no logotipo da aplicação, para que os servidores sejam apresentados.

Sempre que o utilizador pressionar a tecla *back* do dispositivo no presente ecrã, a aplicação termina.

4.4.3.2. Ecrã de criação de uma nova conta (2)

A Figura 28 apresenta o ecrã para criação de uma nova conta. Para este efeito, o utilizador necessita de preencher o email, palavra-passe e de aceitar os termos e condições do Cloogy.

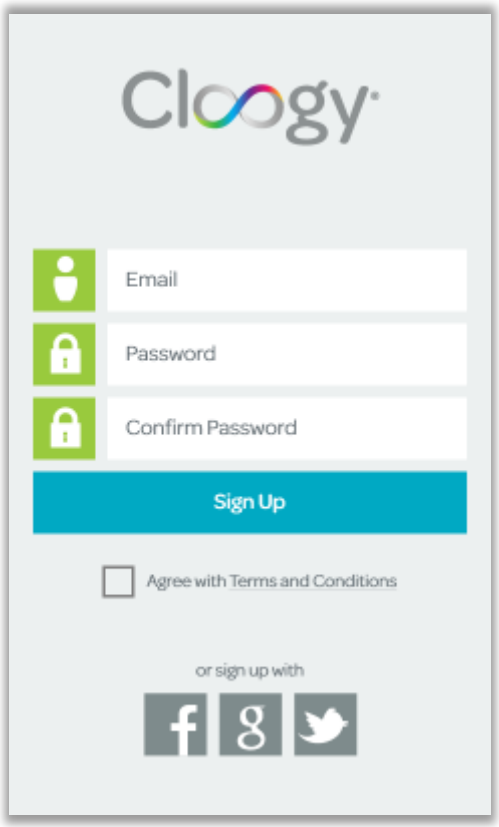


Figura 28 Ecrã de criação de uma nova conta

É apresentada uma mensagem ao utilizador sempre que:

- O email seja inválido ou que já exista no sistema;
- A palavra-passe seja demasiado curta;
- A palavra-passe de confirmação não corresponda à palavra-passe introduzida anteriormente;
- O utilizador não aceite os termos e condições do Cloogy;
- Indisponibilidade por parte do sistema;
- Sucesso da criação da nova conta.

Deste ecrã, o utilizador pode navegar para os ecrãs:

- De login;
- De criação de uma nova conta a partir do Facebook;
- Das definições da aplicação.

Sempre que o utilizador pressionar a tecla *back* do dispositivo, a aplicação voltará para o ecrã de *login*.

4.4.3.3. Ecrã para recuperação da palavra-passe (3)

A Figura 29 apresenta o ecrã para recuperação da palavra-passe. Para este efeito, o utilizador necessita apenas de preencher o seu email.

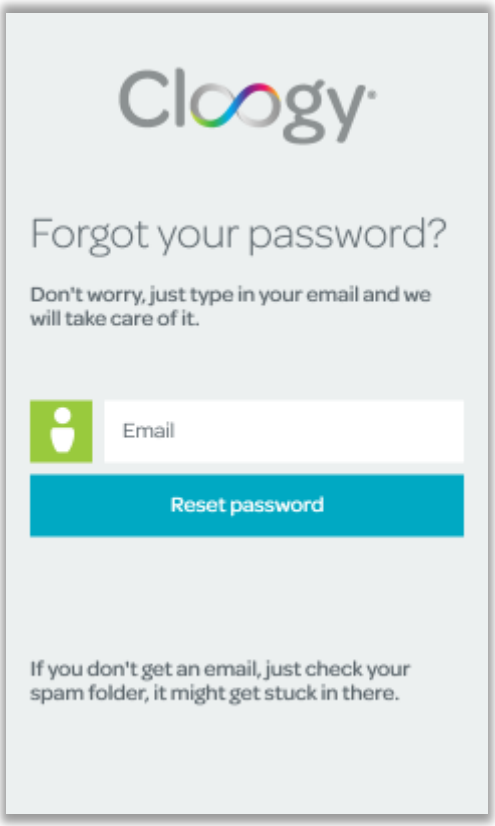


Figura 29 Ecrã para recuperação da palavra-passe

É apresentada uma mensagem ao utilizador sempre que:

- O email seja inválido ou que não exista no sistema;
- Indisponibilidade por parte do sistema;
- Sucesso na recuperação da palavra-passe.

Deste ecrã, o utilizador pode navegar apenas para o ecrã de login, através de dois cenários diferentes:

- O utilizador pressiona a tecla *back* do dispositivo;
- A operação de recuperar a palavra-passe é bem-sucedida, sendo o utilizador redirecionado automaticamente para o ecrã de login, após a mensagem de notificação.

4.4.3.4. *Login* e registo a partir do Facebook (4)

Para início de sessão e para registo de uma nova conta no Cloogy, o utilizador poderá utilizar a sua conta do Facebook, como apresentado na Figura 30.

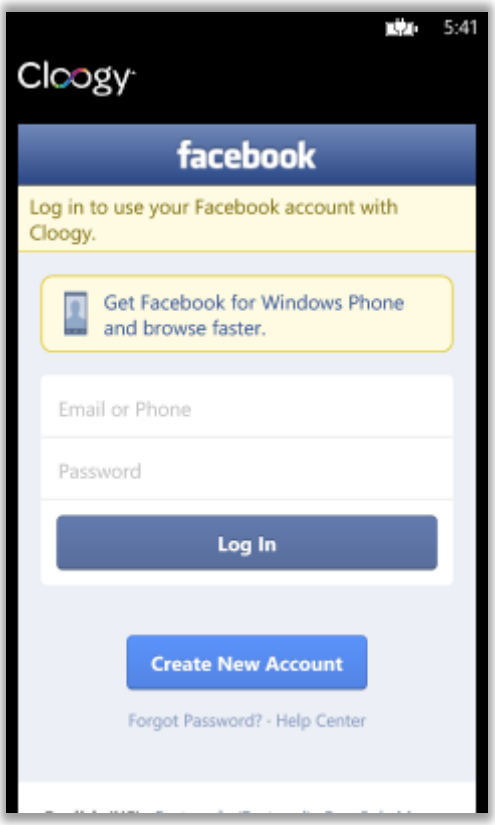


Figura 30 Login e registo de uma nova conta a partir do Facebook

Deste ecrã o utilizador poderá navegar para os ecrãs:

- De login com a tecla *back* do dispositivo caso a sua navegação tenha sido oriunda deste ecrã;
- De criação de uma nova conta com a tecla *back* do dispositivo caso a sua navegação tenha sido oriunda deste ecrã;
- *Dashboard*, depois do sucesso de início de sessão.

Em caso de insucesso das operações, é apresentada uma *Message Box* ao utilizador a fim de o notificar.

4.4.3.5. *Panorama Control* para visualização da *Dashboard*, *Electricity* e *Power Plugs* (5)

O *Panorama Control* permite ao utilizador reter parte da informação acerca do seu Cloogy, de uma forma bastante rápida. Como apresentada na Figura 31, esta informação encontra-se dividida em três secções, nomeadamente:

- *Dashboard* (Visão Geral);
- *Electricity* (Eletricidade);
- *Power Plugs* (Tomadas inteligentes).

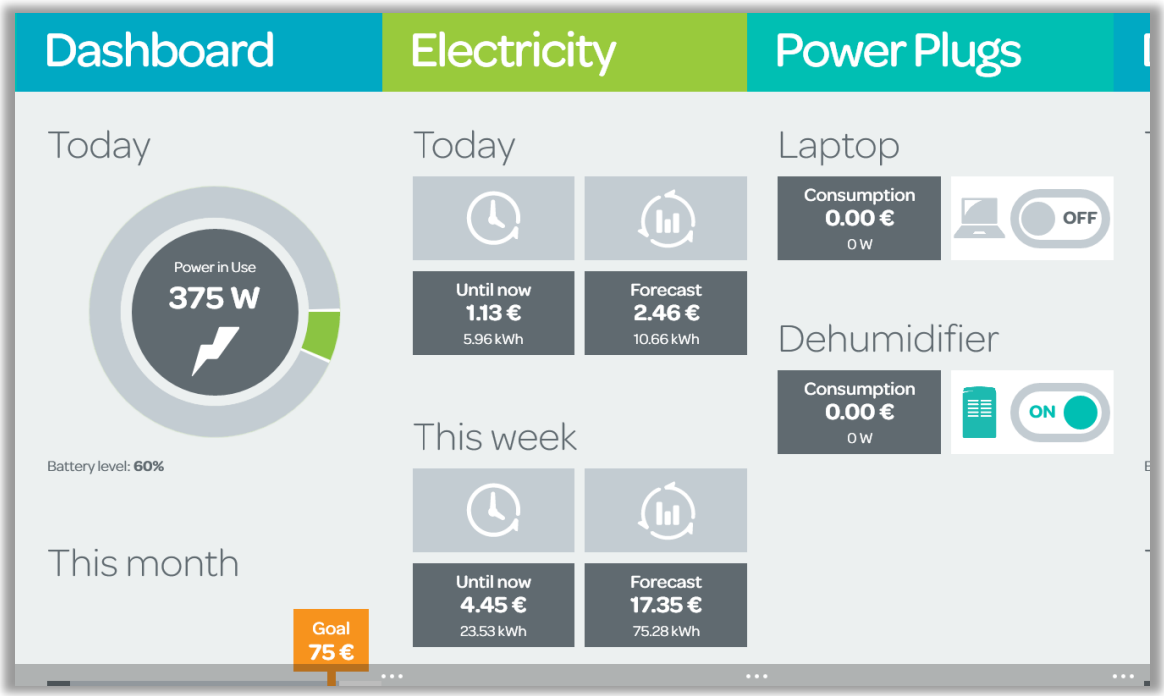


Figura 31 Panorama Control para visualização da Dashboard, Eletricity e Power Plugs

A Visão Geral encontra-se dividida em duas secções. A primeira secção, em cima, permite ao utilizador visualizar a sua potência energética global em tempo real, através de um medidor circular. Esta secção disponibiliza também a informação da percentagem de bateria do dispositivo do utilizador. A segunda secção, em baixo, apresenta ao utilizador um *widget* infográfico com base nos seguintes fatores:

- Custo energético mensal gasto até ao momento;
- Previsão do custo energético mensal;
- Objetivo do valor mensal inserido pelo utilizado.

A Eletricidade contém os consumos e previsões em euros e kWh dos valores energéticos gastos para a seguinte lista:

- Dia atual;
- Semana atual;
- Mês atual;
- Ano atual.

Esta secção contém informação *drilldown*, ou seja, sempre que o utilizador selecionar uma determinada área desta secção, será redirecionado para o gráfico correspondente ao tipo de granularidade selecionado. Este aspeto encontra-se descrito no capítulo 4.4.3.6.

A secção das Tomadas Inteligentes apresenta uma lista com todas as tomadas inteligentes que o utilizador possui no seu *Kit* Cloogy. Em cada tomada é apresentado o custo em euros gasto no dia e o valor da potência energética em tempo real. Também é fornecido ao utilizador um botão para ligar ou desligar a tomada correspondente.

Cada tomada é selecionável, redirecionando assim o utilizador para um controlo diferente, nomeadamente um *Pivot Control*, mencionado no capítulo 4.4.3.7.

Todas as secções deste controlo contém uma *application bar*, cujo objetivo é redirecionar o utilizador para o ecrã das definições da aplicação.

Sempre que o utilizador pressionar a tecla *back* do dispositivo, a aplicação é terminada.

4.4.3.6. Ecrã para visualizar os consumos energéticos graficamente (6)

O gráfico presente na Figura 32 é apresentado ao utilizador sempre que este faça *drilldown* na secção *Electricity* no *Panorama Control*.

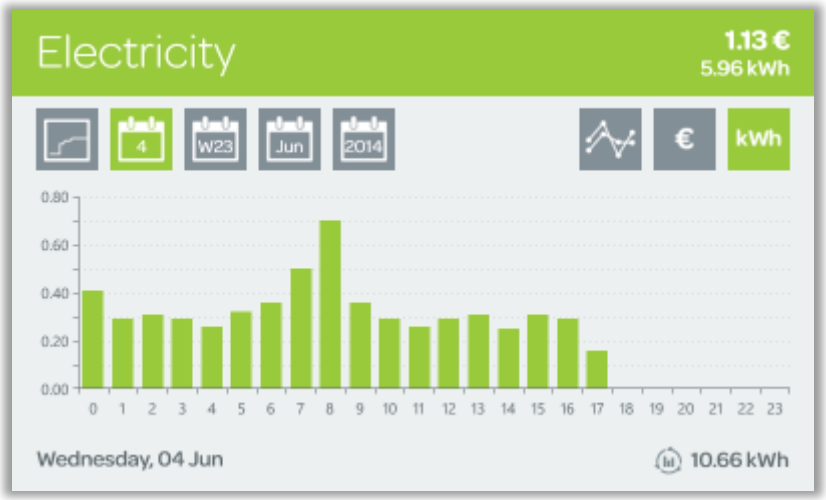


Figura 32 Visualização gráfica dos consumos energéticos

Este gráfico permite ao utilizador visualizar graficamente todo o histórico dos consumos energéticos, podendo navegar no tempo, arrastando o dedo sobre o gráfico.

Esta vista permite ao utilizador visualizar a potência energética global em tempo real, os consumos e previsões em euros e kWh do dia, semana, mês e do ano.

No caso de o utilizador visualizar os consumos de um mês ou de um ano, é apresentado a sua pegada ecológica.

Sempre que o utilizador esteja a visualizar o período atual (o dia atual, a semana atual, o mês atual ou o ano atual), é apresentada a previsão de consumo até ao final do período correspondente.

Neste ecrã, como se pode observar na Figura 33, também é fornecido ao utilizador um botão para comparar consumos. Estes consumos são comparados com base na data escolhida, com um período anterior, dependente do tipo de granularidade escolhida.

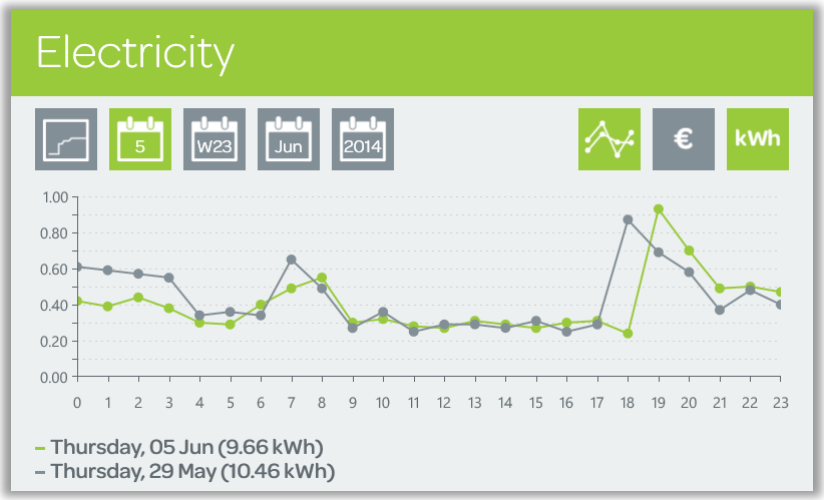


Figura 33 Comparação do consumo energético do dia atual com o dia correspondente da semana anterior

Por exemplo, no caso do dia selecionado ser o dia atual, este é comparado com o dia correspondente da semana anterior.

Sempre que este modo é escolhido, a pegada ecológica deixa de ser apresentado ao utilizador.

Sempre que o utilizador pressionar a tecla *back* do dispositivo, é redirecionado para a secção *Electricity* do *Panorama Control*, independentemente do tipo de granularidade escolhida.

4.4.3.7. *Pivot Control* para visualização das tomadas inteligentes (7)

Como apresentado na Figura 34, o *Pivot Control* fornecido permite apresentar todas as Tomadas Inteligentes do utilizador. Este controlo é apresentado ao utilizador sempre que este faça *drilldown* na secção *Power Plugs* no *Panorama Control*.

Figura 34 Pivot Control para visualização das tomadas inteligentes

Este controlo também permite ligar e desligar a tomada escolhida.

Esta vista contém uma *application bar* de modo que o utilizador possa:

- Visualizar os consumos graficamente;
- Editar tomadas inteligentes;
- Criar/Editar/Eliminar agendamentos;
- Navegar para o ecrã das definições da aplicação.

Sempre que o utilizador pressionar a tecla *back* do dispositivo, é redirecionado para a secção *Power Plugs* do *Panorama Control*.

4.4.3.8. Ecrã para visualizar os consumos energéticos graficamente das tomadas inteligentes (8)

Como apresentado na Figura 35, o gráfico das tomadas inteligentes contém o mesmo funcionamento do que o gráfico utilizado para visualizar os consumos energéticos da secção *Electricity*, residente no *Panorama Control* mencionado no capítulo 4.4.3.6.

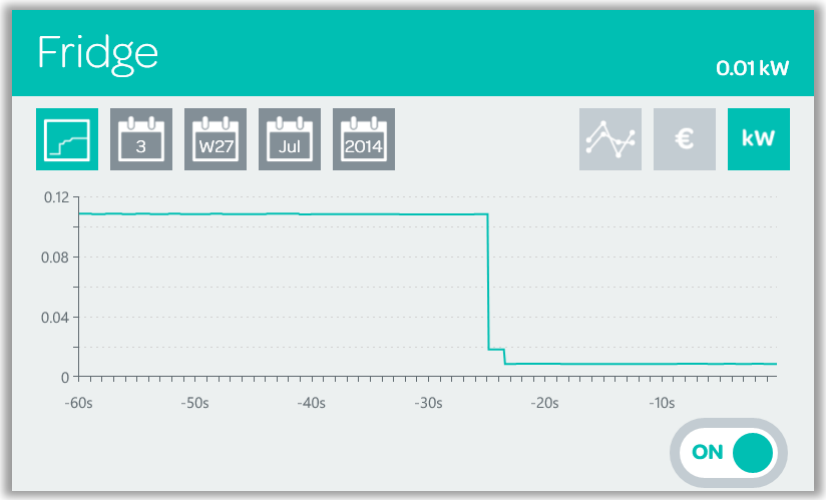


Figura 35 Visualização gráfica do consumo energético em tempo real de uma tomada inteligente

Em relação ao gráfico utilizado na secção *Electricity,* o presente gráfico contém três alterações:

- As cores foram mudadas de modo a corresponder às cores usadas na área das tomadas inteligentes;
- Foi adicionado um botão para ligar e desligar a tomada inteligente na área de visualização da potência energética da tomada em tempo real;
- Sempre que o utilizador pressionar a tecla *back* do dispositivo, é redirecionado para o *Pivot Control* para visualização das tomadas inteligentes, mencionado no capítulo 4.4.3.7.

4.4.3.9. Ecrã para visualizar os agendamentos das tomadas inteligentes (9)

Como apresentado na Figura 36, o presente ecrã contém uma lista dos agendamentos de uma tomada estabelecidos pelo utilizador.

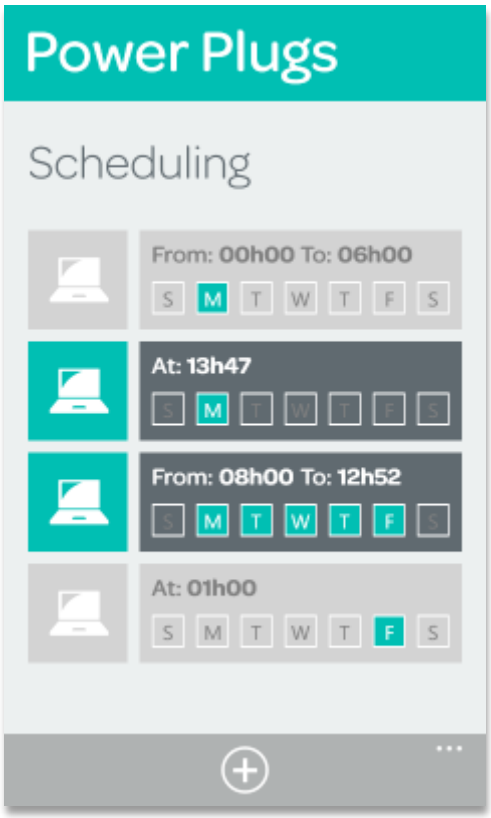


Figura 36 Visualização dos agendamentos de uma tomada inteligente

O presente cenário apresenta quatro agendamentos diferentes. Os agendamentos definidos para ligar uma tomada, são apresentados sob as cores azul e cinzento-escuro. Os agendamentos definidos para desligar uma tomada são apresentados sob cinzento-claro.

Quando o agendamento tem apenas a ação de ligar ou desligar, aparece a hora que vai ocorrer esse mesmo agendamento. Quando o agendamento tem uma ação (ligar ou desligar) e a sua ação contrária, então aparecem as horas de ambas as ações.

Sempre que se defina uma repetição do agendamento, é retratado a partir dos quadrados azuis, correspondentes ao dia da semana.

A *application bar* fornecida ao utilizador tem o objetivo de criar um novo agendamento, sendo redirecionado para o ecrã responsável por essa ação.

Sempre que o utilizador selecionar um agendamento, é redirecionado para o ecrã responsável por editar e eliminar o agendamento correspondente, mencionado no capítulo 4.4.3.10.

Caso o utilizador pressione na tecla *back*, é redirecionado para o *Pivot Control* mencionado no capítulo 4.4.3.7.

4.4.3.10. Ecrã para criar/eliminar/atualizar os agendamentos das tomadas inteligentes (10)

O ecrã apresentado na Figura 37 permite criar, atualizar e eliminar agendamentos.

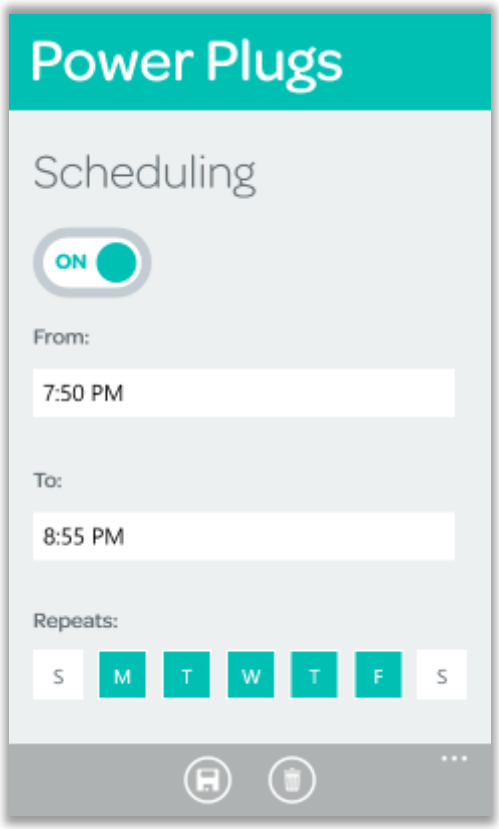


Figura 37 Ecrã para criar, editar e eliminar um agendamento

Caso o utilizador tenha sido redirecionado para o presente ecrã, através da *application bar* presente no ecrã da listagem dos agendamentos, mencionado no capítulo 4.4.3.9, o botão de eliminar da *application bar* deste ecrã fica inativo, permitindo assim criar um novo agendamento.

Caso contrário, significa que o utilizador selecionou um agendamento, a fim de o editar ou eliminar, ficando os botões de guardar e eliminar, ativos na *application bar* deste ecrã.

O utilizador é notificado com uma *Message Box* com determinadas mensagens no caso:

- De não preencher corretamente os detalhes do agendamento;
- De indisponibilidade do sistema;
- De aviso do agendamento ocorrer apenas numa determinada data.

Sempre que o botão *back* do dispositivo seja selecionado, o utilizador é redirecionado para o ecrã anterior, ou seja, para a listagem dos agendamentos, mencionado no capítulo 4.4.3.9.

4.4.3.11. Ecrã para editar as tomadas inteligentes (11)

O ecrã apresentado na Figura 38 é fornecido ao utilizador a fim que este possa editar o nome e o eletrodoméstico associado à tomada inteligente correspondente.

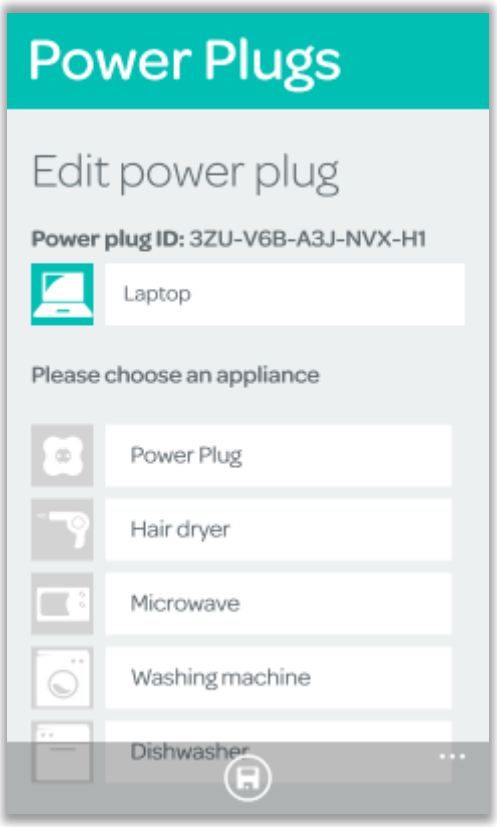


Figura 38 Ecrã para edição de tomadas inteligentes

Sempre que o utilizador pressionar a tecla *back* do dispositivo, será redirecionado para o *Pivot Control* mencionado no capítulo 4.4.3.7.

4.4.3.12. Ecrã das definições da aplicação (12)

A Figura 39 retrata as definições da aplicação.

Figura 39 Ecrã das definições da aplicação

Este ecrã permite ao utilizador:

- Terminar a sua sessão da aplicação, sendo redirecionado para o ecrã de login;
- Visualizar a versão da aplicação;
- Aceder ao *link* para o *website* da ISA.

Sempre que o utilizador pressione a tecla *back* do dispositivo, é redirecionado para o ecrã anterior.

4.4.4. Utilização de *Resources*

Uma das necessidades da aplicação é ser multilingue, sendo este o principal requisito para utilização de *Resources*.

A tecnologia utilizada permite criar ficheiros denominados *Resources*, cuja extensão é *.resx.* Estes ficheiros consistem em entradas XML, podendo ser abertos com um editor de texto normal, de modo a serem manipulados.

Neste projeto, estes ficheiros foram utilizados de forma a alojarem vários tipos de informação:

- Propriedades de controlos, como definição de cores, tipos de letra e formatos de datas;
- Endereços dos vários ambientes da API;
- *Strings* a serem visualizadas pelo utilizador.

Tanto o código em C# como XAML são capazes de aceder às entradas dos *Resources,* podendo assim ter acesso à informação atrás referida. Desta forma, sempre que seja necessário mudar esta informação, não é necessário editar o código, apenas o *resource* correspondente.

A tecnologia usada permite facilmente apresentar as *strings* ao utilizador na sua língua, apenas sendo necessário criar um *resource* com um nome específico, para cada cultura que a aplicação pretende suportar, podendo assim facilmente suportar línguas adicionais.

Posteriormente é necessário, para cada entrada XML atribuir o mesmo nome dos atributos dos ficheiros anteriores com a respetiva tradução para a cultura desejada. A aplicação carregará automaticamente o *resource* responsável por apresentar os textos na sua cultura.

Este mecanismo poderá ser consultado no Anexo 3, no capítulo 3.3.

4.4.5. Necessidade de criação de *User Controls*

A aplicação desenvolvida contém controlos personalizados em várias vistas. De forma a reutilizar esses controlos sem ter que repetir código, a tecnologia utilizada permite criar componentes chamados *User Controls*.

Um *User Control* é um controlo próprio que apenas é criado uma única vez e que se pode reutilizar sempre que seja necessário. Neste projeto foram criados os seguintes controlos:

- *Loading* da aplicação;
- *Widget* do objetivo mensal residente na *dashboard*;
- Botão personalizado para ligar e desligar tomadas inteligentes do Cloogy.

4.4.5.1. *Loading* da aplicação

De modo a dar a ideia de carregamento de dados ao utilizador nas operações mais demoradas, foi criado o presente *User Control*. A Figura 40 retrata o *loading* da aplicação.

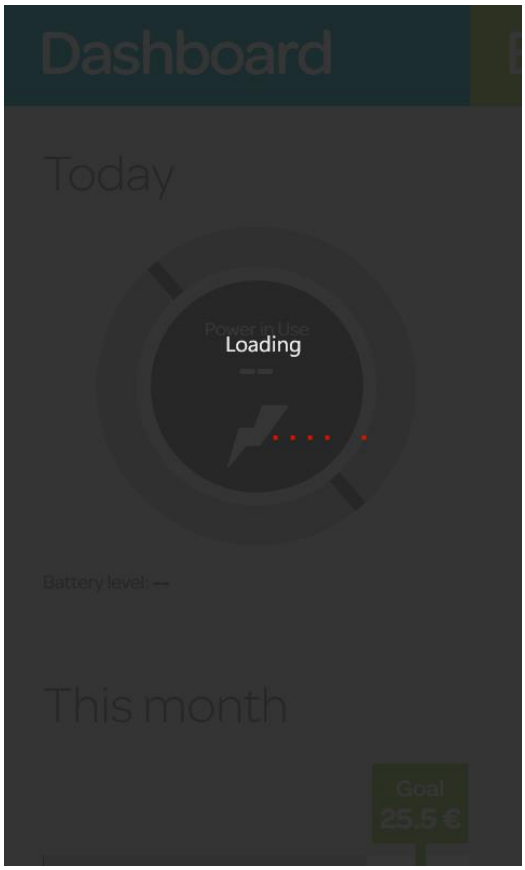


Figura 40 Loading da aplicação quando se inicia a dashboard

Sempre que é necessário efetuar uma operação demorada, este controlo é apresentado ao utilizador, na cultura Inglesa ou Portuguesa, dependente da sua cultura, fazendo uso de *Resources*.

4.4.5.2. *Widget* para o objetivo mensal

A Tabela 16 apresenta os três estados normais do *Widget* do objetivo mensal, assim como a descrição de cada um.

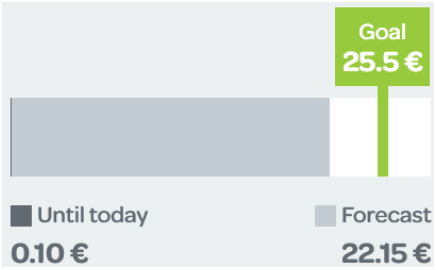
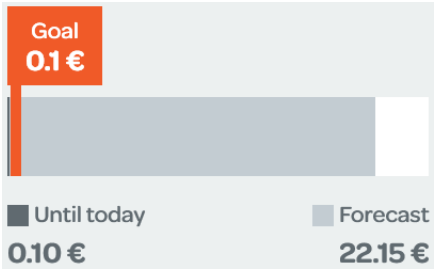
Widget	Descrição
	O objetivo mensal em euros, definido pelo utilizador, encontra-se acima da previsão efetuada pelo sistema e do valor consumido atualmente.
	O objetivo mensal em euros, definido pelo utilizador, encontra-se acima do valor consumido atualmente. No entanto a previsão efetuada pelo sistema indica que o utilizador irá gastar mais do que o seu objetivo.
	O consumo energético em euros ultrapassou o objetivo do utilizador.

Tabela 16 Estados do widget do objetivo mensal na dashboard

Sempre que o utilizador selecionar a área do objetivo mensal, é apresentada uma *dialog* e um teclado numérico, de forma a poder definir o seu objetivo mensal em euros.

4.4.5.3. Botão para ligar/desligar tomadas inteligentes

A Tabela 17 apresenta os quatro estados do botão criado para ligar e desligar tomadas inteligentes.

Botão	Descrição
	Tomada inteligente ligada.
	Pedido ao sistema para desligar a tomada inteligente. Enquanto o sistema não devolver resposta, o botão fica no estado inativo.
	Tomada inteligente desligada.
	Pedido ao sistema para ligar a tomada inteligente. Enquanto o sistema não devolver resposta, o botão fica no estado desativo.

Tabela 17 Estados do botão para ligar e desligar tomadas inteligentes

Todas as transições dos quatro estados do presente *User Control* são efetuadas com a ajuda de *storyboards*[33] para proceder às animações das transições.

4.4.6. Testes e resultados obtidos

Este capítulo tem como objetivo apresentar os testes efetuados à aplicação desenvolvida, de forma a detetar possíveis falhas das funcionalidades desenvolvidas.

[33] Storyboard – Representação visual da animação de determinado *User Control.*

4.4.6.1. Testes unitários

Ao longo do desenvolvimento deste projeto, foram implementados vários testes unitários. Estes testes permitiram testar cada método individualmente.

Foram criados métodos de forma a fornecer uma determinada entrada ao método a testar e averiguar se a saída é a esperada. Caso o resultado do método seja diferente do esperado, é descoberta assim uma falha interna, podendo ser corrigido de modo a não se propagar pelo resto da aplicação.

Para realizar testes unitários, é necessário adicionar um novo *Unit Test App Project* à solução existente. Como é mostrado na Figura 41, é necessário adicionar os atributos:

- [TestClass] para cada *class* criada;
- [TestMethod] para cada atributo a testar.

```
[TestClass]
public class UnitTest1
{
    Util util = new Util();

    [TestMethod]
    public void IsValidEmail1()
    {
        Assert.IsTrue(util.IsValidEmail("xpto_asd_@xxxx.com"));
    }
    [TestMethod]
    public void IsValidEmail2()
    {
        Assert.IsFalse(util.IsValidEmail("xpto_asd_xxxx.com"));
    }
    [TestMethod]
    public void IsValidEmail3()
    {
        Assert.IsFalse(util.IsValidEmail("@xxxx.com"));
    }
    [TestMethod]
    public void IsValidEmail4()
    {
        Assert.IsFalse(util.IsValidEmail("a@a."));
    }
```

Figura 41 Exemplo de testes unitários

Depois de serem adicionados os atributos anteriormente mencionados, é necessário criar as respetivas funções para testar os métodos da aplicação. Estes testes são feitos com a ajuda de métodos de uma classe do sistema, nomeadamente a *class Assert*. Os métodos mais utilizados são:

- *AreEqual;*
- *AreNotEqual;*
- *Equals;*
- *IsNotNull;*
- *IsNul;*
- *IsFalse;*
- *IsTrue.*

Como apresentado na Figura 42, foram realizados 76 testes unitários aos diversos métodos da aplicação, dos quais todos passam com sucesso.

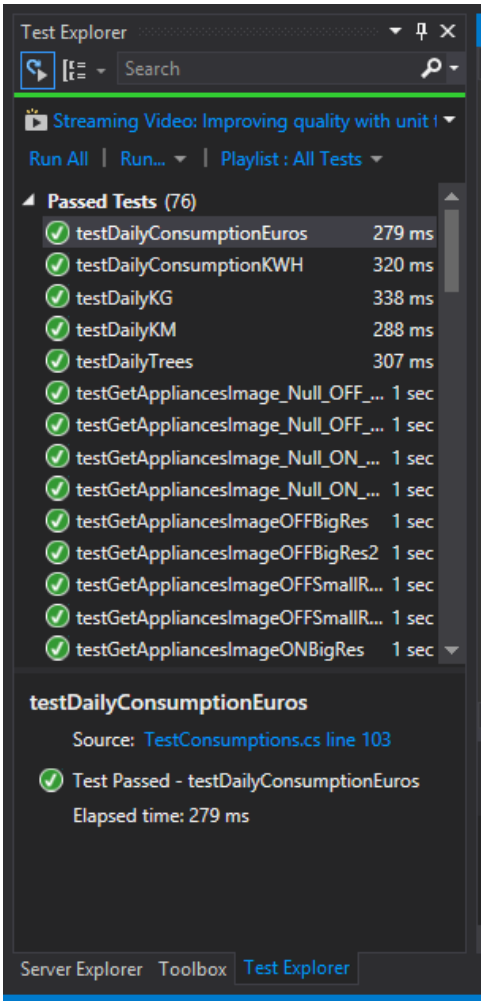


Figura 42 Resultado dos testes unitários efetuados

A realização destes testes permitiram detetar falhas em alguns métodos da aplicação, podendo assim serem corrigidos.

4.4.6.2. Testes de aceitação

Durante o desenvolvimento da aplicação, no fim de cada *sprint*, nomeadamente na *Sprint Review*, foram realizados constantes testes de aceitação. Testes estes efetuados pela equipa do Cloogy, de forma a averiguar que todas as *user stories* fossem implementadas de acordo com o definido, sendo realizados vários testes para a mesma *user story*.

Estes testes permitiram assim validar e verificar as funcionalidades implementadas em cada *sprint*, garantindo que a aplicação fosse capaz de executar as funcionalidades acordadas.

4.4.6.3. Publicação da aplicação na Windows Phone Store

De modo a que os utilizadores do Cloogy possam usufruir da aplicação desenvolvida neste projeto, a mesma foi publicada na Windows Phone Store, uma plataforma de distribuição digital de aplicações para Windows Phone, onde os seus utilizadores podem fazer *download* das aplicações desejadas.

Antes de publicar uma aplicação na Windows Phone Store, é necessário perceber o seu processo de funcionamento.

O primeiro passo deste processo consiste na criação de uma conta Microsoft e registá-la como uma conta cujo perfil seja de programador. Posteriormente, é necessário fazer o *upload* e descrever a aplicação. Este passo consiste no *upload* do pacote XAP (ficheiro que contém o ficheiro executável da aplicação, assim como todos os recursos necessário para o bom funcionamento da mesma), onde deve ser garantido que este pacote contenha os *tiles*[34] da aplicação, assim como imagens promocionais e respetivos *screenshots*.

É igualmente necessário dar um nome à aplicação, indicar a categoria onde se insere, quais os idiomas de suporte, qual mercado de destino, custo da aplicação, e o modo como esta irá ser publicada, se de forma automática ou manual.

[34] *Tile* – Imagem que representa uma aplicação no ecrã inicial do Windows Phone.

Caso a aplicação seja publicada automaticamente, esta é publicada assim que a certificação seja obtida. Na publicação manual, a aplicação é publicada quando o programador assim o entender depois da obtenção da certificação da aplicação.

De modo a facilitar a pesquisa da aplicação na Windows Store, é imprescindível que a aplicação contenha *keywords* e uma boa descrição. Esta descrição deve ser única, sucinta e prática, de modo a cativar o utilizador a fazer o *download* da aplicação.

Posteriormente pode efetuar-se a submissão da aplicação para a plataforma. Neste passo, a aplicação é submetida a um conjunto de testes realizados pela Microsoft, com o propósito de obter uma certificação garantindo assim que a aplicação reúne todas as condições necessárias para ser publicada.

Como apresentado na Tabela 18, a aplicação submetida pode passar por dez estados bem definidos:

	Estado da submissão	Descrição
1.	*Not completed*	Submissão iniciada mas a aplicação ainda não está submetida.
2.	*Processing submission*	A Microsoft começou a processar a submissão da aplicação.
3.	*XAP processing failed*	Um ou mais ficheiros XAP da submissão falharam na validação. Necessário corrigir os erros e submeter um novo XAP.
4.	*In signing stage*	Os ficheiros XAP da submissão estão a ser assinados para que possam ser confiáveis no Windows Phone.
5.	*Signing passed* ou	Os ficheiros XAP da submissão foram assinados com sucesso.
	Signing failed ou	A assinatura de um ou mais ficheiros XAP falhou.
	Malware detected	A assinatura de um ou mais ficheiros XAP falhou, devido à deteção de *malware*.
6.	*Pending certification*	A submissão da aplicação foi submetida para certificação.
7.	*Certification passed* ou	A submissão da aplicação passou com sucesso por meio da certificação.
	Certification failed	A submissão da aplicação falhou na certificação. Poder-se-á ver os detalhes na página Dev Center.
8.	*Ready to be published*	A submissão passou na certificação e está pronta para ser publicada aos utilizadores.
9.	*Published*	A submissão foi publicada. Poderá demorar até 24 horas para que a aplicação esteja disponível na Windows Phone Store
10.	*Submission canceled*	A submissão foi cancelada.

Tabela 18 Estados da submissão de uma aplicação na WIndows Phone Store

De forma a diminuir a probabilidade da submissão da aplicação falhar na certificação, atenuando possíveis falhas dessa submissão, existe um conjunto de testes que se podem efetuar, de modo a aumentar o grau de confiança da submissão da aplicação. Estes testes estão descritos na documentação da Microsoft.

Esta entidade disponibiliza juntamente com o IDE Visual Studio, uma ferramenta que fornece um conjunto de testes automatizados e manuais, a Store Test Kit. Esta ferramenta ajuda a

identificar problemas existentes na aplicação e que sejam resolvidos antes da submissão da aplicação na Store. Isto leva com que não se desperdice tempo no processo de submissão, pois os testes ajudam a preparar a aplicação de modo a ser aceite na Windows Phone Store, assim que seja submetida.

Como se pode visualizar na Figura 43, os testes automáticos da ferramenta Store Test Kit permitem validar o tamanho do ficheiro XAP e o conteúdo dos restantes ficheiros, validar os ícones da aplicação e os seus *screenshots*.

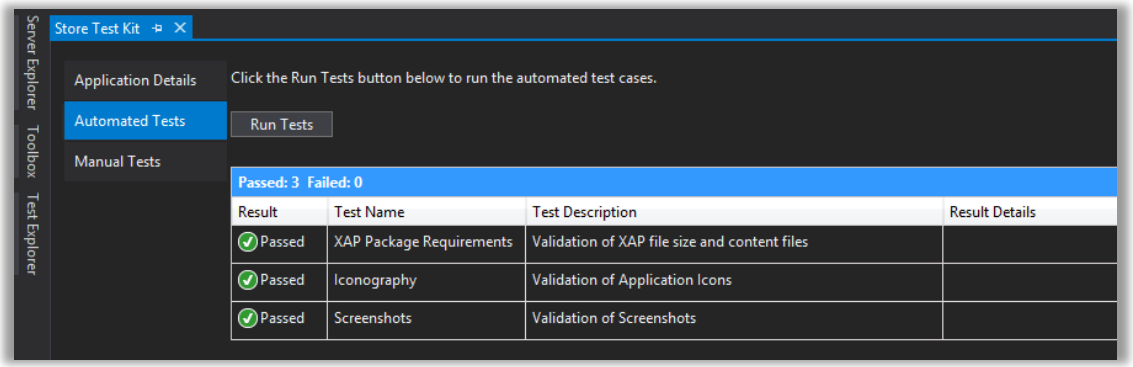


Figura 43 Testes automáticos da ferramenta Store Test Kit

Como apresentado na Figura 44, os testes manuais da ferramenta Store Test Kit permitem efetuar um conjunto de cenários, de modo a verificar possíveis falhas da aplicação ou situações imprevistas.

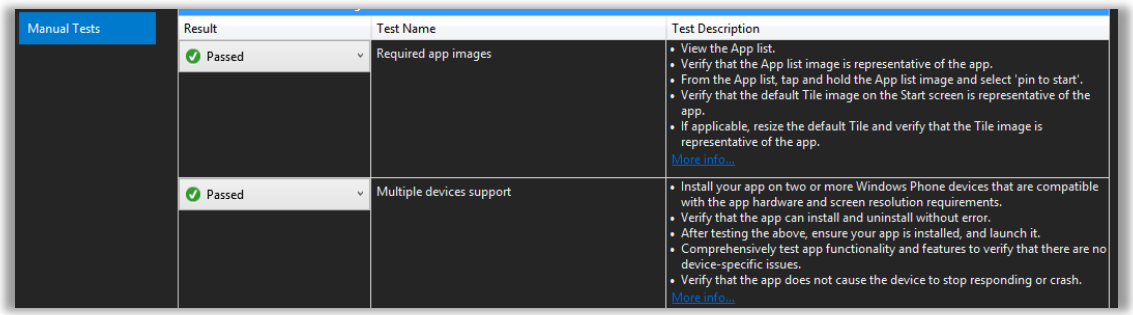


Figura 44 Testes manuais da ferramenta Store Test Kit

Os testes manuais são constituídos por 61 testes genéricos, de forma a abranger todo o tipo de aplicações, necessitando assim do programador decidir quais os testes adequados para a sua aplicação.

Para analisar questões de desempenho, poder-se-á usar ainda a ferramenta Windows Phone Application Analysis.

A aplicação foi publicada duas vezes, na qual a primeira vez uma versão beta, apenas disponível aos beta *testers*, e a segunda vez publicada para toda a comunidade Windows Phone. Ambas as publicações obtiveram a certificação da Microsoft sem problemas.

4.4.6.4. Code analysis

O Visual Studio fornece uma ferramenta denominada Code Analysis, que tem o intuito de analisar o código desenvolvido e indicar violações de regras de código.

O uso desta ferramenta permitiu detetar quatro avisos de violações de código, correspondentes a duas categorias, nomeadamente avisos sobre a libertação de objetos e da declaração de *event handlers* corretamente.

Como apresentado na Figura 45, todos os avisos gerados pela ferramenta foram corrigidos com sucesso.

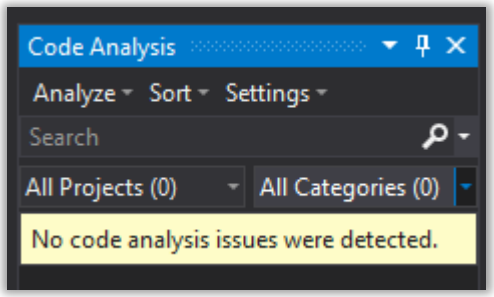


Figura 45 Resultado da análise efetuada da ferramenta Code Analysis

5. Propostas de melhoria à metodologia usada no Cloogy

Este capítulo tem como objetivo propor melhorias à metodologia usada pelas equipas de desenvolvimento *mobile* da ISA.

Como mencionado anteriormente, a equipa de desenvolvimento do Cloogy estabeleceu a sua própria metodologia de trabalho, fazendo um *tailoring* do processo C&D e adotando o *Scrum*.

No entanto, no desenrolar deste estágio, principalmente nas reuniões de retrospetiva no final de cada *sprint*, foram elencadas algumas dificuldades sentidas pela equipa de desenvolvimento do Cloogy, dificuldades essas causadas pela falta de ajustes no presente método de trabalho.

De forma a evitar estas dificuldades, é importante "limar algumas arestas" de modo a trazer mais valor ao método de trabalho usado atualmente, introduzindo novas políticas e procedendo à alteração de algumas existentes.

5.1. Proposta 1: Introdução do *Burndown Chart*

No desenvolvimento de *software*, por vezes o esforço das tarefas definidas no *Sprint Backlog* não é ajustado à equipa de desenvolvimento, sendo necessário ajustar o *Sprint Backlog*:

- Adiar a execução de tarefas, passando-as para a próxima *sprint*, caso a equipa tenha tarefas em demasia;
- Adicionar novas tarefas residentes no *Product Backlog* para o *Sprint Backlog* corrente, caso a equipa tenha poucas tarefas para executar, ou tarefas que surjam e que se revelem necessárias.

Contudo é necessário perceber o progresso do trabalho em desenvolvimento da *sprint*.

O *Scrum* tem como prática a criação de um artefacto essencial à equipa de desenvolvimento, nomeadamente o *Burndown Chart*, apresentado na Figura 46, onde o eixo horizontal mostra os dias da *sprint* e o eixo vertical mostra o tempo restante, em esforço, para o fim da *sprint*, ou

seja, no fim de cada dia de trabalho, este gráfico mostra a relação entre o trabalho realizado e o trabalho planeado.

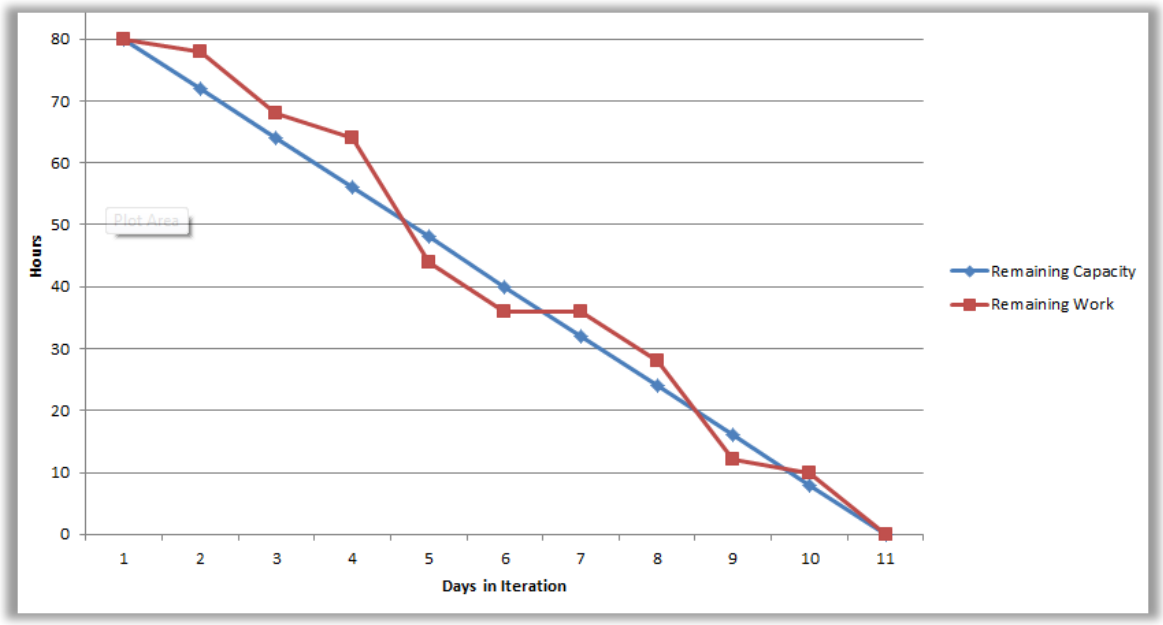


Figura 46 Exemplo de um Burndown Chart

Tipicamente este gráfico mostra duas linhas. A primeira linha é uma diagonal, servindo como guia para mostrar o desempenho diário da equipa. Esta diagonal é composta pelo ponto inicial (x_1,y_1) e pelo ponto final (x_2,y_2), onde:

- x_1 é o primeiro dia da *sprint*;
- y_1 é o tempo em esforço planeado para a execução de todas as tarefas da *sprint*;
- x_2 é o último dia da *sprint*;
- y_2 é traduzido como zero horas do esforço planeado, isto é, trabalho concluído.

A segunda linha, corresponde ao trabalho total que falta efetuar no final de cada dia da *sprint*. Ou seja, este gráfico permite visualizar três cenários:

- A segunda linha encontra-se acima da primeira, o que significa que a equipa está atrasada em relação ao planeado;
- A segunda linha encontra-se abaixo da primeira, o que significa que a equipa está adiantada em relação ao planeado;
- Dois segmentos de reta coincidentes, o que significa que a equipa está de acordo com o planeado.

Este gráfico permite assim visualizar de forma fácil e rápida o progresso do trabalho da equipa na *sprint*, facilitando assim o ajuste do *Sprint Backlog*, migrando tarefas para a próxima *sprint*, caso a equipa não consiga cumprir o seu compromisso, ou alocar novas tarefas caso a equipa esteja adiantada em relação ao planeado.

Em suma, o *Burndown Chart* facilita a visualização de discrepâncias entre o planeado e o desenvolvido, permitindo assim controlar o trabalho desenvolvido, estimar melhor na próxima *sprint* e melhorar a estratégia usada pela equipa.

Contudo, mesmo revelando-se uma mais-valia, a equipa de desenvolvimento do Cloogy não tem implementado esta prática. Este será um ponto a acrescentar de modo a trazer mais valor à metodologia usada pelo Cloogy.

Uma vez que a equipa já utiliza o JIRA, a introdução deste artefacto na equipa é bastante simples, pois esta ferramenta cria automaticamente o *Burndown Chart* com base no início/fim de cada *sprint* e esforço planeado para cada tarefa.

5.2. Proposta 2: Introdução da técnica *Planning Poker*

Um dos grandes desafios das equipas de desenvolvimento de *software* é estimar o esforço necessário para a realização de tarefas. Estas equipas tendem em subestimar o tempo que vão precisar para executar as suas tarefas, mesmo quando reúnem um histórico suficiente para poder realizar uma boa estimativa. Isto deve-se a dois fatores:

- A chefia querer o trabalho concluído num curto espaço de tempo;
- Os elementos das equipas têm receio de fazer estimativas pessimistas às suas próprias tarefas, pois pode dar a ideia de serem preguiçosos ou ineficientes.

Devido a isto, as estimativas das tarefas a executar na *sprint* deverão ser alimentadas com a técnica denominada de *Planning Poker*, um método eficaz usado em metodologias ágeis que permite com que as equipas consigam estimar, sem que os seus elementos sejam influenciados pelas respostas dos colegas.

O *Planning Poker* permite à equipa chegar a um consenso das estimativas efetuadas para cada tarefa ou *User Story*.

Todos os elementos da equipa participam nesta atividade, onde recebem um baralho de cartas próprio para este método, constituído pela sequência de Fibonacci[35]. Cada carta contém um número, *story point*, que é a unidade de estimativa desta técnica. Cada *story point* corresponde ao esforço necessário para executar determinada tarefa, ou seja, uma tarefa com 3 pontos significa que irá demorar o triplo do tempo de uma tarefa com 1 ponto. O tempo de cada *story point* é decidido pela própria equipa.

Este baralho contempla uma carta diferente, tipicamente um "Rei" ou um "?", o que significa que a tarefa/*User Story* a estimar é demasiado grande ou complexa, necessitando de ser dividida em várias tarefas/*User Story*.

O *Product Owner* deverá exercer a função de moderador de forma a:

- Apresentar as tarefas a serem estimadas;
- Avisar a equipa do início da votação;
- Controlar o tempo de eventuais discussões.

Depois do moderador apresentar a respetiva tarefa/*User Story*, todos os elementos da equipa escolhem uma carta e deverão virar as cartas para cima, assim que este avisar.

Todos os elementos que apresentarem resultados discrepantes dos resultados dos restantes elementos da equipa deverão apresentar uma justificação da sua votação. Sempre que haja discussão, esta não deverá ultrapassar o tempo estabelecido para estas situações, de modo a manter o controlo da reunião.

O esforço atribuído à respetiva tarefa/*User Story* será o consenso chegado pela equipa com base nas cartas apresentadas.

Esta proposta irá fazer com que as estimativas sejam mais eficientes.

Sempre que a equipa não consiga acabar de estimar todas as tarefas no próprio dia, a equipa deverá continuar a reunião no dia seguinte, antes de executar outras tarefas. Este método evita o excesso/ausência de tarefas perante os elementos, o que irá melhorar a coordenação entre a equipa no início de cada *sprint.*

[35] Sequência de Fibonacci – sequência de números começada em 0 ou 1, seguida do número 1. O número seguinte é igual à soma dos dois números anteriores. $F(0) = 0,1,1,2,3,5,8,13,21,34$ …

5.3. Proposta 3: Trabalhar sob *branches* do repositório

No fim de cada dia, todas as tarefas que estão terminadas devem ser *commited* no repositório do projeto, de modo a mantê-lo atualizado e para que não haja perda de informação em casos mais extravagantes.

No entanto, por vezes é feito *commit* de tarefas incompletas, o que poderá comprometer as tarefas de outros elementos da equipa. Para que isto não aconteça, é uma mais-valia que todos os elementos da equipa trabalhem sob *branches* da *trunk* do projeto.

Desta forma, sempre que os elementos da equipa necessitarem de fazer *commits* de tarefas incompletas fazem-no para a sua *branch*, salvaguardando assim o seu trabalho, sem afetar as tarefas dos restantes elementos da equipa. Sempre que as tarefas estejam completas, dever-se-á fazer *merge* para a *trunk* do projeto.

Esta técnica faz com que se mantenha sempre a *trunk* do projeto limpa e funcional.

5.4. Proposta 4: Melhoria na criação/atualização de tarefas

Por vezes os requisitos das tarefas não são explícitos o suficiente, surgindo assim a necessidade de recorrer ao *Product Owner* (PO) ou ao *Product Manager* (PM) para esclarecimento dessas dúvidas. Por vezes estes esclarecimentos não ficam anotados, o que mais tarde, aquando da necessidade da sua consulta, provoca conflitos entre *Stakeholders*.

O resultado destes pedidos de auxílio, sempre que pertinentes ao produto, devem ser traduzidos em novas tarefas, ou em atualizações de tarefas já existentes na ferramenta JIRA.

Mesmo que sejam pequenos detalhes de uma tarefa que parecem ter pouca importância, qualquer que seja a decisão tomada, deverá estar sempre retratada no JIRA. Qualquer rascunho que seja feito, um diagrama ou mesmo um *mockup* informal em papel, deverá ser passado para o JIRA.

Isto permite que as decisões não sejam perdidas, e venham a poder ser consultadas sempre que necessário.

5.5. Proposta 5: Criação de um documento com informação dos ambientes de desenvolvimento

A equipa de desenvolvimento do Cloogy trabalha com vários ambientes, de modo a poder desenvolver, testar e oferecer o seu produto ao cliente.

Sempre que se executa uma demo no final de cada *sprint*, esta é preparada no dia anterior por um elemento da equipa, sendo necessário ter conhecimento dos vários servidores existentes, base de dados e versões de produtos desenvolvidos. Porém, esta informação não se encontra disponível a todos os elementos, o que por vezes leva a uma maior dificuldade na preparação da demo.

De modo a atenuar esta dificuldade, é uma mais-valia a criação de um documento no repositório da equipa, com toda a informação essencial sobre os ambientes de desenvolvimento utilizados.

5.6. Proposta 6: Criação de uma agenda e convocatória para a demo

É imprescindível que todos os *stakeholders* estejam presentes nas demostrações do *software* no fim das *sprints*, de modo a validar as funcionalidades implementadas e a sua qualidade validada.

Como há projetos que são constituídos por vários componentes, nem todos os *stakeholders* estão envolvidos nos mesmos componentes, não sendo necessária a sua presença.

Com isto, deverá implementar-se outro artefacto, uma agenda da demo e convocatória dos respetivos *stakeholders*. Esta prática irá fazer com que os *stakeholders* apenas estejam presentes nas demostrações dos componentes em que estão envolvidos.

5.7. Proposta 7: Introdução da prática *code freeze*

Devido ao desenvolvimento do projeto ser feito em paralelo com a sua preparação para a demo, por vezes é entregue ao orador da demo a última versão apenas no término da *sprint*.

Isto leva com que a demo seja feita com versões de componentes diferentes das versões com que foi preparada.

Para que isto não aconteça, é imprescindível fazer "*code freeze"* ao projeto, ou seja, a partir de um ponto temporal previamente estabelecido, não se efetuam alterações ao projeto, de modo a que todo o trabalho efetuado até esse ponto temporal seja preparado para a demo.

A fim de implementar esta prática e que os restantes elementos da equipa possam continuar a trabalhar, é uma mais-valia abrir uma *tag* do projeto no SVN, x horas de esforço antes do fim da *sprint*, de modo a ser preparada para a demo.

5.8. Proposta 8: Prática a implementar para melhorar a preparação das demos

Como mencionado anteriormente, atualmente a demo é preparada por um elemento da equipa no dia anterior.

Porém, mesmo com a ajuda do documento que identifique todos os ambientes de desenvolvimento e versões dos projetos, como o projeto é constituído por vários componentes diferentes, por vezes os elementos não reúnem conhecimento suficiente sobre o componente em causa, o que por vezes atrasa a preparação da demo, provocando o insucesso da mesma.

Com isto, a preparação da demo deverá ser efetuada, não por um elemento, mas sim por *n* elementos, onde *n* corresponde ao número de componentes a demonstrar.

Este método irá fazer com que a preparação da demo seja mais rápida e eficiente, diminuindo a probabilidade de insucesso da demo.

5.9. Proposta 9: Seguir de *code conventions*

Ao longo da vida do *software* por vezes é necessário fazer manutenção ao seu código. A falta de consistência e uniformização do código dificulta esta tarefa.

É necessário facilitar a manutenção ou alteração do código das aplicações, sendo que, neste momento, a equipa que desenvolve o Cloogy, não tem implementado qualquer linha orientadora que permita seguir esta estratégia.

Estas linhas orientadoras tornarão o código mais consistente e limpo o que facilita a leitura e interpretação desse mesmo código ao leitor/revisor.

Sendo assim surgiu a oportunidade de trazer mais valor à equipa, desenvolvendo um documento com as linhas orientadoras de codificação, podendo ser consultadas no Anexo 1 *Code Conventions*.

6. *Framework* de desenvolvimento para Windows Phone 8

Baseado no estudo do SGI da ISA, nos procedimentos seguidos pela equipa de desenvolvimento do Cloogy e pela experiência adquirida dentro da equipa, o presente capítulo expõe uma *framework* de desenvolvimento de aplicações para a plataforma Windows Phone 8 na ISA, onde é proposto uma metodologia de trabalho, linhas orientadoras de desenvolvimento, ferramentas a serem usadas e *code conventions* a serem usados nos futuros projetos na plataforma Windows Phone.

6.1. Metodologia de trabalho a adotar

Uma vez analisado o método de trabalho da equipa de desenvolvimento do Cloogy nos capítulos anteriores, é descrito no presente capítulo uma metodologia de trabalho, de modo a atenuar dificuldades encontradas pela equipa perante o fluxo do método de trabalho usado atualmente.

A metodologia de trabalho sugerida tem como base o *Scrum*, o método de trabalho usado pela equipa de desenvolvimento do Cloogy, porém contendo algumas diferenças.

A Figura 47 representa as três fases do *Scrum*. Seguidamente serão explicadas as regras, os artefactos a serem criados e as ferramentas a usar.

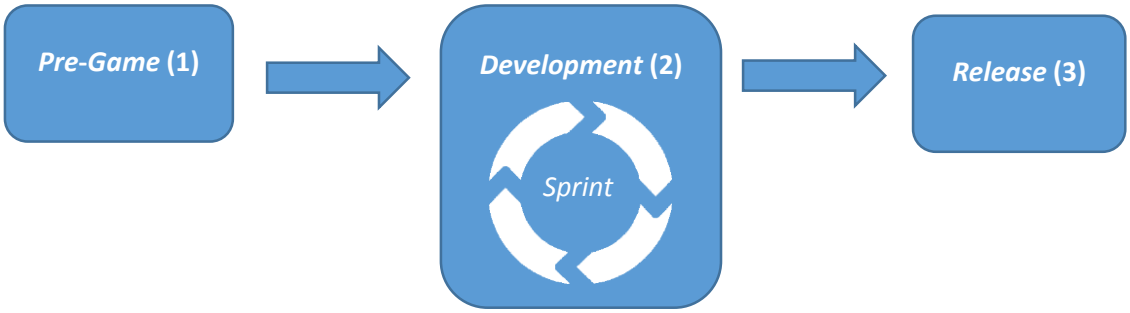


Figura 47 Ciclo de vida do Scum

6.1.1. *Pre-Game* (1)

A Figura 48 apresenta a primeira fase da Metodologia a adotar, a *Pre-Game*.

Figura 48 Artefactos e ferramentas da fase Pre-Game

Como estabelecido no C&D, os impressos IP.33 Ficha de Projeto e IP.170 *Tailoring* e Gestão de Configurações devem ser preenchidos. De forma a poder preencher o IP.33 Ficha de Projeto, nesta primeira fase deverá ser estabelecido:

- A visão do projeto;
- Métodos de trabalho;
- Papéis de cada um dos elementos da equipa;
- O repositório onde serão guardados todos os documentos relacionados com o novo projeto.

Nesta primeira fase, o *Product Manager* (PM) deverá:

- Inserir no JIRA todas as *User Stories* contadas por ele, assim como todos os requisitos provenientes do cliente, dos utilizadores finais, entre outros *stakeholders*;
- Assinalar a respetiva *User Story* ao *Scrum Master*.

Nesta fase é estabelecida uma *baseline* com o intuito de estabelecer a *Definition of Done*.

De forma a facilitar a implementação da interface dos produtos e para que não sejam implementadas vistas que mais tarde possam ser rejeitadas na fase seguinte, deverão ser construídos *mockups* dos primeiros ecrãs das aplicações a desenvolver.

Essa construção deve ser efetuada através do Balsamiq Mockups, respeitando as linhas orientadoras concebidas pela Microsoft. A partir dos *mockups* estabelecidos, é feito o trabalho de *design* de acordo com as necessidades dos *stakeholders*.

Dever-se-á também utilizar a ferramenta EA para elaborar a arquitetura alto nível do produto a desenvolver.

No capítulo 6.2.1 podem ser consultadas com maior detalhe, as ferramentas a usar no desenvolvimento de aplicações para Windows Phone 8.

6.1.2. *Development* (2)

A Figura 49 apresenta a segunda fase do *Scrum*, denominada por *Development*.

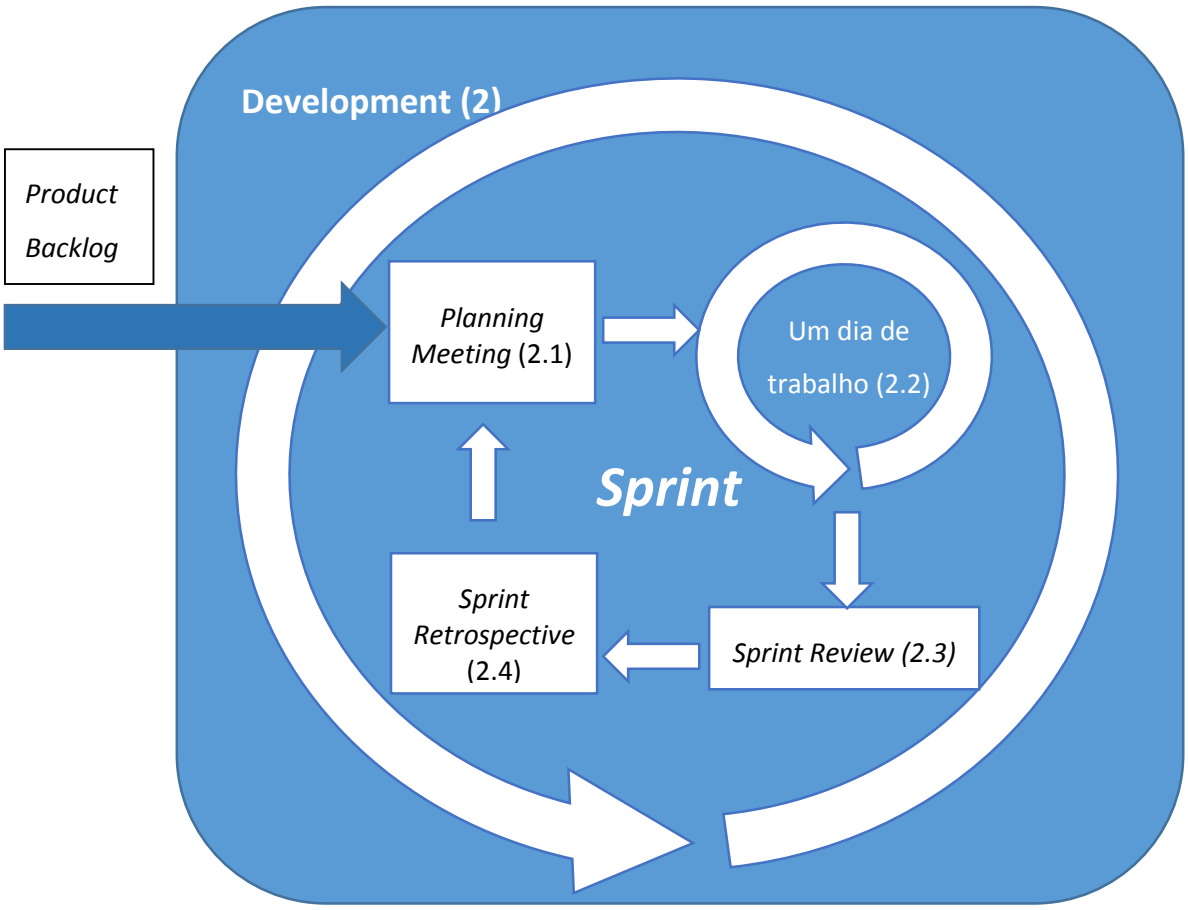


Figura 49 Ciclo de vida da fase Development

Esta fase é elaborada por várias *sprints* com duração de duas semanas, em que cada uma delas é constituída por quatro fases bem distintas:

- Uma *Planning Meeting* (2.1);
- Vários dias de trabalho (2.2);
- Uma *Sprint Review* (2.3) para demonstração do trabalho desenvolvido;
- Uma *Sprint Retrospective* (2.4).

A primeira *sprint* deverá ter uma duração maior, de modo a estabelecer a *baseline* da arquitetura.

As ferramentas necessárias nesta fase do desenvolvimento do projeto encontram-se detalhadas no capítulo 6.2.1.

6.1.2.1. *Planning Meeting* (2.1)

A primeira reunião de toda a equipa da segunda fase do *Scrum* denominada por *Planning Meeting*, deverá ser aproximadamente de quatro horas onde deverá ser para planear as tarefas oriundas das *User Stories* do *Product Backlog* previamente priorizadas pelo *Product Manager, Product Owner, Scrum Master* e *business analyst* na reunião *Backlog Grooming*.

De modo a que o planeamento das tarefas seja mais eficiente, deverá ser alimentado pela técnica denominada por *Planning Poker*.

Como apresentado na Figura 50, todas as tarefas devem ser planeadas de forma a que o esforço para a sua realização esteja dentro do estabelecido pela equipa.

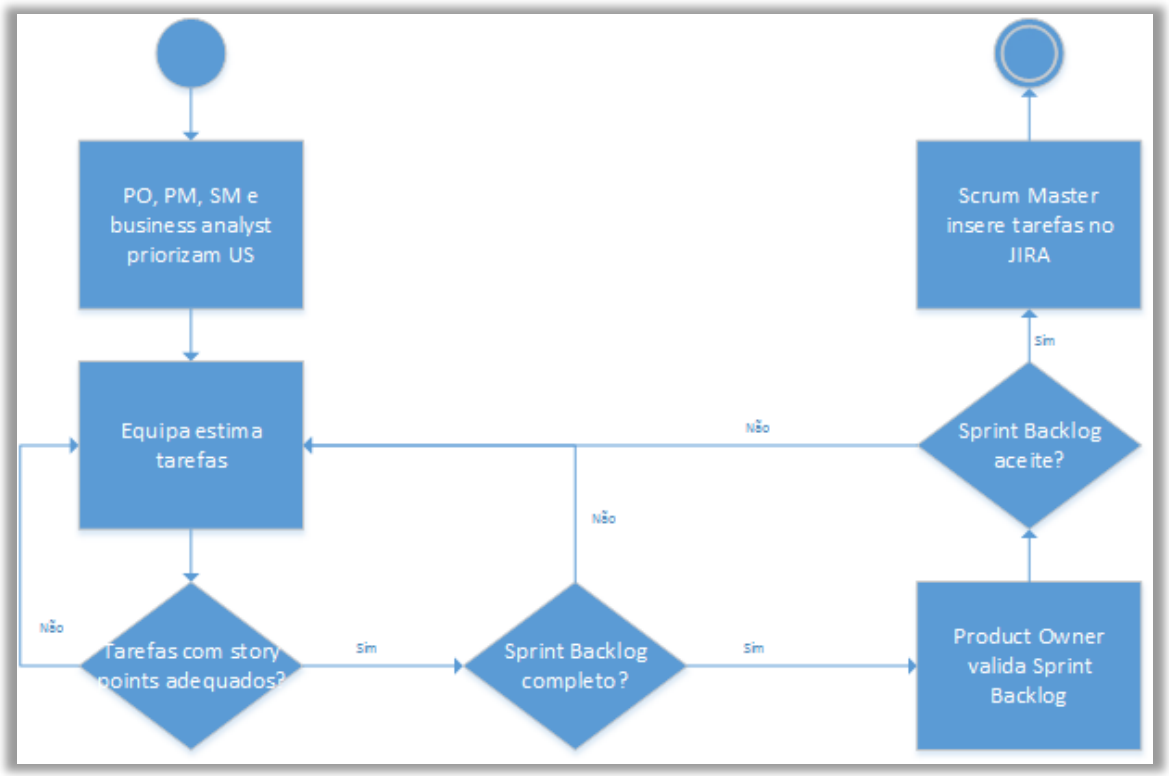


Figura 50 Fluxo para definição do Sprint Backlog

Todas as tarefas que não satisfaçam este requisito deverão ser replaneadas.

Assim que o *Sprint Backlog* estiver completo, deve ser enviado ao *Product Owner*, para que este o valide:

- Se o *Product Owner* não aprovar as tarefas do *Sprint Backlog*, deverá reportar as causas à equipa, para que esta consiga atualizar o *Sprint Backlog*;
- Se o *Product Owner* aprovar o *Sprint Backlog*, o *Scrum Master* deverá inserir todas as tarefas existentes dessa lista no JIRA, incluindo as descrições e esforço planeado.

No fim desta fase, deverá ser criado o *Burndown Chart* no JIRA para controlo e monitorização da *sprint*.

Caso a equipa não consiga completar o *Sprint Backlog* no próprio dia, é imprescindível que o faça no dia seguinte, de modo a evitar o atraso da *sprint*.

Assim que esta reunião terminar, o *Scrum Master* deverá enviar um email de compromisso a todos os stakeholders do projeto, sendo posteriormente guardado no repositório do projeto, de forma a evidenciar o compromisso da equipa como definido no C&D.

6.1.2.2. Um dia de trabalho (2.2)

A Figura 51 retrata o fluxo de um dia de trabalho, indicando as seguintes ações:

- Uma *Daily Meeting* ao início do dia;
- Possíveis pedidos de esclarecimento de requisitos devido a uma má especificação ou entrada de *Blockers*;
- Atualização do JIRA e do SVN.

Figura 51 Fluxo de um dia de trabalho

Ao início do dia a equipa deverá reunir-se para fazer a *Daily Meeting*, liderada pelo *Scrum Master*. Caso os recursos não estejam juntos presencialmente, devem ser usadas ferramentas de *conference call* como meio de comunicação.

Com base no estado de cada recurso e do *Burndown Chart* da equipa, poder-se-á transferir tarefas entre recursos, de modo a não atrasar a *sprint*. Sempre que possível dever-se-á atualizar o *Sprint Backlog*.

Ao longo do dia, poderão surgir novos *issues* com prioridade elevada, denominados por *Blockers*. Devem ser resolvidos logo após terem chegado à equipa. Por norma se não forem resolvidos terão um impacto enorme no sistema ou no Cliente, podendo assim interromper a *sprint*.

Para resolver um *Blocker* é necessário que este seja validado pelo *Product Owner*, para que a equipa o resolva. Caso não tenha informação suficiente para ser resolvido, deverá ser devolvido a quem reportou o *issue* no JIRA, para que possa detalhar melhor.

Sempre que seja necessário pedir auxílio ao *Product Manager* para ajustar requisitos, ou pedir ao *Product Owner* para tomar uma decisão de teor mais técnico, as opções tomadas deverão ser introduzidas no JIRA, para que não haja perda de informação.

No final de cada dia, todos os elementos da equipa deverão atualizar o JIRA, de modo a que o *Burndown Chart* esteja atualizado para a *Daily Meeting* do dia seguinte.

Dever-se-á igualmente atualizar todos os artefactos existentes no repositório da equipa, quer estejam nas *branches* ou na *trunk*, de modo a atualizar o projeto e todos os documentos que o suportam.

6.1.2.3. *Sprint Review* (2.3)

Como estabelecido na *baseline* de verificação, deve-se garantir a viabilidade da realização da reunião de *Sprint Review* no último dia de cada *sprint*.

Para se proceder à demonstração (demo) de cada *sprint* dever-se-á:

- Fazer "*code freeze*" aos componentes a demonstrar;
- Preparar a demo com *n* elementos correspondente a *n* componentes do projeto;
- Elaborar uma reunião entre os oradores;
- Enviar uma agenda e convocatória aos *stakeholders*;
- Ter atualizado o documento que identifique todos os ambientes de desenvolvimento e versões dos projetos.

Os componentes a demonstrar deverão ser os mesmos dos que foram preparados até ao "*code freeze*".

O final da preparação da demo, deverá ser marcado com uma reunião entre os oradores da mesma, para que possa ser enviada via email, a agenda e convocatória dos respetivos *stakeholders*, para que estes possam estar presentes e assistir apenas aos projetos onde estão envolvidos.

Durante a execução da demo, deverão ser apresentados os tópicos existentes na agenda previamente enviada, ou seja, todos os requisitos pertencentes ao *Sprint Backlog*.

O *Scrum Master* deverá ficar responsável por anotar todas as decisões nesta reunião, nomeadamente requisitos aprovados e alterações necessárias a desenvolver, no caso dos requisitos rejeitados.

6.1.2.4. *Sprint Retrospective* (2.4)

O término de cada *sprint* deverá ser marcado com uma reunião denominada por *Sprint Retrospective*. Nesta reunião são discutidos assuntos de carácter mais pessoal, sendo dispensados todos os *stakeholders* que não sejam constituintes da equipa. Esta reunião deverá ser subdividida em três fases, para que cada elemento da equipa consiga dar a sua opinião:

- Na primeira fase são mencionados os aspetos positivos;
- Na segunda fase são explicadas as dificuldades sentidas e os aspetos negativos da *sprint*;
- Na terceira fase são mencionadas possíveis melhorias para atenuar as dificuldades sentidas e promover uma melhoria contínua do método utilizado.

Em cada uma dessas fases, o *Scrum Master* deverá anotar as opiniões mencionadas pelos elementos da equipa, de modo a que sejam disponibilizadas à equipa QAPE como definido no C&D.

A melhoria contínua mencionada anteriormente poderá ser suportada por uma técnica denominada de *Five Whys*[36] de modo a descobrir a origem dos problemas.

6.1.3. *Release* (3)

O início da terceira fase do *Scrum*, nomeadamente a *Release*, terá início assim que o projeto reúna condições para ser *deployed*, como definido na primeira *baseline*.

Dever-se-á garantir que o *package* do projeto está completo e que os testes efetuados tenham sido todos aceites, quer sejam testes manuais ou testes automáticos realizados pelas ferramentas fornecidas pela Microsoft, para que o projeto possa ser publicado na Windows Phone Store com sucesso.

[36] *Five Whys* – técnica iterativa de pergunta-resposta usada para explorar as relações de causa-efeito de um problema.

Aquando da publicação da aplicação na Windows Phone Store, dever-se-á lançar uma versão prévia exclusiva para *beta testers*, de modo a que estes testem primeiro a aplicação desenvolvida em ambiente real. Assim que a aplicação se demonstrar estável, dever-se-á lançar uma versão para toda a comunidade Windows Phone.

Como definido no C&D, de forma a evidenciar as *releases* efetuadas, devem ser publicadas na *Intranet* da ISA.

6.2. Linhas Orientadoras de desenvolvimento para Windows Phone 8

No desenvolvimento da *framework* para desenvolvimento de novas áreas na plataforma Windows Phone 8 para a ISA, tornou-se imprescindível o estudo das linhas orientadoras de desenvolvimento relacionadas com *User Interface* (UI), modos de interação, modelos de estrutura e navegação desta plataforma concebidas pela própria Microsoft. Desta forma, foi feito um estudo sobre estas linhas orientadoras, que visam fazer uma primeira abordagem ao ambiente da plataforma em questão, cuidados a ter com a implementação do *design* das aplicações, com os modos de interação e com as resoluções disponíveis dos dispositivos que correm esta plataforma. O estudo efetuado no início do projeto pode ser consultado no Anexo 2 Windows Phone 8 *guidelines*.

6.2.1. Ferramentas a usar no desenvolvimento de aplicações para Windows Phone 8

Este capítulo tem o intuito de definir as ferramentas para o desenvolvimento de aplicações para a plataforma Windows Phone 8.

Todas as ferramentas necessárias para o desenvolvimento dos projetos para a plataforma Windows Phone 8, devem ser instaladas e asseguradas pelo Departamento de Sistemas de Informação (DSI). O DSI deverá assegurar o bom funcionamento das ferramentas definidas.

A Tabela 19 apresenta as ferramentas necessárias a cada elemento da equipa de desenvolvimento do projeto, onde a letra X significa a necessidade da sua utilização:

	Scrum Master	*Product Manager*	*Product Owner*	*Development Team*
JIRA	X	X	X	X
TortoiseSVN	X	X	X	X
Balsamiq Mockups	-	X	X	X
Visual Studio	-	-	X	X
Windows Phone SDK	-	-	X	X
Enterprise Architect	-	-	X	X
Blend	-	-	-	X

Tabela 19 Ferramentas usadas pelos diversos elementos da equipa

As ferramentas JIRA e SVN são utilizadas por todos os elementos da equipa, ao contrário das restantes ferramentas, que apenas são utilizadas por alguns elementos.

No Anexo 3 podem ser consultados tutoriais das ferramentas:

- JIRA;
- Balsamiq Mockups;
- Visual Studio e Windows Phone SDK;
- Blend.

Para consulta da ferramenta EA, o processo C&D disponibiliza um guia de utilizador desta ferramenta no manual interno denominado por MI.13 Guia de Utilizador do Enterprise Architect definido no C&D.

6.2.1.1. Visual Studio e Windows Phone SDK

Esta *framework* estabelece duas ferramentas como sendo essenciais no desenvolvimento de aplicações nesta plataforma, nomeadamente o IDE Visual Studio e o Windows Phone SDK, cujas ferramentas fornecem os mecanismos necessários para o desenvolvimento de aplicações para a plataforma Windows Phone.

No Visual Studio, dever-se-á criar um novo projeto Windows Phone usando a linguagem C# e XAML, usando de preferência os *templates* fornecidos pela Microsoft. Estes *templates* permitem aos programadores pouparem tempo na criação de ficheiros, e posteriormente adaptá-los à medida.

Para que haja uniformização entre código de aplicações desta plataforma, no seu desenvolvimento dever-se-á utilizar o MVVM como padrão arquitetural. Este padrão permite separar o desenvolvimento da vista da aplicação, do desenvolvimento da lógica de negócio, atenuando assim o *code-behind* substituído por classes que implementem a interface *INotifyPropertyChanged* de forma a fazer *databinding* com o código XAML.

Este IDE fornece duas ferramentas importantes no desenvolvimento de aplicações, nomeadamente as ferramentas:

- Code Analysis;
- Store Test Kit.

O Code Analysis tem como o intuito de analisar o código desenvolvido e indicar violações de regras de código. Os avisos gerados estão agrupados por áreas, como *design*, desempenho, segurança e localização. Cada aviso gerado significa uma violação de uma regra.

A segunda ferramenta, Store Test Kit, ajuda a identificar problemas existentes na aplicação antes da submissão da mesma na Windows Phone Store, fornecendo testes automáticos e um conjunto de testes manuais para serem executados pelo programador.

Como mostrado na Tabela 20, a plataforma Windows Phone 8 disponibiliza neste momento quatro resoluções diferentes. Durante o desenvolvimento é imprescindível averiguar a consistência da aplicação perante as várias resoluções, para isto é aconselhável testar as aplicações nos vários emuladores fornecidos pelo SDK.

Nome	Resolução	Aspeto
WVGA	480x800	15:9
WXGA	768x1280	15:9
720p	720x1280	16:9
1080p	1080x1920	16:9

Tabela 20 Diferentes resoluções da plataforma Windows Phone

Na solução do projeto, para além do produto a desenvolver, dever-se-á criar um projeto para execução de testes unitários. O Visual Studio oferece este tipo de soluções, facilitando assim o desenvolvimento dos testes unitários.

Para um maior detalhe da utilização deste IDE com o respetivo SDK, pode ser consultado um tutorial no capítulo 3 do Anexo 3.

6.2.1.2. Enterprise Architect

Com o intuito de ajudar a definir a arquitetura das aplicações, poderá ser usada o Enterprise Architect (EA), esta ferramenta contém um conjunto de funcionalidades muito importantes que permitem a criação dos diversos tipos de requisitos das aplicações e criar dependências entre módulos. Desta forma é possível agrupar as diversas arquiteturas dos vários projetos e apresentando-os como uma única solução.

De forma a perceber melhor o funcionamento desta ferramenta, poderá ser consultado o guia de utilizador desta ferramenta no manual interno MI.13 Guia de Utilizador Enterprise Architect.

6.2.1.3. Blend

O Blend é uma ferramenta que é usada para facilitar a criação de controlos personalizados, e animações dos mesmos. É possível abrir esta ferramenta diretamente a partir do Visual Studio e visualizar as propriedades de um controlo.

Como mostrado na Figura 52, esta ferramenta permite ainda visualizar o conteúdo de controlos de outras origens, possibilitando assim personalizá-los de acordo com as necessidades dos *stakeholders*.

Figura 52 Editar um controlo no Blend [18]

Por vezes é necessário construir controlos personalizados em XAML, definindo assim novas figuras, texturas, cores, tipos de letra ou mesmo animações de controlos. Tudo isto poderá ser desenvolvido no Visual Studio através do XAML, porém o Blend permite aos recursos da equipa uma maior facilidade nesse tipo de construções.

Para um maior detalhe da utilização desta ferramenta pode ser consultado um tutorial no capítulo 4 do Anexo 3.

6.2.1.4. Balsamiq Mockups

Para desenvolvimento dos *mockups* das aplicações, poderá ser usada a ferramenta Balsamiq Mockups. Esta ferramenta permite de uma forma rápida e simples criar *mockups*, devido ao *drag-and-drop* de controlos pré construídos pela ferramenta. Esta permite ainda aos seus utilizadores criar *templates* para que possam ser utilizados quando necessário.

No capítulo 2 do Anexo 3, pode ser consultado um tutorial da presente ferramenta.

6.2.2. *Code conventions*

Num projeto que envolva muitos programadores é necessário manter uma coerência na forma como estes programam, de forma a uniformizar o código. Ou seja, para ajudar na sua organização, é necessário estabelecer nomenclatura de artefactos e modos de inicialização.

Assim, no desenvolvimento de *software* é importante definir linhas orientadoras de codificação. Estas linhas permitem um código mais consistente e limpo o que facilita a leitura e interpretação desse mesmo código ao leitor/revisor.

As linhas orientadoras de codificação definidas no decorrer deste projeto podem ser consultadas no Anexo 1 *Code Conventions*.

7. Conclusões e trabalho futuro

A realização do presente estágio permitiu adquirir novos conhecimentos e aperfeiçoar a capacidade de planeamento e organização de trabalho.

A possibilidade de estar inserido na equipa de desenvolvimento do Cloogy mostrou-se uma mais-valia, pois surgiu a oportunidade de experienciar a realidade do mundo do desenvolvimento de *software.* Com as respetivas cerimónias do *Scrum*, foi possível receber ajuda dos elementos da equipa e de analisar o seu método de trabalho.

7.1. Análise da metodologia usada

A ISA possui um Sistema de Gestão Integrado (SGI) de Qualidade e Inovação tendo como finalidade servir de linha orientadora para os seus colaboradores desenvolverem as suas ações e garantir a melhoria contínua dos processos definidos.

Embora a ISA possua o processo Conceção e Desenvolvimento (C&D) para desenvolvimento de genérico de *software*, a equipa do Cloogy estabeleceu os seus próprios métodos de trabalho, fazendo um *tailoring* bastante amplo do processo C&D, optando por trabalhar sob uma metodologia ágil, nomeadamente o *Scrum*.

O *Scrum* não é um processo ou uma técnica para desenvolvimento de *software*, é uma *framework* de suporte ao seu desenvolvimento e manutenção, que permite aplicar vários processos e técnicas. Apesar de ser possível implementar apenas partes do *Scrum*, o resultado não é *Scrum*, mas sim "*Scrum-But"*, ou seja, uso do *Scrum* mas com ajustes de forma a enquadrar as necessidades de cada entidade, por exemplo "*We use Scrum, but Daily Scrum meetings are too much overhead so we only have them once a week*." [19].

Aquando do uso de *Scrum*, dever-se-á sempre seguir todos os seus valores e práticas, para que seja possível tirar o máximo do seu proveito. É imprescindível implementar de forma correta a estratégia 3334 ou seja, três valores, três papéis, três artefactos e quatro cerimónias. Para além destes, a equipa do Cloogy adicionou um papel e uma cerimónia: *Product Manager* e *Backlog Grooming*.

Os três valores que sustentam a implementação do *Scrum* são transparência, inspeção e adaptação. Os três papéis estabelecidos são indispensáveis na realização de um projeto, sejam eles a *Development Team*, *Product Owner* e *Scrum Master*. Todos os elementos das equipas

Scrum deverão comparecer nas quatro cerimónias denominados por *Planning Meeting*, *Daily Meeting, Sprint Review* e *Sprint Retrospective*. Ao longo das *sprints*, deverão ser criados e monitorizados os artefactos *Product Backlog*, *Sprint Backlog* e *Burndown Chart*.

Uma vez o *Scrum* bem implementado, este método de trabalho vai de acordo com o propósito do SGI da ISA, as equipas conseguem promover uma melhoria contínua do método utilizado a partir das *Daily Meetings* e das *Sprint Retrospectives*.

No decorrer do presente estágio, essencialmente durante as cerimónias *Daily Meeting* e *Sprint Retrospective*, foram elencadas dificuldades sentidas pela equipa de desenvolvimento do Cloogy. Com isto, surgiu a oportunidade de apresentar propostas de melhoria à metodologia usada, a fim de trazer mais valor ao método usado. As nove propostas apresentadas passaram por introduzir novos artefactos, técnicas e melhorias ao planeamento e acompanhamento de tarefas.

Estas propostas foram induzidas na *framework* de desenvolvimento de aplicações para a plataforma Windows Phone 8, criada no presente estágio, para orientar as equipas de desenvolvimento *mobile* da ISA, estabelecendo assim linhas orientadoras de desenvolvimento para esta plataforma, ferramentas a serem usadas e *code conventions*, que até ao momento eram inexistentes na empresa.

A *framework* estabelecida irá fazer com que os futuros *developers* possam desenvolver os seus projetos *mobile* e que as presentes equipas possam melhorar o seu método de trabalho.

7.2. Análise do desenvolvimento da aplicação Cloogy para Windows Phone 8

A inexperiência no desenvolvimento de aplicações Windows Phone no início do presente estágio revelou-se um enorme desafio, sendo necessária a familiarização da plataforma, tecnologias e métodos usados.

A aplicação desenvolvida permitiu adquirir novas competências na plataforma Windows Phone, abordando assim:

- Cuidados a ter com o design das aplicações;
- Modelos de navegação da plataforma;
- Resoluções da plataforma;

- Decisões arquiteturais;
- Ferramentas a serem usadas;
- Criação de *User Controls*;
- Utilização de *Resources*;
- Criação de animações gráficas;
- Criação de testes unitários;
- Publicação na Windows Phone Store.

O presente estágio permitiu desenvolver um vasto conjunto de tarefas que fazem parte do desenvolvimento de *software*. Possibilitou o levantamento e especificação de requisitos, definição da arquitetura do *software* e implementação e testes da aplicação. Na última fase do projeto, surgiu a oportunidade da publicação da aplicação Cloogy na Windows Phone Store.

7.3. Trabalho futuro

No que diz respeito à aplicação desenvolvida, de forma trazer-lhe mais valor, irão ser acrescentadas novas funcionalidades à aplicação Cloogy para Windows Phone 8:

- Adicionar, substituir e remover tomadas inteligentes;
- Visualizar o histórico de atuações das tomadas inteligentes;
- Visualizar os consumos dos tanques de gás;
- Gerir o perfil de utilizador;
- Simulação de tarifas;
- Acrescentar um *wizard* para ativação de um Kit Cloogy;
- Permitir iniciar sessão e registar uma nova conta a partir das redes sociais LinkedIn e Google+.

O Cloogy possui uma visão de *Smart Living*, pelo que a evolução da aplicação passará por novas funcionalidades que alimentem este conceito, como integração de novos sensores, novas plataformas e novas APIs.

Durante a execução do presente estágio não foi possível comparar a aplicação desenvolvida com as aplicações desenvolvidas para sistemas operativos Android e iOS, sendo esta tarefa uma mais-valia para as aplicações, de modo a melhorar o seu desempenho.

8. Referências

[1] “Cloogy,” ISA, [Online]. Disponível em: http://cloogy.pt/pt/apresenta%C3%A7%C3%A3o/. [Acedido em Junho 2014].

[2] “ZigBee Specification FAQ,” 2014. [Online]. Disponível em: http://www.zigbee.org/Specifications/ZigBee/FAQ.aspx. [Acedido em Junho 2014].

[3] B. Mitchell, “Bluetooth,” About.com, 2014. [Online]. Disponível em: http://compnetworking.about.com/cs/bluetooth/g/bldef_bluetooth.htm. [Acedido em Junho 2014].

[4] “Agile Buddha,” [Online]. Disponível em: http://www.agilebuddha.com/trainings-workshops/scrum-training-workshop/. [Acedido em Junho 2014].

[5] E. v. d. Valk, “Practicing patterns,” 14 Agosto 2009. [Online]. Disponível em: http://blogs.msdn.com/b/erwinvandervalk/archive/2009/08/14/the-difference-between-model-view-viewmodel-and-other-separated-presentation-patterns.aspx. [Acedido em Junho 2014].

[6] M. Fowler, “Presentation Model,” Julho 2004. [Online]. Disponível em: http://martinfowler.com/eaaDev/PresentationModel.html. [Acedido em Junho 2014].

[7] J. Smith, “WPF Apps With The Model-View-ViewModel Design Pattern,” *msdn magazine,* Fevereiro 2009.

[8] “Using the Model-View-ViewModel (MVVM) pattern in Hilo (Windows Store apps using C++ and XAML),” MSDN, [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/windows/apps/jj160324.aspx. [Acedido em Junho 2014].

[9] “Visão geral do XAML,” MSDN, [Online]. Disponível em: http://msdn.microsoft.com/pt-br/library/windows/apps/hh700354.aspx. [Acedido em Junho 2014].

[10] M. Corporation, “C# Language Specification 5.0,” [Online]. Disponível em: http://www.microsoft.com/en-us/download/confirmation.aspx?id=7029. [Acedido em Junho 2014].

[11] “Introducing JSON,” [Online]. Disponível em: http://www.json.org/. [Acedido em Junho 2014].

[12] “LINQ (Language-Integrated Query),” MSDN, 2013. [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/bb397926.aspx. [Acedido em Junho 2014].

[13] “Introduction to LINQ Queries (C#),” MSDN, [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/bb397906.aspx. [Acedido em Junho 2014].

[14] J. Belfiore , “The Windows Phone Toolkit,” CodePlex, 15 Agosto 2013. [Online]. Disponível em: http://phone.codeplex.com/. [Acedido em Junho 2014].

[15] “Central app hub with home page menu (Panorama or Pivot control) for Windows Phone,” MSDN, 1 Abril 2014. [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/windowsphone/design/hh202892(v=vs.105).aspx. [Acedido em Junho 2014].

[16] “Pivot control design guidelines for Windows Phone,” MSDN, 1 Abril 2014. [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/windowsphone/design/hh202919(v=vs.105).aspx. [Acedido em Junho 2014].

[17] “List with details drilldown for Windows Phone,” MSDN, 1 Abril 2014. [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/windowsphone/design/hh202886(v=vs.105).aspx. [Acedido em Junho 2014].

[18] A. Chatterjee, “Building Windows Phone 7 User Interface with Expression Blend,” MSDN, 11 Maio 2011. [Online]. Disponível em: http://blogs.msdn.com/b/amit_chatterjee/archive/2011/05/11/building-windows-phone-7-user-interface-with-expression-blend.aspx. [Acedido em Junho 2014].

[19] R. Bunning, “Kicking ScrumBut,” Outubro 2009. [Online]. Disponível em: http://www.scrumalliance.org/resource_download/1122. [Acedido em Junho 2014].

[20] E. Johnson, “Improve Your Scrum Meeting, 15 Minutes to Heaven,” 24 Fevereiro 2014. [Online]. Disponível em: http://www.intland.com/blog/agile/improve-your-scrum-meeting-15-minutes-to-heaven/. [Acedido em Junho 2014].

[21] “Entries for 'scrum',” 2014. [Online]. Disponível em: https://www.scrum.org/About/All-Articles/articletype/tagview/tag/scrum. [Acedido em Junho 2014].

[22] D. Bernstein, “Seven Strategies for Improving Your Scrum Process,” To Be Agile, 24 Julho 2012. [Online]. Disponível em: http://tobeagile.com/2012/07/24/seven-strategies-for-improving-your-scrum-process/. [Acedido em Junho 2014].

[23] A. Singleton, “3 Problems with Scrum,” 13 Março 2012. [Online]. Disponível em: http://blog.assembla.com/assemblablog/tabid/12618/bid/79430/3-Problems-with-Scrum.aspx. [Acedido em Junho 2014].

[24] D. Mezick, “Scrum,” New Technology Solutions, Inc., [Online]. Disponível em: http://www.newtechusa.com/Resources/Scrums3333.pdf. [Acedido em Junho 2014].

[25] B. Rinko-Gay, “You May Be a Scrum-But,” Scrum Alliance, 1 Fevereiro 2013. [Online]. Disponível em: http://www.scrumalliance.org/community/articles/2013/february/you-may-be-a-scrum-but. [Acedido em Junho 2014].

[26] G. Villanueva, “Scrum and Continuous Process Improvement,” Scrum Alliance, 14 Janeiro 2014. [Online]. Disponível em: http://www.scrumalliance.org/community/articles/2014/january/scrum-and-continous-process-improvement-(1). [Acedido em Junho 2014].

[27] E. Lima, “Burndown chart - Mede o progresso da sprint e dá indicativos do processo de trabalho da equipe,” 9 Janeiro 2012. [Online]. Disponível em: http://blog.myscrumhalf.com/2012/01/burndown-chart-medindo-o-progresso-de-sua-

sprint-e-trazendo-indicativos-do-processo-de-trabalho-da-equipe/. [Acedido em Junho 2014].

[28] M. Stal, “O Planning Poker evita falácias nas estimativas de esforço,” 29 Agosto 2012. [Online]. Disponível em: http://www.infoq.com/br/news/2012/08/planning-poker-e-falacias. [Acedido em Junho 2014].

[29] M. Rouse, “Planning Poker,” Julho 2011. [Online]. Disponível em: http://searchsoftwarequality.techtarget.com/definition/planning-poker. [Acedido em Junho 2014].

[30] K. Rubin, “Scrum Framework,” 24 Setembro 2013. [Online]. Disponível em: http://agileatlas.org/articles/item/scrum-framework. [Acedido em Junho 2014].

[31] “Os pilares da teoria do SCRUM,” CRM ZEN, 15 Ooutubro 2013. [Online]. Disponível em: http://blog.crmzen.com.br/post/64126024323/os-pilares-da-teoria-do-scrum. [Acedido em Junho 2014].

[32] M. James, “Scrum Reference Card,” 2014. [Online]. Disponível em: http://scrumreferencecard.com/scrum-reference-card/. [Acedido em Junho 2014].

[33] K. Schwaber e J. Sutherland, “The Scrum Guide™,” Julho 2013. [Online]. Disponível em: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf. [Acedido em Junho 2014].

[34] “UI for Windows Phone,” Telerik, [Online]. Disponível em: http://www.telerik.com/products/windows-phone.aspx. [Acedido em Junho 2014].

[35] M. Rouse, “Fibonacci sequence,” Julho 2007. [Online]. Disponível em: http://whatis.techtarget.com/definition/Fibonacci-sequence. [Acedido em Junho 2014].

[36] J. Spool, “The Difference Between Usability and User Experience,” Março 2007. [Online]. Disponível em: The Difference Between Usability and User Experience. [Acedido em Junho 2014].

[37] “Resources in .Resx File Format,” MSDN, [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/ekyft91f(v=vs.90).aspx. [Acedido em Junho 2014].

[38] “Walkthrough: Creating and Running Unit Tests for Managed Code,” MSDN, [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/ms182532.aspx. [Acedido em Junho 2014].

[39] L. Fischer, “A Beginner's Guide to HTTP and REST,” 9 Janeiro 2013. [Online]. Disponível em: http://code.tutsplus.com/tutorials/a-beginners-introduction-to-http-and-rest--net-16340. [Acedido em junho 2014].

[40] E. Alecrim, “O que é ERP (Enterprise Resource Planning)?,” 9 Junho 2010. [Online]. Disponível em: http://www.infowester.com/erp.php. [Acedido em Junho 2014].

[41] s. methodology, “Scrum Backlog Grooming,” [Online]. Disponível em: http://scrummethodology.com/scrum-backlog-grooming/. [Acedido em Junho 2014].

[42] “Mobile OS (Operating System) Percent Market Share — Europe,” areppim AG, 26 Março 2014. [Online]. Disponível em: http://stats.areppim.com/stats/stats_mobiosxtime_eu.htm. [Acedido em Junho 2014].

[43] “3.9 Automatic memory management,” MSDN, [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/aa691138(v=vs.71).aspx. [Acedido em Junho 2014].

[44] “How to navigate using the back stack for Windows Phone 8,” MSDN, [Online]. Disponível em: http://msdn.microsoft.com/en-US/library/windowsphone/develop/hh394012(v=vs.105).aspx. [Acedido em Junho 2014].

[45] “Planning Poker®,” Crisp, [Online]. Disponível em: http://www.crisp.se/bocker-och-produkter/planning-poker. [Acedido em Junho 2014].

[46] “Design library for Windows Phone,” MSDN, 1 Abril 2014. [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/windowsphone/design/hh202915(v=vs.105).aspx. [Acedido em Junho 2014].

Anexo 1 *Code Conventions*

Antes de definir as linhas orientadoras para o desenvolvimento de aplicações para Windows Phone 8, é necessário perceber algumas definições:

- ***Camel Case*** – Uma "palavra" com a primeira letra minúscula. Caso seja constituída por vários nomes, a primeira letra de cada nome é maiúscula.
    - Exemplo: nomeUtilizador.
- ***Pascal case*** – Uma "palavra" com a primeira letra maiúscula. Caso seja constituída por vários nomes, a primeira letra de cada nome é maiúscula.
    - Exemplo: NomeUtilizador.
- ***Hungarian notation*** – uso de um prefixo para especificar o tipo de uma variável.
    - Exemplo: strNomeUtilizador para uma variável do tipo *string*.

Em seguida serão mostradas as linhas orientadoras estabelecidas para o desenvolvimento de *software*, nomeadamente para Windows Phone 8. Serão mostradas linhas orientadoras gerais e em seguida as linhas específicas para cada zona do código.

1. Linhas gerais

Independentemente da zona do código onde o programador esteja a programar, é importante estabelecer algumas diretrizes gerais:

- Como apresentado na Tabela 21 as variáveis e propriedades deverão descrever uma entidade não o seu tipo ou tamanho, devido a isto, não é recomendado seguir a notação Húngara para especificação o tipo das variáveis.

Correto	Incorreto
int carNumber	int iCarNumber
string driverName	string strDriverName

Tabela 21 Exemplos de notações Húngaras para variáveis

- Evitar o uso de abreviaturas, é recomendado o uso de nomes completos como apresentado na Tabela 22.

Correto	Incorreto
UserGroup userGroup;	UserGroup usrGrp;
Assignment userAssignment;	Assignment urAssignment;

Tabela 22 Exemplo de abreviaturas no nome de variáveis

  - Contudo existem algumas exceções, como por exemplo:
    - ClientId clientId;
    - XmlDocument xmlDocument;
    - FtpHelper ftpHelper;

- Como apresentado na Tabela 23, não é aconselhável omitir os access *modifiers:*

Correto	Incorreto
private void CountItem(){}	void CountItem(){}

Tabela 23 Exemplo do uso de access modifiers

- Evitar avaliar condições booleanas com *true* ou *false* como apresentado na Tabela 24:

Correto	Evitar
If(isExist){}	If(isExist == true){}

Tabela 24 Exemplo do uso de condições boleanas

- Tornar as condições de fácil leitura:
  - Evitar:

```
if (((value > highScore) && (value != highScore)) && (value < maxScore)){}
```

  - Solução:

```
isHighScore = (value > highScore);

isTiedHigh = (value == highScore);

isValid = (value < maxValue);
```

2. Classes

Os nomes das classes devem ser constituídas por um nome ou por um conjunto de nomes sem abreviaturas e deverão seguir a terminologia Pascal.

Todas as classes deverão ter um comentário XML com o propósito da mesma.

```
// <summary>
///This class performs an important function
/// </summary>
public class Car{}
```

3. Interfaces

Os nomes das interfaces deverão ser constituídos por nomes ou adjetivos e seguir a terminologia Pascal. Para além disto, deverão incluir o prefixo "I".

- *public interface INotifyPropertyChanged{}*

4. Variáveis

Os nomes das variáveis deverão ser constituídos por mais do que um carater e seguir a terminologia Camel:

- *int carNumber = 0;*

Com a exceção de variáveis dentro de loops, no caso de um FOR, a variável pode ser constituída por um único carater:

- *for(int i=0;i< carNumber; i++){}*

Como apresentado na Tabela 25, não é recomendado o uso de *Underscores*, com exceção do uso do mesmo como prefixo de uma variável privada:

Correto	**Exceção**
public DateTime clientAppointment;	private DateTime _registrationDate;

Tabela 25 Exemplo da nomenclatura de uma variável privada

É recomendado usar nomes de tipos predefinidos, em vez de nomes dos tipos de sistema, como Int32, String, Boolean, como apresentado na Tabela 26:

Correto	**Evitar**
string firstName;	String firstName;
int lastIndex;	Int lastIndex;
bool isExist;	Bool isExist;

Tabela 26 Exemplo de tipos de variáveis a usar

Como apresentado na Tabela 27, todas as variáveis deverão ser inicializadas com valores por defeito, caso seja necessário inicializar com outro valor, deverá ser comentada a razão.

Tipo	**Inicialização**
int	0
string	“”
bool	false
decimal	0
float	0
double	0

Tabela 27 Inicialização de variáveis

Exemplo:

```
bool isExist = false;
int carNumber = 0;
```

5. Public Properties

Os nomes das propriedades públicas deverão ser constituídos por um nome que represente o que a propriedade retorna e seguir a terminologia Pascal.

Exemplo: *public string Name{ get; set; }*

6. Métodos e argumentos

Os nomes dos métodos não deverão conter abreviaturas e deverão ser constituídos por um verbo e seguir a terminologia Pascal. Os argumentos de cada método deverão seguir a terminologia Camel e deverão ser nomes completos sem abreviaturas.

Antes de cada método deverá existir um comentário em XML com uma pequena descrição do método, descrição dos argumentos e do retorno da função.

```
// <summary>
///This method clear the statistics
/// </summary>
///<param name=”value”>Value to be convert</param>
```

7. Componentes

Como apresentado na Tabela 28, sempre que sejam adicionados nomes aos componentes usados em XAML, deverão seguir a terminologia Camel, contendo um prefixo de acordo com o componente usado:

Componente	Prefixo
TextBox	tbox
TextBlock	tblock
Button	btn
Grid	grid
Arc	arc
Rectangle	rect
StackPanel	stPanel
ItemsControl	itemCntrl
Panorama	pan
PanoramaItem	panItem
LongListSelector	llistSelect

Tabela 28 Prefixo do nome dos componentes em XAML

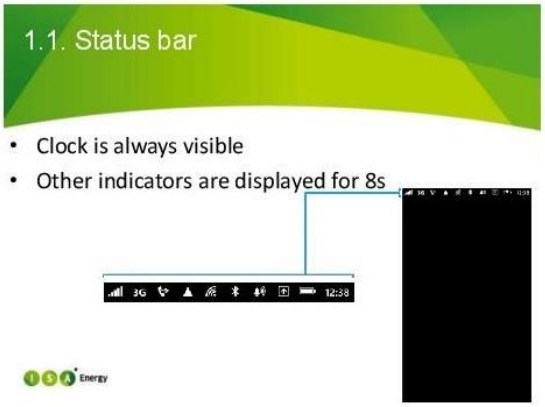
8. Comentários

Um artefacto importante no código de uma aplicação é nomeadamente os comentários. Sempre que seja preciso fazer manutenção do código, estes ajudam bastante, pois permitem obter um resumo rápido do conteúdo em questão.

Sempre que seja adicionada uma nova classe ou função, deverá existir um comentário em XML no início das mesmas, como mencionado anteriormente. Para além deste caso, sempre que hajam operações que não sejam triviais ou que sejam específicas para um certo caso, também deverão existir comentários, de modo a obter respostas às perguntas, “O que é isto?” e “Porquê que foi feito desta maneira?”.

Como mencionado anteriormente, variáveis inicializadas com valores que não sejam valores por defeito, também deverão ser comentadas, desta forma, um indivíduo que esteja a ver o código, obtém logo resposta à pergunta “Porquê este valor?”.

Anexo 2 Windows Phone 8 *guidelines*



2. Hardware Buttons

- 1 - Power/sleep
- 2 - Volume up/down
- 3 - Camera
- 4 - Back
- 5 - Start
- 6 - Search

3. Implementing WP app design

- To create a great app is necessary:
  - What will your app do?
  - Who is your app for?
  - How does your app fit in?
  - Where and when will your app be used?
  - What **kind of content** will be displayed?

3. Implementing WP app design

- Before create final mockups:
  - Structure and navigation models
  - Motions and interactions
  - Controls design guidelines

4. Structure and navigation models

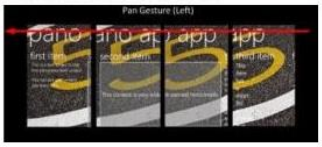
- 3 common navigation models:
  - Panorama control
  - Pivot control
  - List with details drilldown
- Mixing models

4.1. Panorama Control

- Consists:
  - Background image
  - Panorama title
  - Panorama section titles
  - Panorama sections

4.1. Panorama Control

- It is Portrait orientation (only)
- It **is not** recommended:
  - Have more that 5 sections
  - Use Pivot control inside Panorama

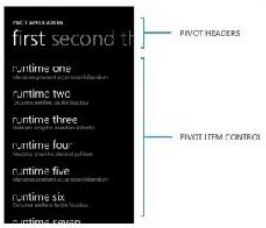


4.1. Panorama Control

- Multiple content types
- Don't use controls that can pan or scroll
  - Solution: navigate to single subpage
- Back button doesn't return to the previous section

4.2. Pivot Control

- Consists:
  - Title
  - Pivot Headers
  - Pivot item control



4.2. Pivot Control

- Support Portrait and landscape
- It **is not** recommended:
  - Have more 4 pivot pages
  - Have discrepancy information

4.2. Pivot Control

- Don't use controls that can pan or scroll
  - Solution: navigate to single subpage
- Back button doesn't return to the previous item

4.3. List with details drilldown

- Simple navigation model
- Each entry in the list redirects to the details page

4.4. Mixing models

- Pivot model + list with details drilldown

5. Pages and frames

- System supports **back stack**
  - User navigates from page 1 (p1) to page 2 (p2) to p1 to p2 to page 3 (p3) to p1
  - creates a back stack of p1, p2, p1, p2, p3, p1
- Don't build a Back button!
  - Note: splash screen isn't a page, it's a screen

5.1. Back button

- Pressing the Back button
  - Previous pages
  - First screen
    - Close the app
  - Context menus and dialogs
    - Close the menu and dialog

6. Multi-Resolution

- WP 8 supports WVGA, WXGA, 720p and 1080p resolutions

Resolution	Resolution	Aspect ratio	Delta from Windows Phone OS 7.1	Scaled resolution
WVGA	480 × 800	15:9	None. This is the only supported resolution for Windows Phone OS 7.1.	480 × 800
WXGA	768 × 1280	15:9	1.6x scale	480 × 800
720p	720 × 1280	16:9	1.5x scale, 80 pixels taller (53 pixels, after scaling)	480 × 853
1080p	1080 × 1920	16:9	1.5x scale, 80 pixels taller (53 pixels, after scaling)	480 × 853

6. Multi-Resolution

- To make a page render correctly:
  - Don't hard code the height and width
  - Don't hard code the margins of controls
- Put the controls in grid
  - Use "*" and "Auto" value to height and width

6.1. Backgrounds and assets

- Assets increase app's size:
  - it is recommended include only WXGA
- WXGA has highest quality to be scaled

6.1. Splash screens

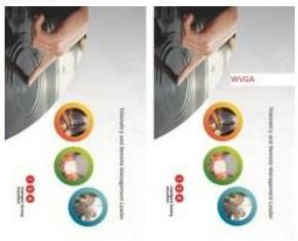
- To provide pixel-perfect:
  - SplashScreenImage.Screen-WVGA.jpg
  - SplashScreenImage.Screen-WXGA.jpg
  - SplashScreenImage.Screen-720p.jpg
- Else
  - Use single image 768x1280
    - The image is automatically scaled to the correct size.

6.1. Splash screens

- 720p resolution

6.1. Splash screens

- WVGA resolution



6.1. Splash screens

- WXGA resolution

6.2. Tiles and app icons

- The phone automatically scales the images to the correct size

6.3. Tiles

- Consists:
  - Count
  - Background Image
  - Title
- 3 kinds of tile templates:
  - Flip
  - Icon
  - Cycle

6.3. Tiles

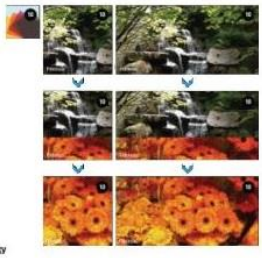
- Flip

6.3. Tiles

- Icon

6.3. Tiles

- Cycle



6.3. Tiles

- WP 8 supports 3 tile sizes:
  - Small
  - Medium
  - Wide

	Flip and Cycle
Small	159 × 159 pixels
Medium	336 × 336 pixels
Wide	691 × 336 pixels

6.3. Tiles

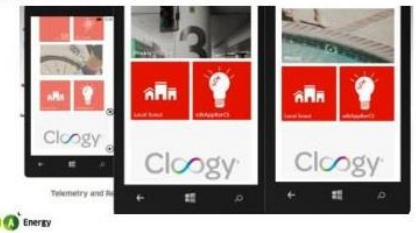
- Small

6.3. Tiles

- Medium

6.3. Tiles

- Wide



6.4. Application icon

- Used for all resolutions:
  - 99x99px

7. Demonstration

References

- Design library for Windows Phone
- First look at Windows Phone
- Tile design guidelines for WP
- Design resources for Windows Phone
- Multi-resolution apps for WP 8
- WP 8: Multiple Screen Resolutions
- How to navigate using the back stack for WP

Anexo 3 Tutorial de ferramentas para desenvolvimento de aplicações Windows Phone 8

Este anexo tem como objetivo apresentar todas as ferramentas necessárias ao desenvolvimento de aplicações na plataforma Windows Phone 8, quer sejam ferramentas de desenvolvimento ou monitorização.

1. JIRA

Este capítulo tem como objetivo apresentar um tutorial da ferramenta JIRA, onde conste algumas das funcionalidades desta ferramenta.

1.1. Criar um issue

Este capítulo tem como objetivo exemplificar a criação de um novo *Issue* na ferramenta JIRA.

1. Clicar no botão *Create Issue* como apresentado na Figura 53 ou simplesmente na tecla *C*

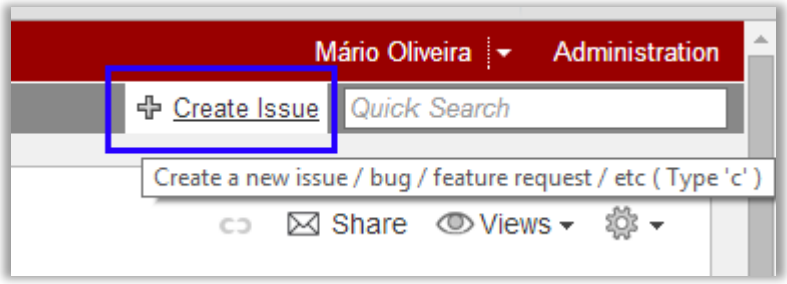


Figura 53 Criar issue

2. Preencher campos obrigatórios:
    a. Selecionar projeto
    b. Indicar o tipo de *issue*:
        i. Backlog Item,
        ii. Generic Bug,
        iii. Support Request ou

iv. *Story*

c. Preencher o sumário

d. Indicar o repórter do *issue*

e. Indicar a prioridade do *issue*:

i. *Blocker,*

ii. *Must have,*

iii. *Should have,*

iv. *Nice to have* ou

v. *Trivial*

3. Dever-se-á preencher também:

a. A data de vencimento

b. Versão afetada pelo *issue*

c. Indicar o *Assignee*

d. Uma descrição do pretendido e se possível anexar informação

e. Indicar a estimativa de esforço para realização do *issue*

1.2. Procurar e atualizar um *issue*

Neste capítulo são apresentadas formas de procurar um *issue* e como o editar.

1. Procurar um *issue* na lista de issues ou diretamente a partir de uma query, onde se pode identificar o projeto, *sprint*, estado do *issue* ou mesmo o seu id (Figura 54)

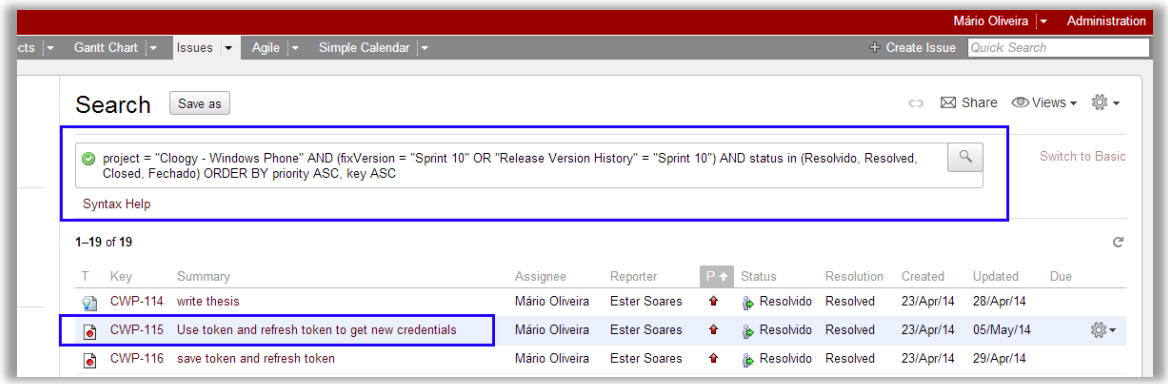


Figura 54 Procurar um issue

2. Selecionar o issue pretendido
3. Editar o *issue* como pretendido:
    a. Para registar horas de trabalho, clicar no botão para o respetivo efeito como se pode ver na Figura 55.

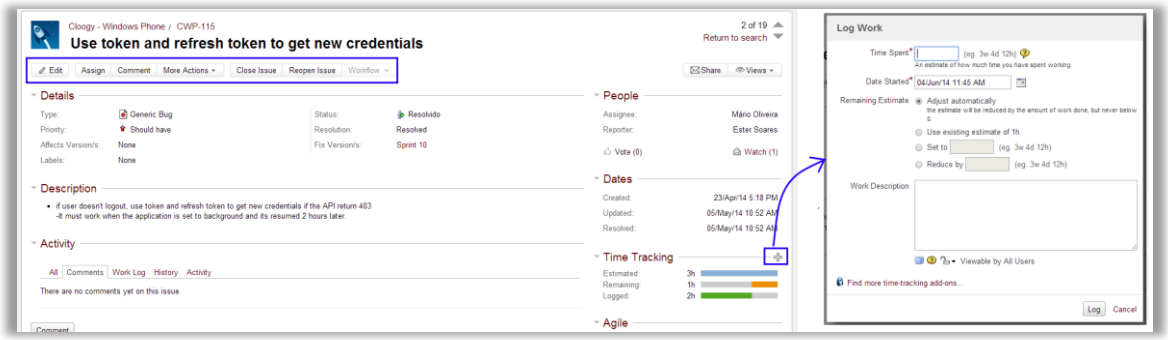


Figura 55 Editar um issue e registar horas de trabalho

1.3. Adicionar um Burndown Chart à dashboard

No desenvolvimento de *software* torna-se útil utilizar um *Burndown Chart* de modo a visualizar o progresso do trabalho da equipa na *sprint* de uma forma rápida e fácil.

1. Selecionar *Dashboards* no menu da ferramenta JIRA
2. Selecionar o Botão *Add Gadget* para escolher o *Burndown Chart* desejado

Estes passos podem ser visíveis na Figura 56.

Figura 56 Adicionar um Gadget à Dashboard

2. Balsamiq Mockups

Neste capítulo é apresentado um tutorial para uso da ferramenta Balsamiq Mockups, de modo a facilitar a criação de *mockups* para o desenvolvimento de aplicações. São mencionados os

passos necessários para criar e alterar um mockup e técnicas que ajudam o seu desenvolvimento.

2.1. Criar um novo *mockup*

Para criar um novo mockup é necessário clicar em *File* no menu da ferramenta Balsamiq Mockups. Como mostrado na Figura 57, esta ferramenta fornece ao utilizador duas formas de criar um novo mockup:

- Criar um novo mockup vazio, ou
- A partir de um mockup que esteja a ser visualizado.

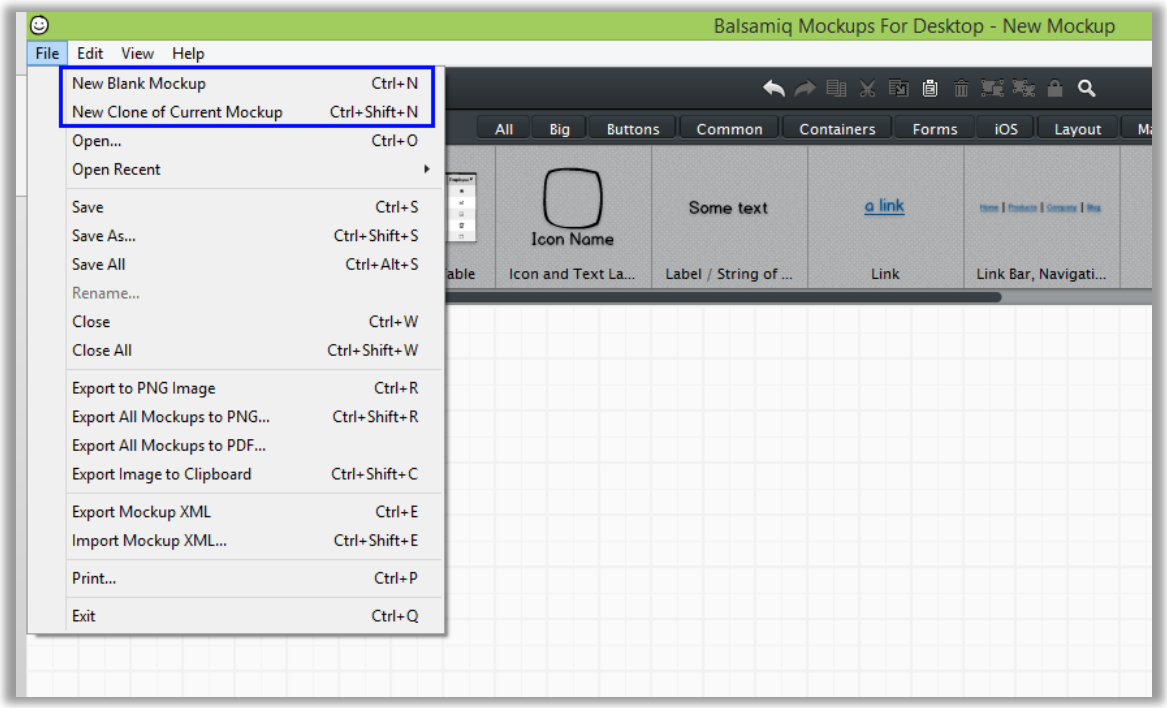


Figura 57 Criar um novo mockup

Uma vez que foram criados *mockups* para a aplicação Cloogy para a plataforma Windows Phone 8 ao longo deste estágio, no repositório do mesmo, poder-se-á aceder a esses ficheiros. É aconselhado assim que sejam criados *mockups* a partir desses, de forma a poupar tempo na criação de outros *templates*.

2.2. Adicionar controlos

Para adicionar controlos, esta ferramenta permite fazê-lo de duas formas:

- Pesquisa rápida (Figura 58) e
- *Drag and drop* (Figura 59).

Figura 58 Pesquisa rápida de um controlo

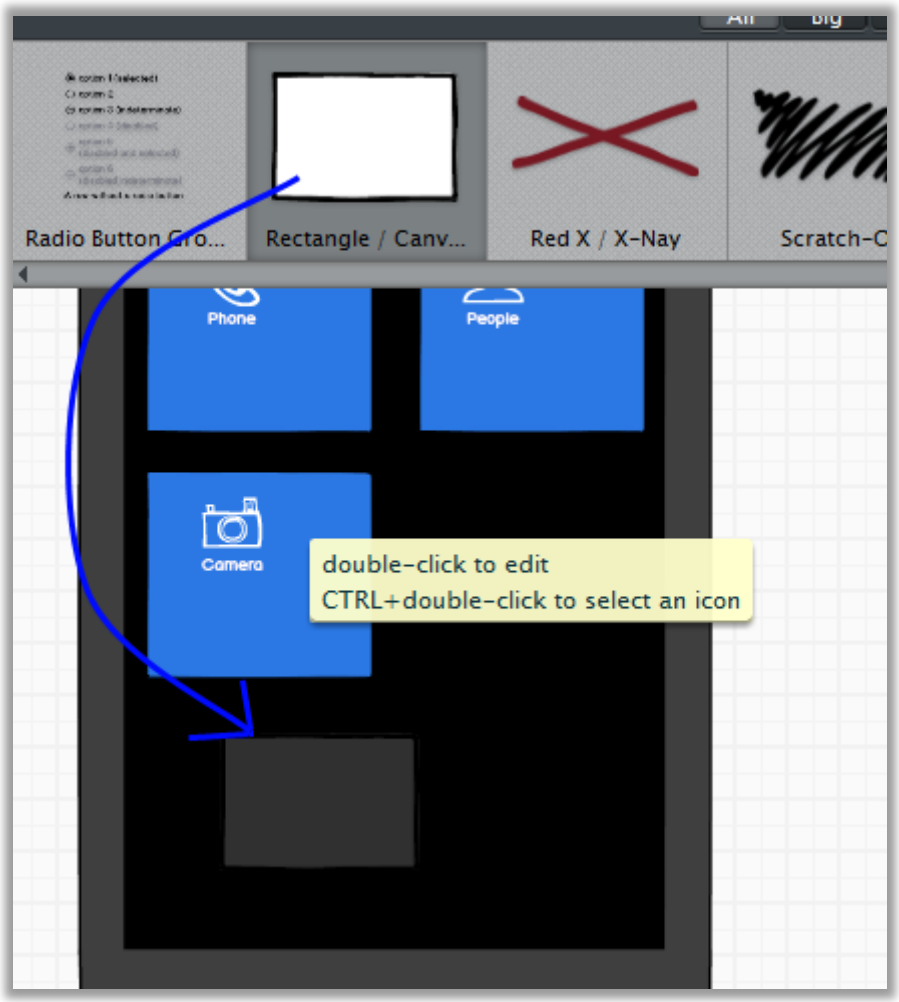


Figura 59 Drag and drop de elementos

A caixa de propriedades que se pode visualizar na Figura 60 e na Figura 61, é dependente do tipo de controlo que é escolhido. Esta caixa de propriedades permite ao utilizador personalizar o controlo à sua medida, ou seja, permite alterar o tamanho do próprio controlo e do texto, o seu posicionamento, opacidade, estilos do bordo, entre outras propriedades.

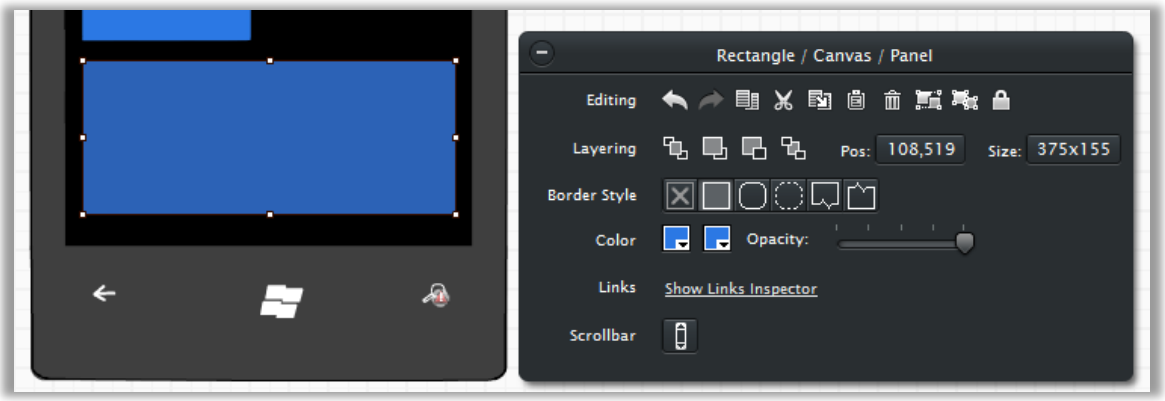


Figura 60 Caixa de propriedades de um retângulo

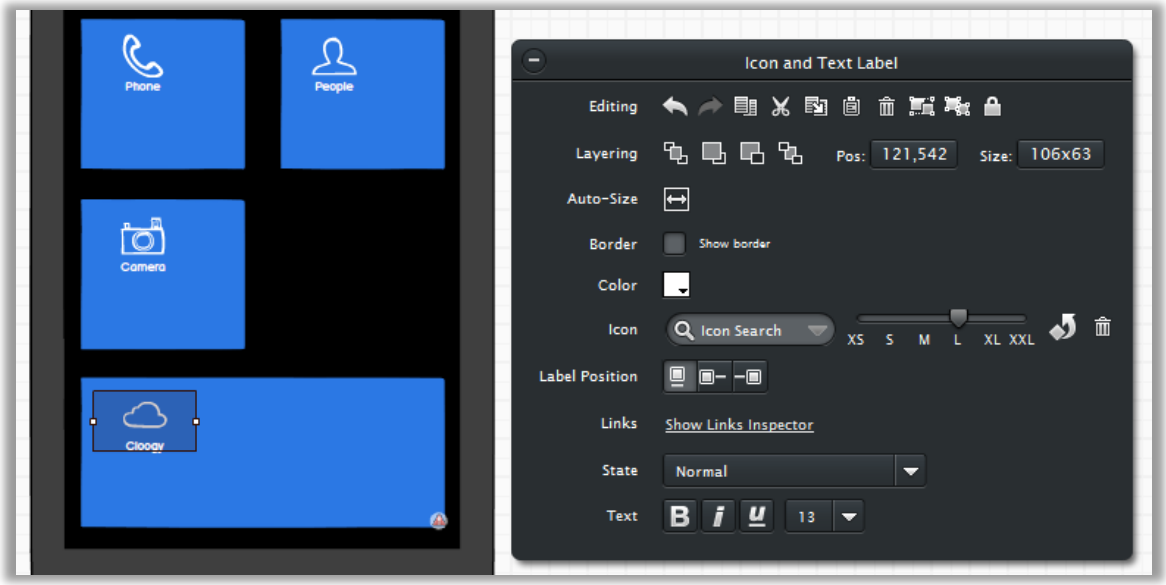


Figura 61 Caixa de propriedades de um Ícone

Independentemente do tipo de controlo que se escolha, a caixa de propriedades apresenta sempre um conjunto de opções, tais como:

- Desfazer/ Refazer,
- Duplicar,
- Cortar/ Copiar/ Colar/ Eliminar,
- Agrupar/ Desagrupar,
- Bloquear na posição corrente.

Para agrupar controlos é necessário:

1. Selecionar os elementos desejados alternadamente com o conjunto de teclas CRTL + Botão esquerdo do rato,
2. Clicar no ícone *Group* presente na caixa de propriedades.

Para bloquear um elemento numa posição, apenas é necessário selecionar o elemento desejado e clicar no ícone *Lock in Place* presente na caixa de propriedades.

O resultado destas duas operações pode ser visualizado na Figura 62.

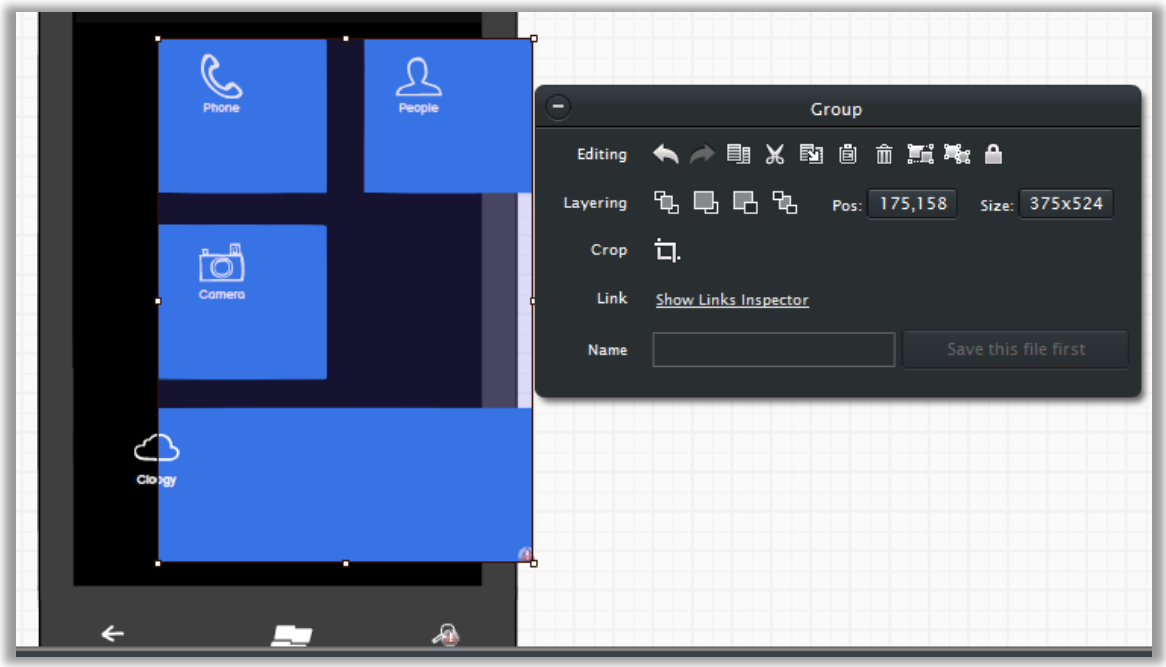


Figura 62 Exemplo de controlos agrupados e bloqueados numa posição

2.3. Modo de apresentação

A ferramenta Balsamiq Mockups permite criar um fluxo com todos os *mockups* criados, para isto é necessário:

1. Guardar todos os *mockups* no mesmo local em disco,
2. Selecionar um controlo,
3. Atribuir um *link* a partir da caixa de propriedades

O modo de apresentação permite visualizar todo o fluxo criado anteriormente, para isto, apenas é necessário clicar no botão para o respetivo efeito posicionado no canto superior direito, como se pode ver na Figura 63.

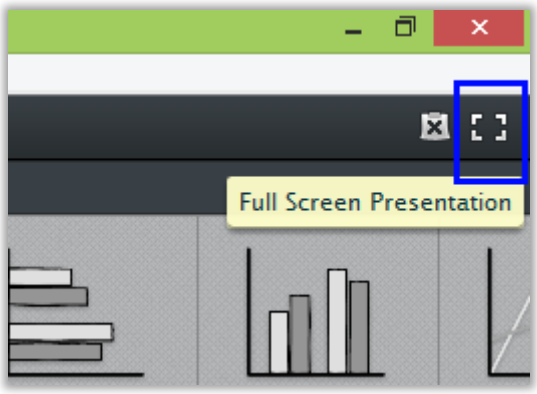


Figura 63 Como selecionar o modo de apresentação

A Figura 64 exemplifica o modo de apresentação. Nesta mesma imagem é possível visualizar um comentário e um *link* para outro mockup. Caso o utilizador não queira que seja visível a dica do respetivo *link* e o comentário, pode escondê-los a partir de uma caixa posicionada no canto inferior direito do ecrã. Esta caixa permite ainda sair do modo de apresentação e ir para o modo de edição.

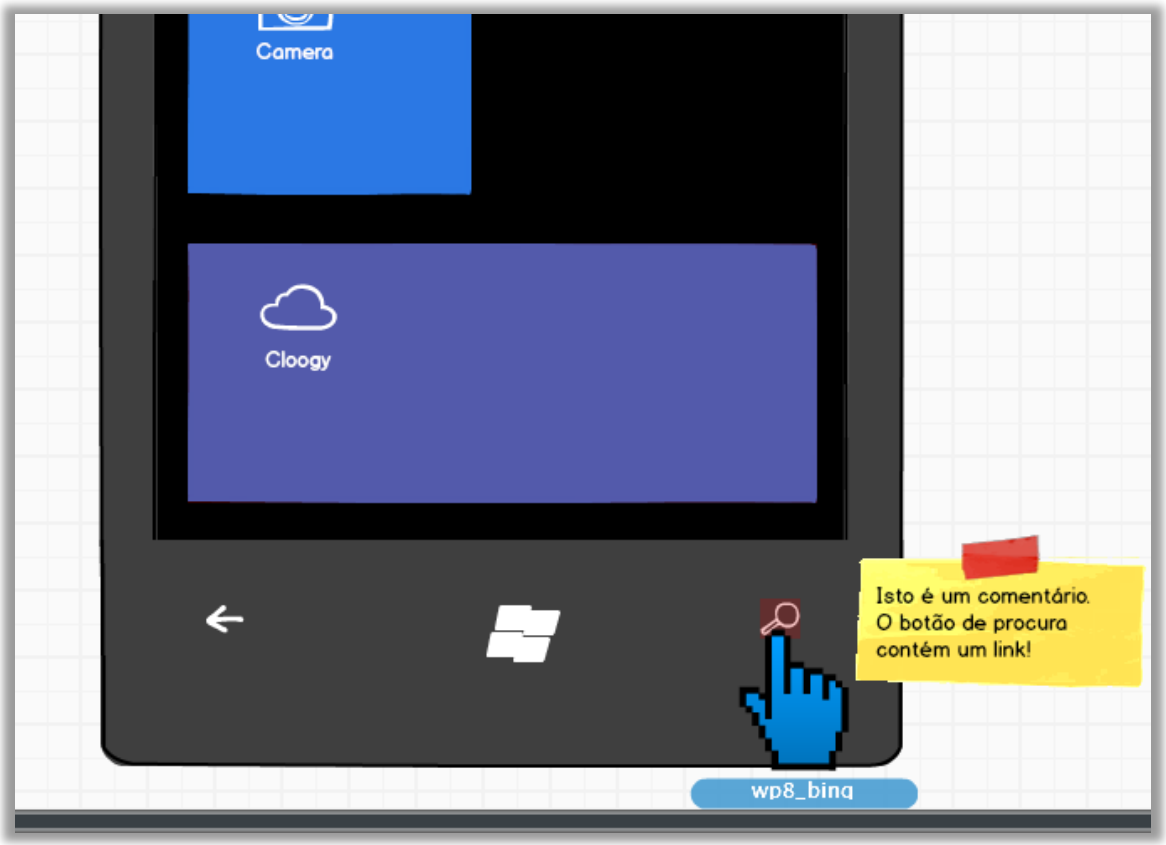


Figura 64 Modo de apresentação

2.4. Exportar para ficheiro no formato PDF

É possível exportar os *mockups* para vários formatos, inclusive para PDF. Para isso, é necessário:

1. Clicar em *File* residente na primeira entrada do menu da ferramenta,
2. Escolher *Export All Mockups to PDF.*

A Figura 65 exemplifica os passos anteriores descritos.

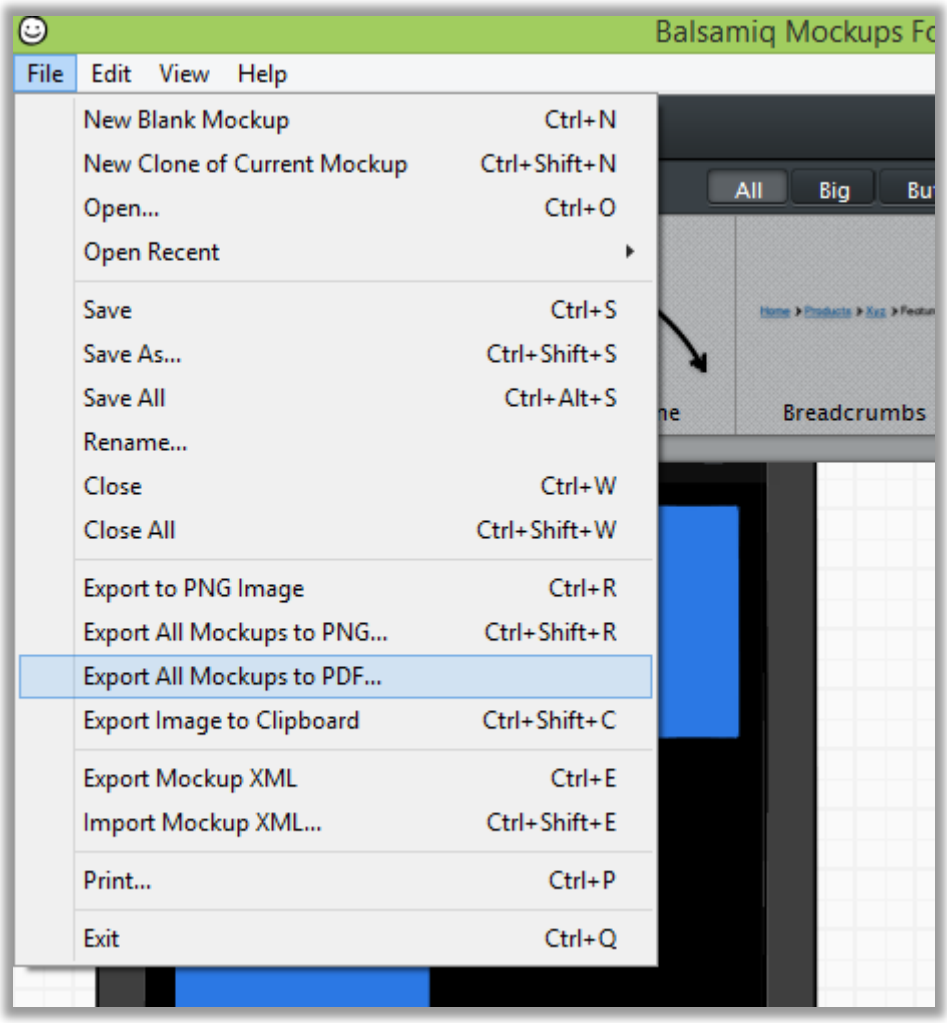


Figura 65 Exportar mockups para PDF

3. Visual Studio e Windows Phone SDK

Este capítulo tem como objetivo demonstrar um tutorial como criar uma nova aplicação para Windows Phone 8. Será demonstrado os passos necessários para criar um novo projeto, adicionar *resources*, correr o programa no emulador e por fim, os passos necessários para utilizar a ferramenta *Store Test Kit* disponibilizada pelo Visual Studio.

3.1. Criar um novo projeto Windows Phone 8

Neste capítulo é demonstrado um tutorial para criar um novo projeto para Windows Phone 8, nomeadamente um projeto com um panorama.

1. No menu do Visual Studio selecionar *File – New – Project*
2. Dos *templates* fornecidos escolher *Panorama App (Windows Phone Silverlight)* como mostrado na Figura 66.

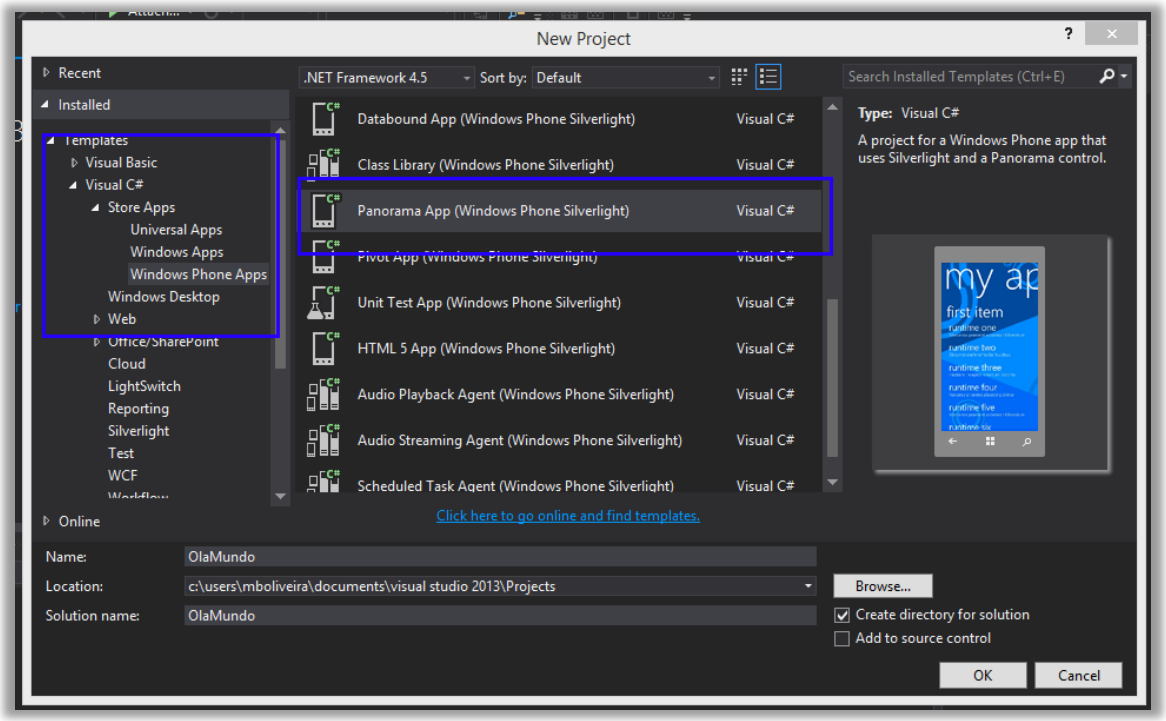


Figura 66 Criar um projeto Olá Mundo

3. Escolher o target Windows Phone 8.0

3.2. Implementação do padrão *Model-View-ViewModel*

No capítulo anterior, é descrito como criar um novo projeto usando um *template* previamente fornecido pela Microsoft. Este *template* já implementa o padrão arquitetural MVVM. É explicado de seguida o modo de funcionamento do mesmo:

1. Criar um *data Model*
    a. Limita-se a uma *class* que implementa a interface *INotifyPropertyChanged* e o evento *PropertyChanged*, como mostrado na Figura 67. Esta *class* permite assim

notificar a vista quando a propriedade *value* muda, fazendo com que a vista seja atualizada.

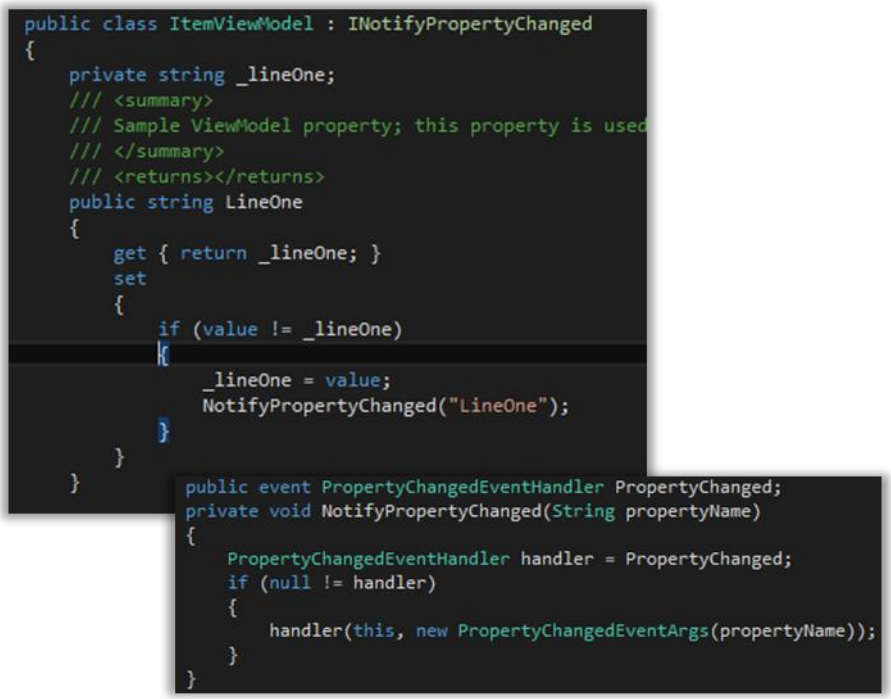
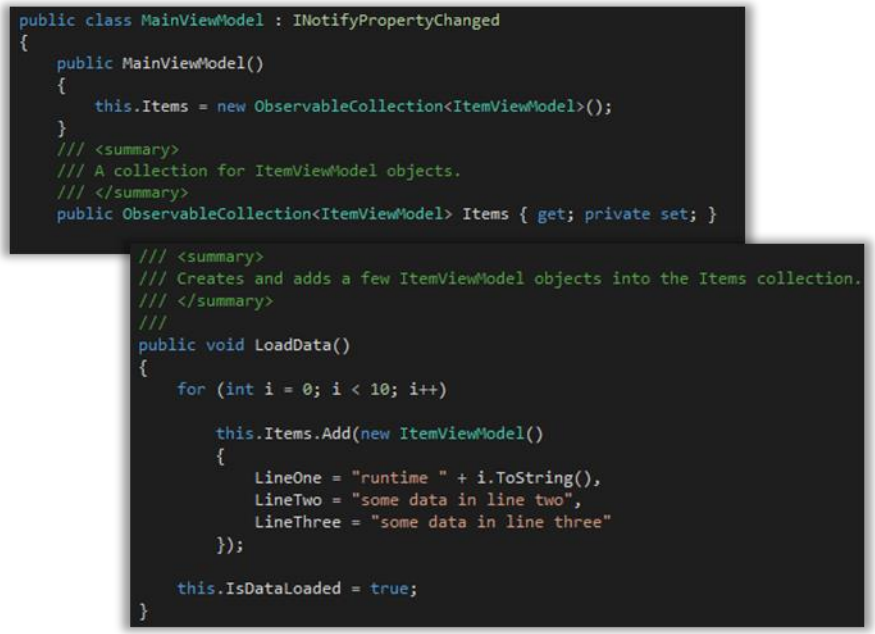


Figura 67 Criar data Model

2. Criar *ViewModel*
    a. O *ViewModel* é uma *class* que exerce a ligação entre a *View* e o *Model*. Como apresentado na Figura 68, esta classe contém uma coleção de objetos do *Model*, usando uma *ObservableCollection<T>*. Isto permite notificar as vistas quando os valores dos itens são mudados.

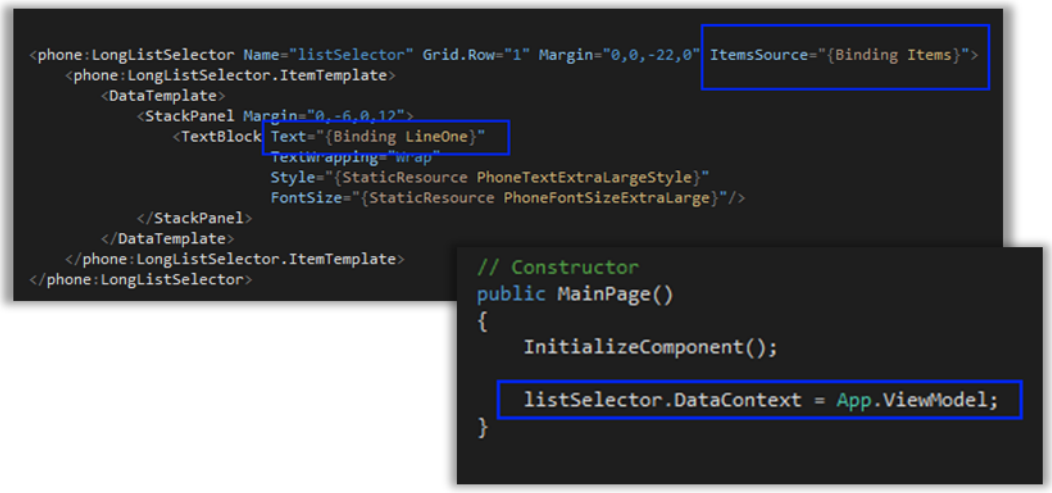


```
public class MainViewModel : INotifyPropertyChanged
{
    public MainViewModel()
    {
        this.Items = new ObservableCollection<ItemViewModel>();
    }
    /// <summary>
    /// A collection for ItemViewModel objects.
    /// </summary>
    public ObservableCollection<ItemViewModel> Items { get; private set; }

    /// <summary>
    /// Creates and adds a few ItemViewModel objects into the Items collection.
    /// </summary>
    ///
    public void LoadData()
    {
        for (int i = 0; i < 10; i++)

            this.Items.Add(new ItemViewModel()
            {
                LineOne = "runtime " + i.ToString(),
                LineTwo = "some data in line two",
                LineThree = "some data in line three"
            });

        this.IsDataLoaded = true;
    }
```

Figura 68 Criar ViewModel

3. Criar a *View*

    a. Para atualizar a *View*, torna-se necessário definir a propriedade *DataContext* do controlo que se quer atualizar, e os atributos da informação a visualizar, de modo a atenuar o *code-behind* e usar apenas *databinding* para tratamento da informação, como se pode ver na Figura 69.



```
<phone:LongListSelector Name="listSelector" Grid.Row="1" Margin="0,0,-22,0" ItemsSource="{Binding Items}">
    <phone:LongListSelector.ItemTemplate>
        <DataTemplate>
            <StackPanel Margin="0,-6,0,12">
                <TextBlock Text="{Binding LineOne}"
                           TextWrapping="Wrap"
                           Style="{StaticResource PhoneTextExtraLargeStyle}"
                           FontSize="{StaticResource PhoneFontSizeExtraLarge}"/>
            </StackPanel>
        </DataTemplate>
    </phone:LongListSelector.ItemTemplate>
</phone:LongListSelector>
```

```
// Constructor
public MainPage()
{
    InitializeComponent();

    listSelector.DataContext = App.ViewModel;
}
```

Figura 69 Criar View

3.3. Adicionar *Resources* para suporte de culturas

1. Na *Solution Explorer* do projeto, clicar com o botão esquerdo do rato na pasta *Resources,* depois selecionar *Add* e de seguida *New Item*
2. Na secção *Visual C#*, escolher *Resources Files*
3. Atribuir o nome *AppResources.pt-PT.resx*
1. Abrir o ficheiro *AppResources.pt-PT* e atribuir os valores *titlePanorama* na coluna *Name* e *Olá Mundo* na coluna *Value* como mostrado na Figura 70

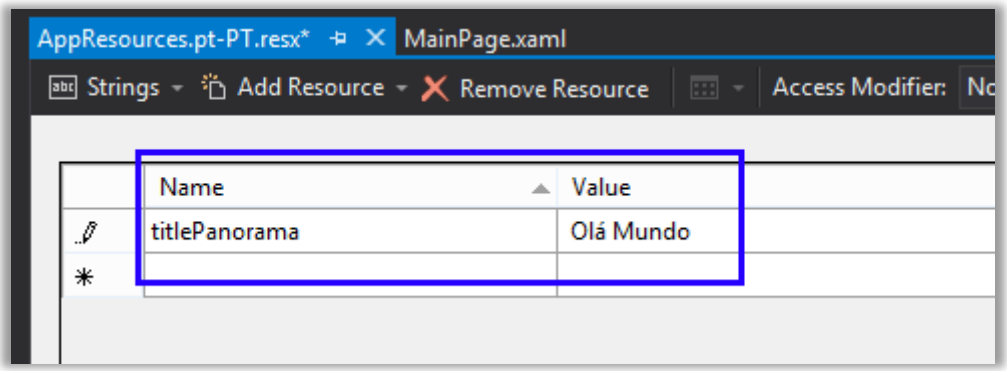


Figura 70 Criar resources

2. Fechar *AppResources.pt-PT* e gravar
3. Clicar com o botão esquerdo do rato no projeto e selecionar *Properties*.
4. Na *tab Application*, selecionar *Portuguese (Portugal)* nas *Supported Cultures* como mostrado na Figura 71

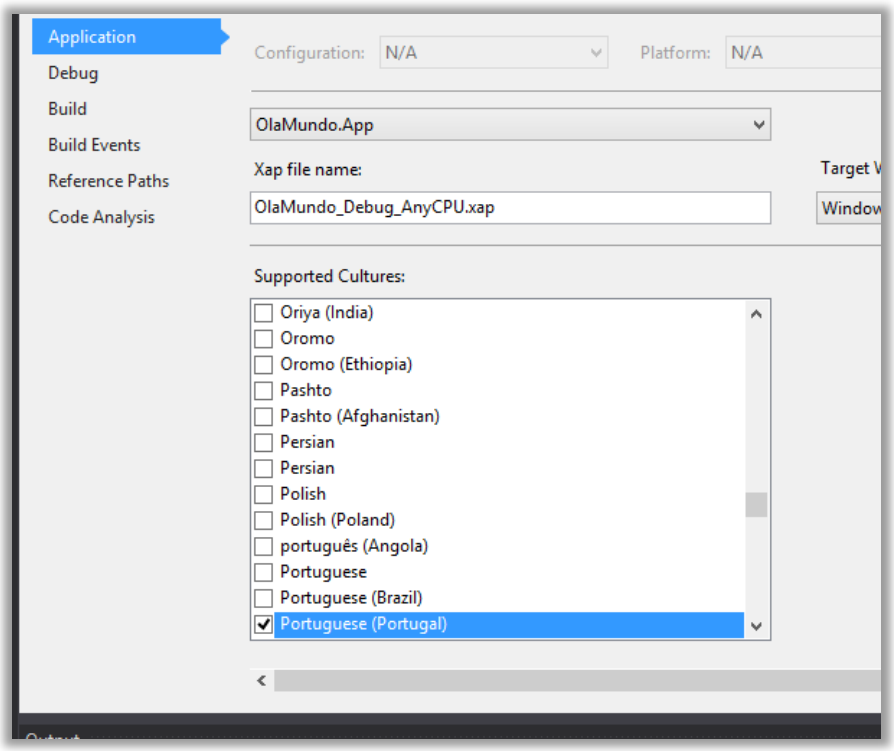


Figura 71 Culturas suportadas

5. Abrir o ficheiro *AppResources* e atribuir os valores *titlePanorama* na coluna *Name* e *Hello World* na coluna *Value*
6. Fechar *AppResources* e gravar
7. Localizar a linha de código `<phone:Panorama Title="my application">` no ficheiro *MainPage.xaml* e substituir por `<phone:Panorama Title = "{Binding Path = LocalizedResources.titlePanorama, Source = {StaticResource LocalizedStrings}}">`

3.4. Correr programa no emulador

É possível testar a aplicação com vários emuladores, para isto apenas é necessário clicar no botão para o efeito e selecionar os emuladores pretendidos, como mostrado na Figura 72.

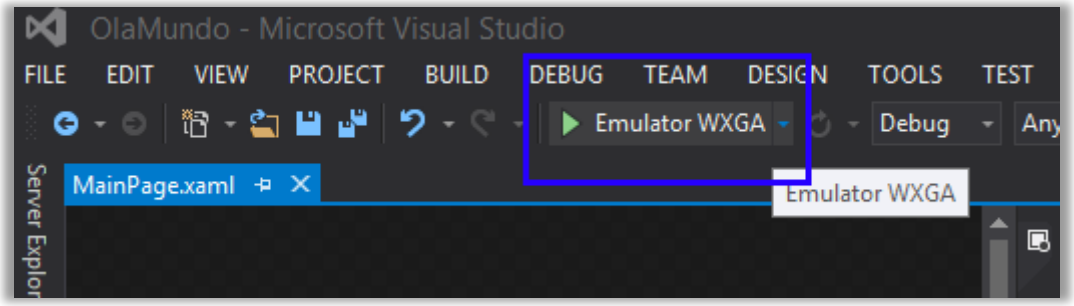


Figura 72 Botão para correr aplicação

A Figura 73 demonstra dois emuladores com resoluções WVGA e WXGA com língua inglesa e portuguesa respetivamente.

Figura 73 Demostração da aplicação com culturas e emuladores diferentes

3.5. Store Test Kit

Para utilizar a ferramenta *Store Test Kit*, é necessário selecionar *Open Store Test Kit* localizada em *Project* do menu do Visual Studio. Aqui são disponibilizadas três secções:

1. *Application Details*
    a. Permite fazer a submissão de imagens para que sejam efetuados testes de modo a preparar a submissão da aplicação na *Windows Phone Store*
2. *Automated Tests*

a. Para correr os testes automáticos é necessário clicar no botão *Run Tests*

3. *Manual Tests*
    a. É apresentada uma lista de testes que deverão ser efetuados manualmente antes de fazer a submissão da aplicação na *Windows Phone Store*
    b. Cada teste poderá ter os seguintes estados
        i. *Pending*
        ii. *Passed*
        iii. *Failed*

4. Blend

O Blend é uma ferramenta que pode ser usada para facilitar a construção de controlos. Esta ferramenta permite fazer *drag and drop* de elementos para a área de edição, ou a partir do XAML.

4.1. Criar uma *storyboard*

Este capítulo tem como objetivo exemplificar a criação de um novo *User Control* que ilustre uma *loading amimation* de um projeto já existente.

1. Abrir o projeto em causa no Visual Studio
2. Selecionar *Project* na barra de menu e depois clicar em *Open in Blend*
3. Após a ferramenta Blend ter sido iniciada, escolher *File* na primeira entrada do menu da ferramenta
4. Selecionar *New Item*
5. Escolher *Windows Phone User Control* e atribuir um nome
6. Arrastar um arco para a área de edição
7. Através da caixa de propriedades do arco, editar o elemento:
    a. RGBA = 57 85 209 100%
    b. ArcThicknessUnit = 15
    c. Opacity = 100%
    d. StartAngle = 40
    e. EndAngle = 360
8. Na *tab Objects and Timeline* adicionar uma nova *storyboard* e atribuir um nome.

A Figura 74 demostra os passos anteriores descritos.

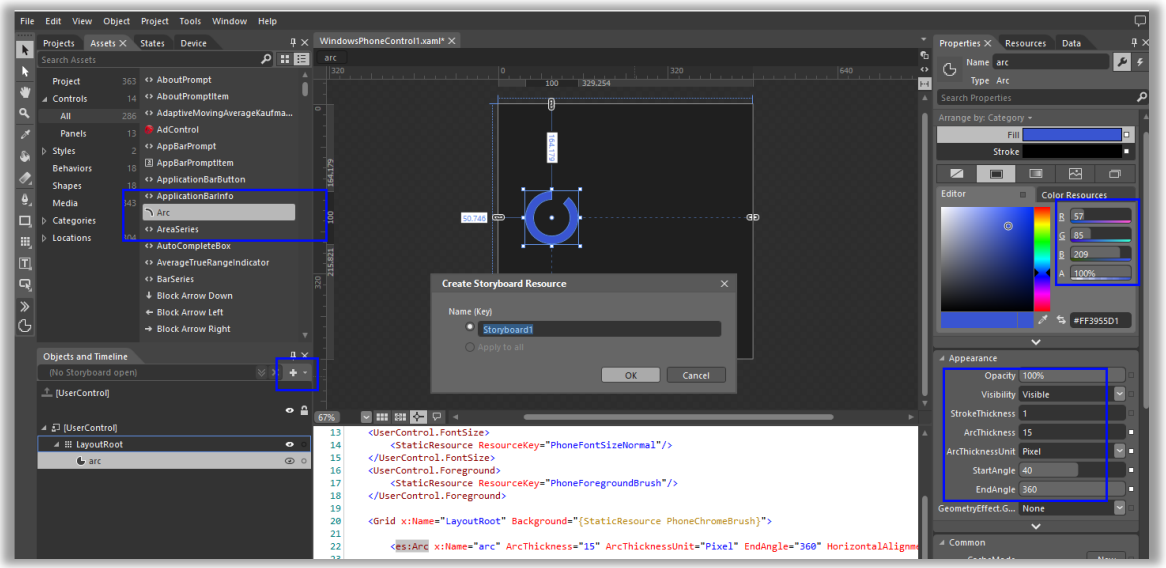


Figura 74 Exemplo de criação e edição de um controlo

9. Na *tab Objects and Timeline* selecionar o ícone responsável para fazer o *Record Keyframe* no início da *Timeline*
10. Proceder as respetivas alterações do arco:
    a. Opacity = 50%
    b. Rotate = 180
11. Selecionar meio segundo na *Timeline* e fazer *Record Keyframe* como se pode visualizar na Figura 75.

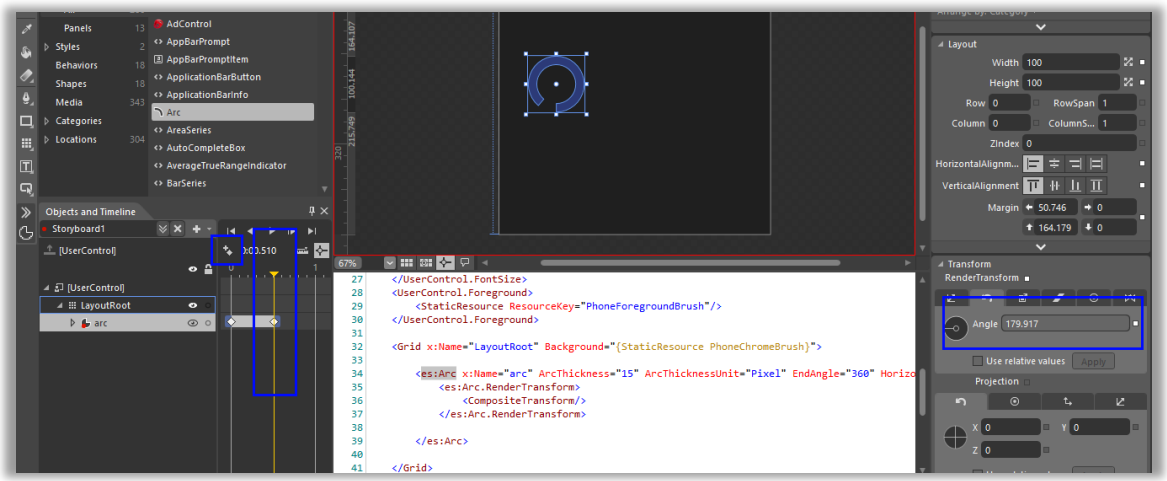


Figura 75 Loading animation

12. Proceder as respetivas alterações do arco:
    a. Opacity = 100%
    b. Rotate = 360
13. Selecionar um segundo na *Timeline* e fazer *Record Keyframe.*
14. Na *tab Objects and Timeline* selecionar o ícone responsável para fazer o *Play à animação*
15. Guardar o item criado através do conjunto de teclas CTRL + S
16. Voltar à ferramenta Visual Studio e fazer *Reload* do *item* criado

Estes passos permitem de uma forma rápida e fácil de criar um *User control* animado num projeto no Visual Studio.

4.2. Editar o estilo de um controlo de fontes externas

1. Abrir o projeto em causa no Visual Studio
2. Selecionar *Project* na barra de menu e depois clicar em *Open in Blend*
3. Após a ferramenta Blend ter sido iniciada, escolher o ficheiro onde o controlo que se pretende alterar reside
4. Com o botão esquerdo do rato, clicar em cima do controlo e selecionar *Edit Template – Edit a Copy*, como mostrado na Figura 76

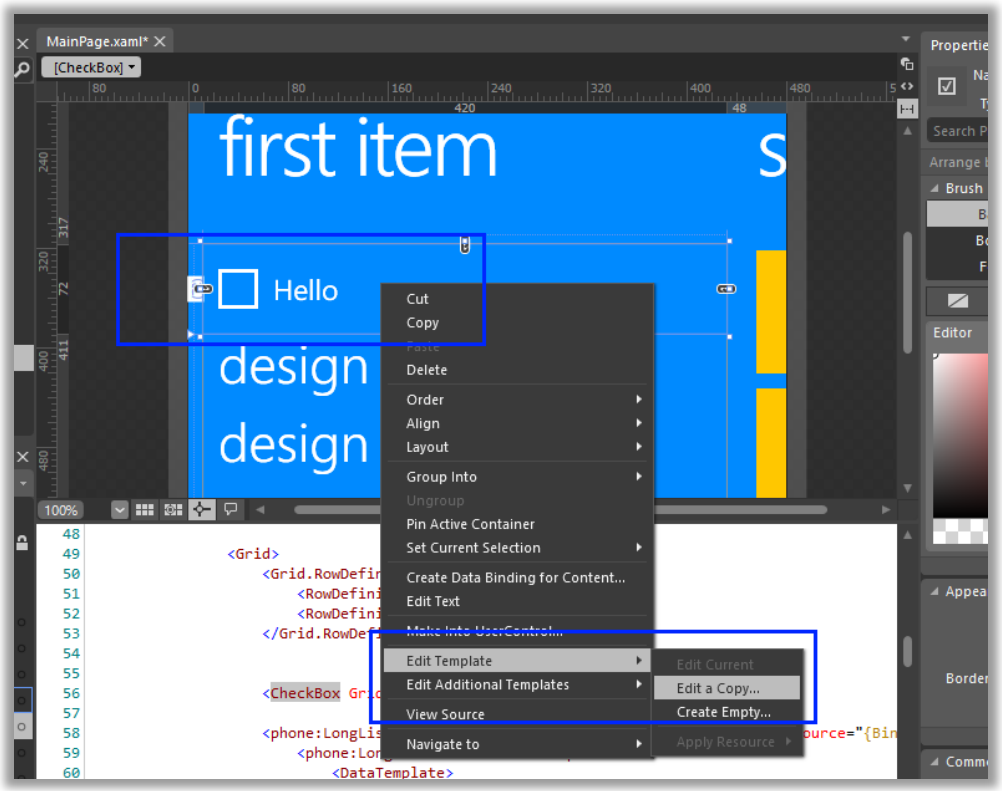


Figura 76 Editar o estilo de um controlo

5. O passo anterior insere automaticamente no código XAML todo o código do estilo desse controlo, inclusive animações se existentes. A Figura 77 mostra um exemplo de como editar uma *checkbox*.

Figura 77 Editar o estilo de uma ckeckbox

6. Guardar o item criado através do conjunto de teclas CTRL + S
7. Voltar à ferramenta Visual Studio e fazer *Reload* do *item* criado
8. A Figura 78 mostra o resultado obtido da *checkbox* editada.

Antes	Depois
	

Figura 78 Checkbox antes e depois das alterações efetuadas