Universidade de Aveiro Departamento de Comunicação e Arte
2015

LUÍS CARLOS VIEIRA ALMEIDA

Plataformas de gestão de cidades inteligentes

Universidade de Aveiro Departamento de Comunicação e Arte
2015

Luís Carlos Vieira Almeida

Plataformas de gestão de cidades inteligentes

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Comunicação Multimédia realizada sob a orientação científica do Doutor Telmo Eduardo Miranda Castelão da Silva, Professor Auxiliar do Departamento de Comunicação e Arte da Universidade de Aveiro.

o júri

presidente

Prof. Doutora Maria João Lopes Antunes
Professor Auxiliar da Universidade de Aveiro

Prof. Doutor João Carlos Lopes Batista
Professor Adjunto da Universidade de Aveiro

Prof. Doutor Telmo Eduardo Miranda Castelão da Silva
Professor Auxiliar da Universidade de Aveiro

agradecimentos

Agradeço aos meus pais, por toda a educação e apoio incondicional que me transmitiram, não só nesta como em todas as etapas da minha vida, e que me levaram a conseguir realizar esta dissertação.

Ao Professor Doutor Telmo Silva, que para além de um excelente professor e orientador que demonstrou um enorme empenho e disponibilidade para este projeto, se tornou um grande amigo.

À Ubiwhere, empresa que me acolheu e impulsionou a minha vida profissional, e que me ajudou no desenvolvimento desta dissertação.

A todos os meus professores, por todos os conhecimentos que me transmitiram ao longo do meu percurso académico, que me possibilitaram aqui chegar.

Ao Tiago, Gil e Ivan, não só pela motivação e companheirismo dados ao longo desta fase, mas sobretudo pela grade amizade vivências proporcionadas durante a nossa vida académica.

A todos aqueles não mencionados que, de uma forma ou outra, me ajudaram a chegar até aqui.

palavras-chave

Cidade inteligente, gestão, plataforma *web*, desenvolvimento, painel de controlo, *frontend development*.

resumo

A percentagem a nível mundial de população a viver em grandes centros urbanos é atualmente de 54%, e a tendência é para que esta aumente. Com o aumento de população a viver em centros urbanos surge uma diminuição da eficiência de prestação de serviços públicos básicos por parte das entidades governadores. O estado atual das tecnologias da comunicação e informação (TICs) permitem que haja uma reestruturação das infraestruturas que dão resposta às necessidades da população em áreas como a segurança pública, tratamento de resíduos, saneamento, fornecimento de energia, entre outras, de modo a que esta prestação seja mais eficiente e sustentável, levando ao conceito de "cidade inteligente" – através da instalação de sensores e actuadores para medição de dados importantes acerca destas áreas, é possível recolher informações que permitam a tomar decisões fundamentadas na gestão de um centro urbano. Assim, é necessária uma plataforma que permita a visualização de toda esta informação, de forma intuitiva e personalizada por parte das entidades reguladoras. Esta investigação visa estudar o processo de desenvolvimento e as tecnologias associadas de uma plataforma para este efeito, neste caso uma plataforma *web*, de modo a obter um protótipo funcional de alta fidelidade de um produto que permita a gestão de uma cidade inteligente, e um conjunto de linhas orientadoras (*guidelines*) para desenvolver plataformas similares.

keywords Smart city, management, *web* platform, development, dashboard, frontend.

abstract At this moment, 54% of the whole global population lives in large urban centers, and the trend is to increase. Alongside with this growth of urban population, emerges a decrease in the efficiency providing the basic public services to the population, like transportation, education, security, sanitation, by the responsible entities. The current state of the communication technologies allows the possibility to restructure the infrastructures that provide the basic services to the population, increasing its efficiency and sustainable, leading to the concept of smart city – though the installation of sensors and actuators, it is possible to collect real time information about different actuation areas, like the occupancy rate of all the car parking spots in the city or how filled are the public waste containers in certain street, allowing the responsible entities to make conscious and grounded decisions. Therefore, it is needed a platform where one can intuitively visualize and analyze all this information. This investigation pretends to study the development process and associated technologies of such a platform, in order to obtain a function high fidelity prototype and a set of guidelines to develop identical platforms.

Índice

Índice das imagens

Índice das tabelas

Índice dos gráficos

1. Introdução

Ao longo dos anos tem-se vindo a assistir a um aumento da percentagem de população mundial a viver em áreas urbanas. Em 2014, mais de metade da população mundial (54%) vive em áreas urbanas, e espera-se que esse número atinja os 66% em 2050. (Department of Economic and Social Affairs, 2014).

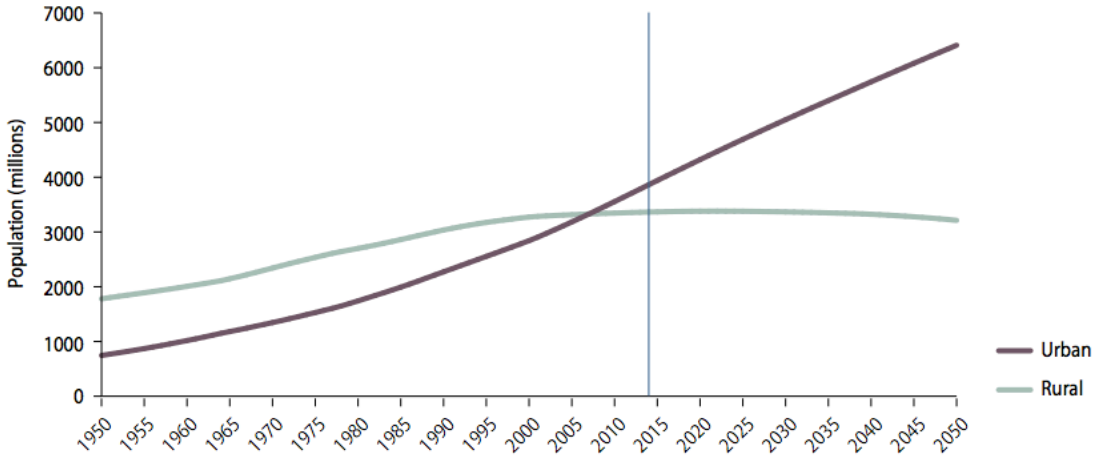


Gráfico 1 | Evolução da população mundial urbana e rural, 1950-2050. Fonte: *Department of Economic and Social Affairs*

Com este aumento de população viver em centros urbanos surgem problemas associados. O governo de uma cidade deixa de ter condições para oferecer de forma eficiente serviços como energia, saúde, educação, transportes, saneamento e segurança pública à medida que a população aumenta (Bhatta, 2010). É necessária uma reestruturação das infraestruturas que apoiam o serviços básicos acima referidos, para que estes se tornem mais eficientes e possibilitem corresponder às necessidades da população de um centro urbano.

As TICs (tecnologias da informação e comunicação) podem fazer parte dessa reestruturação, sendo utilizadas para construir infraestruturas e serviços "inteligentes", designados de "*smart solutions*" (Caragliu, Del Bo, & Nijkamp, 2011).

De acordo com várias investigações, uma cidade inteligente é, na sua essência, uma cidade que faz uso das TICs para construir a infraestrutura e disponibilizar os

serviços básicos de uma cidade (Hernández-Muñoz et al., 2011). De acordo com esta abordagem, podem-se atualmente identificar diversos usos das TICs em cidades e os seus benefícios: ao nível dos transportes, há casos de uso de sensores para controlar o congestionamento, gerir os lugares de estacionamento, e de uma forma geral o trânsito dos transportes públicos; as TICs podem também ser utilizadas para fornecer um melhor serviço de saneamento e de recolha de lixo, utilizando sensores para medir quando e onde é que o lixo necessita de ser recolhido, sendo assim a sua recolha mais eficiente; a segurança pública pode também beneficiar com as TICs, utilizando sistemas de vídeo vigilância mais avançados que se apoiam em sensores de movimento, por exemplo (Hernández-Muñoz et al., 2011).

Estes são apenas alguns exemplos da aplicação das TICs neste contexto, sendo inúmeras as possibilidades que constituem um acréscimo à forma como a cidade controla os seus recursos, gere os seus bens e o disponibiliza os seus serviços. No fundo, pode-se afirmar que as TICs desempenham um papel fundamental para que uma cidade "atinja o estatuto" de inteligente.

Schaffers et al. (2011) refere três processos pelos quais as cidades se devem submeter para se tornarem inteligentes: em primeiro lugar, deve-se assegurar a existência das infraestruturas de comunicação necessárias, como por exemplo de fibra ótica, redes *wireless*, tornando assim possível a conectividade entre pessoas e dispositivos; em segundo lugar, é necessária a instalação de dispositivos como sensores e atuadores, de forma a garantir a captação de dados importantes em tempo real; por último, a criação de aplicações que possibilitem o tratamento e apresentação dos dados às entidades reguladoras (Schaffers et al., 2011), sendo que é neste sentido que surge a pertinência desta investigação.

1.1. Pertinência

Tal como já foi referido anteriormente, os grandes centros urbanos estão cada vez mais povoados, dificultando assim a tarefa das entidades governadoras de oferecerem os serviços básicos esperados pelos seus cidadãos, como por exemplo os serviços de transportes públicos, saneamento, energia, segurança,

entre outros. O facto das TICs, no seu estado de evolução atual permitirem a construção de infraestruturas que possibilitam a recolha e tratamento de dados em tempo real sobre diferentes indicadores de uma cidade (i.e. quantidade de lugares de estacionamento vagos em determinado parque, ou o estado da capacidade de um determinado contentor do lixo), possibilita a criação de plataformas interativas nas quais entidades reguladores podem visualizar toda esta informação e tomar decisões fundamentadas e conscientes nas suas áreas de atuação (saneamento, segurança pública, transportes, etc.).

1.2. Pergunta de investigação

Um projeto de investigação deve ser enunciado desde início sob a forma de uma pergunta de partida, "que tenta exprimir o mais exatamente possível o que procura saber, elucidar, compreender melhor" (Quivy & Campenhoudt, 1998a).

A formulação desta pergunta serve também como uma forma de clarificar as intenções desta investigação e estabelecer limites e objetivos do seu desenvolvimento. Quivy (1998) refere três tipos de qualidades que uma pergunta de partida deve ter, sendo elas qualidades de clareza, exequibilidade e de pertinência.

Uma pergunta de partida, ou de investigação como é aqui chamada, deve ser clara na medida em que não deve ser vaga mas sim precisa, não abrindo assim demasiado o campo de análise.

A pergunta de investigação necessita também de ser realista/exequível. Isto é, deve-se assegurar que existem meios para chegar à resposta a esta pergunta, quer a nível do conhecimento do investigador, quer a nível dos seus recursos de tempo e meios logísticos disponíveis para obter elementos de resposta válidos.

Por fim, a pergunta de investigação deve ser pertinente, referindo-se esta qualidade ao registo em que esta se enquadra – a pergunta não deve procurar julgamentos de valor, mas sim analisar e compreender algo: "O seu objetivo deve ser o do conhecimento, não o de demonstração" (Quivy & Campenhoudt, 1998b).

Tendo em conta as considerações apresentadas previamente, existem agora condições para formular uma pergunta de partida para esta investigação que seja clara, realista e pertinente. Relembrando que o objetivo desta investigação é desenvolver uma plataforma de gestão unificada de uma cidade inteligente, a pergunta de investigação que se formulou é:

"Que aspetos tecnológicos considerar no desenvolvimento de uma plataforma *web* de gestão de uma cidade inteligente?"

Deste modo, a pergunta desta investigação delimita o campo de análise apenas aos aspetos tecnológicos da plataforma que pretendemos desenvolver e especifica também que se trata de uma plataforma *web,* limitando aqui o estudo apenas para este tipo de plataformas. É claro necessário especificar o objetivo da plataforma, sendo neste caso a gestão de uma cidade inteligente.

1.3. Motivações

A *Internet of Things* (IoT) e as cidades inteligentes são duas das áreas estratégias para o crescimento e internacionalização da Ubiwhere, nas quais está a investir esforços para a resolução de vários problemas do mundo real, sendo o problema explicado em pontos anteriores deste documento um deles.

Neste contexto, o investigador surge como um colaborador que pretende ajudar a empresa na resolução do problema, recorrendo aos conhecimentos de desenvolvimento de plataformas *web* ricas em interação e informação. O gosto pelo desenvolvimento deste tipo de plataformas, o facto do tema na qual essa plataforma é inserida ser o tema das cidades inteligentes, que se pode considerar um tema atual e inovador, e também o facto desta investigação se desenrolar em parceria com a Ubiwhere que procura a resolução deste problema em específico para o seu crescimento a nível internacional, constituem as motivações principais para o desenvolvimento da investigação aqui apresentada.

1.4. Finalidades e Objetivos

O principal objetivo desta investigação é responder à pergunta de investigação, contudo, é importante relembrar que essa pergunta de investigação foi "construída" de acordo com um desafio lançado pela empresa Ubiwhere, que passa por desenvolver um protótipo de alta fidelidade de um portal de gestão unificada de uma cidade inteligente. Para tal, e em coordenação com a empresa em questão, foram estipuladas as fases de desenvolvimento deste projeto:

- Análise do estado da arte;
- *Wireframing* e prototipagem da plataforma;
- Definição da arquitetura da informação da plataforma;
- Desenvolvimento da plataforma *web* (componente *frontend*);
- Testes funcionais.

É possível verificar que "missão" do investigador não se restringe apenas ao desenvolvimento da plataforma em si. Faz parte dos objetivos traçados para esta dissertação analisar o estado da arte, realizar os protótipos e *wifreframes* da plataforma e também realizar e analisar o resultado de testes funcionais.

Nesta fase é importante também caracterizar a fase de implementação das funcionalidades do portal: esta dissertação foca-se apenas no desenvolvimento da componente *front-end* da plataforma, sendo a empresa encarregue de disponibilizar materiais e ferramentas para o seu desenvolvimento, mais propriamente, os *layouts* finais que devem ser implementados, e a API (componente *backend*) que irá possibilitar ao investigador ter toda a informação necessária a disponibilizada no portal. A empresa tem também o papel de indicar quais as tecnologias a serem utilizadas no desenvolvimento da plataforma, ainda a que o investigador possua a liberdade para adotar quaisquer outra tecnologia ou ferramenta, desde que devidamente justificada a sua necessidade.

2. Metodologia de Investigação

Este capítulo surge em conjunto com a necessidade de alcançar um rigor científico considerável para uma investigação deste tipo, que possibilite posteriormente uma avaliação e validação positiva, tanto por parte do próprio autor como de outros que procurem investigações na mesma área de estudos. Este capítulo serve também para especificar com clareza qual o plano desta investigação e fundamentar cada um dos passos a seguir.

Independentemente da área de estudos é necessária uma investigação profunda para encontrar respostas e esclarecer questões que surgem ao longo do processo de escrita de uma dissertação. Quivy & Campenhoudt (1998) referem que o processo de investigação ajuda, entre outros aspetos, a "compreender melhor os significados de um acontecimento ou de uma conduta" e "a fazer inteligentemente o ponto da situação" (Quivy & Campenhoudt, 1998a). Coutinho (2011) explica também a extensão do processo de investigação ao afirmar que este necessita de ser planeado "desde o momento em que se seleciona a problemática, se formulam as hipóteses, se definem as variáveis e se escolhem instrumentos, até à fase em que se interpretam e comunicam os resultados." (Coutinho, 2011), verificando-se assim a importância de planear e especificar o processo de investigação, visto ser uma atividade em constante desenvolvimento no decorrer desta dissertação.

Como já foi referido no capítulo das Finalidades e Objetivos, o investigador procura em colaboração com a empresa *Ubiwhere,* e de forma a dar resposta a um desafio lançado por esta, desenvolver uma plataforma *web* que ajude as entidades reguladoras de um centro urbano a tomarem decisões conscientes em áreas de atuação como a gestão da recolha do lixo, parqueamento de veículos automóveis, saneamento, gestão de fornecimento de energia, entre outros. Existem contudo limitações para o desenvolvimento da plataforma, que passam pelas ferramentas e materiais que a empresa deve fornecer ao investigador e pelos prazos se finalização, o que leva à necessidade de impor limites no desenvolvimento e objetivos claros para aquilo que se espera como resultado final

para esta dissertação. Estes objetivos estão referidos no capítulo das Finalidades e Objetivos na introdução desta dissertação, sendo que foram acordados entre o investigador e a empresa em questão.

Segundo Lia Oliveira (2006), e tendo em conta que esta investigação pretende desenvolver uma plataforma com funcionalidades específicas, para um público específico, sendo importante analisar procedimentos a tomar e a opinião dos intervenientes, o procedimento metodológico desta investigação será o de investigação de desenvolvimento (Oliveira, 2006). Lia Oliveira refere também, citando Van der Maren (1996), que este tipo de investigação " (...) pode tomar três formas: desenvolvimento de conceito, desenvolvimento de objeto e desenvolvimento ou aperfeiçoamento de habilidades pessoais enquanto utensílios profissionais". Numa outra abordagem, Rita Richey, James Klein e Wayne Nelson subdividem a investigação de desenvolvimento em dois tipos, que estão descritos na seguinte tabela retirada do estudo dos mesmos autores:

	Type 1	Type 2
Emphasis	Study of specific product or program design, development, &/or evaluation projects	Study of design, development, or evaluation processes, tools, or models
Product	Lessons learned from developing specific products and analyzing the conditions that facilitate their use	New design, development, and evaluation procedures &/or models, and conditions that facilitate their use
	Context-specific Conclusions ⇒ ⇒ ⇒	**Generalized** ⇒ **Conclusions**

Tabela 1 | Sumário dos dois tipos de investigação de desenvolvimento (Richey, Klein, & Nelson, 2004)

Como se pode verificar, o tipo 2 foca-se no estudo do design, desenvolvimento ou processos de avaliação, e das ferramentas ou modelos para que se atinja um produto que é normalmente um novo procedimento de desenvolvimento, design ou avaliação e/ou de modelos e condições que facilitem o seu uso (Richey, Klein, & Nelson, 2004).

No caso desta investigação, pode-se concluir que se enquadra no Tipo 2 descrito na Tabela 1, pois são estudados os procedimentos implícitos ao desenvolvimento de um produto para que assim se possam tomar decisões conscientes e fundamentadas no decorrer desse mesmo desenvolvimento. Como resultado deste estudo pretende-se obter conclusões "generalizadas" sobre estes procedimentos e caminhos a seguir no desenvolvimento de uma plataforma *web* com as características e nas condições já referidas, de modo a ajudar outros investigadores que se encontrem na mesma situação nos seus próprios processos de desenvolvimento.

Deste modo, esta investigação começa "(...) por analisar o possível objeto (...) conceptualizar esse objeto (...) elaborar estratégias de realização, avaliar as possibilidades de concretização, proceder à construção de uma forma provisória desse objeto (protótipo) e implementá-lo" (Oliveira, 2006). Para tornar esta conceptualização possível, e para entender melhor as tecnologias que vão fazer parte deste processo de realização do objeto de estudo, é necessário enquadrar teoricamente, tanto os leitores como o próprio investigador, nos conceitos e tecnologias adjacentes ao tema.

O próximo capítulo tem como finalidade aprofundar e criar bases de conhecimento adequadas para que seja possível fundamentar as escolhas do investigador no desenvolvimento no objeto de estudo que é uma plataforma *web* de gestão de uma cidade inteligente.

3. Enquadramento teórico

Neste capítulo desta dissertação, o investigador pretende enquadrar teoricamente os leitores nos conceitos chave da investigação, definindo e especificando com detalhe cada um desses conceitos, dando assim meios para uma melhor compreensão do assunto a ser investigado, e do produto que se pretende desenvolver.

3.1. Cidades Inteligentes

A cidade inteligente é um conceito presente em todas as etapas desta investigação.

A definição para "cidade inteligente" está longe de ser clara. Hollands (2008), num artigo intitulado *"Will the really smart city please stand up?"* refere a complexidade deste conceito, afirmando que existe uma tendência em se intitular cidades como inteligentes, sendo muitas das vezes errada esta assunção (Hollands, 2008). Contudo, Kominos (2006), constrói uma definição para o termo "*intelligent city*", sendo a seguinte: *"...territories with high capacity for learning and innovation, which is built-in the creativity of their population, their institutions of knowledge creation, and their digital infrastructure for communication and knowledge management*".

Para melhor se perceber a importância deste tema, num fórum mundial sobre cidades inteligentes, em 1997, sugeriu-se que 50.000 (cinquenta mil) cidades a nível mundial iriam adoptar iniciativas "inteligentes" até 2007 (Hollands, 2008). Num estudo mais recente da IHS *Technology* (NYSE: IHS), Arrowsmith (2014) refere que o número de cidades inteligentes vai quadruplicar num intervalo de tempo de 12 anos, que começou em 2013 e acaba em 2025, passando assim de 21 para pelo menos 88 por esta altura. Para os efeitos deste estudo, uma cidade inteligente é "(...) *one that has deployed, or is currently piloting the integration of, ICT solutions across three or more of mobility and transport, energy and sustainability, physical infrastructure, governance, and safety and security city-functions in order to improve efficiency, manage complexity and enhance citizen*

quality of life, leading to a sustainable improvement in city operation" (Arrowsmith, 2014). No gráfico 2, apresentado abaixo, pode-se verificar a evolução do número de cidades inteligentes, subdivididas pelas funções referidas na definição de cidade inteligente no parágrafo anterior, até 2025.

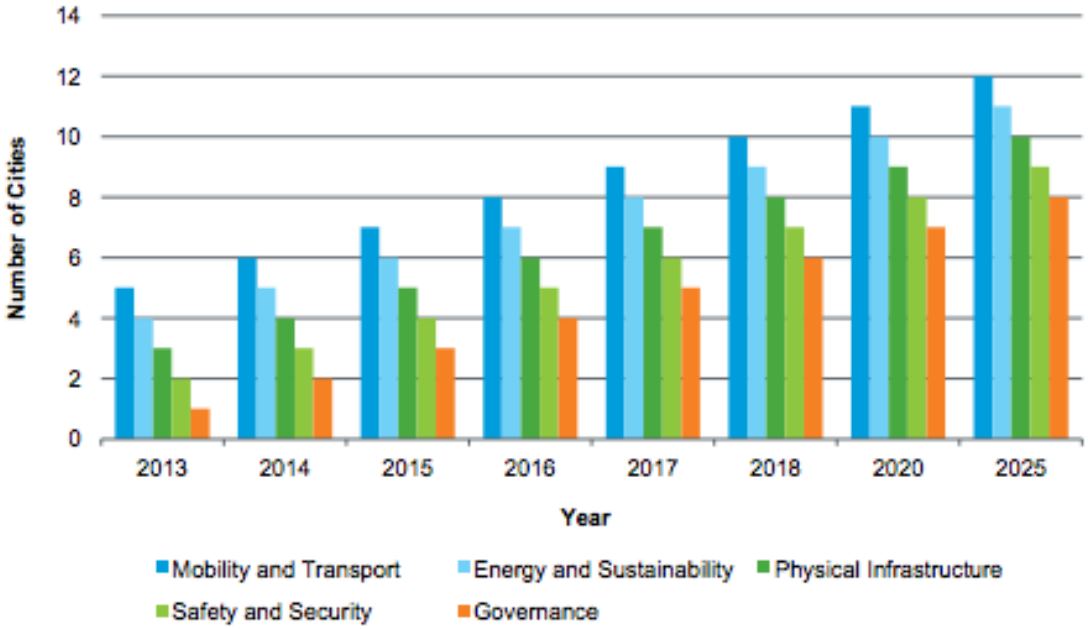


Gráfico 2 | Evolução do número de cidades inteligentes pela sua função, até 2025 (Fonte: *IHS Technology, 2014*)

Os casos mais importantes de cidades inteligentes hoje em dia são, segundo Hollands (2008), *San Diego, San Francisco, Ottawa, Brisbane, Amsterdam, Kyoto* e *Bangalore.* Estas cidades estão na "frente da corrida" das cidades inteligentes, e estabelecem "tendências" a seguir por outras (Hollands, 2008).

A apoiar este conceito de cidades inteligentes e para tornar toda esta ideia possível, está presente uma base tecnológica profunda com muitos anos e investimento em investigação e desenvolvimento em diversas áreas. Nos capítulos que se seguem explicam-se os "pilares" tecnológicos que apoiam o conceito de *"smart cities"* e que no fundo as tornam possíveis.

3.2. Internet of Things (IoT)

Ashton (2009), que afirma ter sido o primeiro a utilizar este termo em 1999, refere que os computadores, e por consequente a Internet, hoje em dia são dependentes

em grande parte de informação que é gerada e processada por seres humanos. Isto torna-se um problema, pois a típica pessoa do século XXI tem tempo, eficiência e atenção muito limitados para gerar e processar essa informação. Surge a necessidade de passar essa tarefa para outros agentes, de modo a que a informação seja adquirida e processada de uma forma totalmente automatizada, sem intervenção extra do ser humano, sendo esta a ideia principal da "internet das coisas".

"*We need to empower computers with their own means of gathering information, so they can see, hear and smell the world for themselves..."* (Ashton, 2009).

No contexto desta investigação, a IoT é um assunto crucial a ser estudado pois é através deste conceito que se consegue chegar à recolha e processamento de toda a informação proveniente das "coisas" de um centro urbano, e que posteriormente possibilita a construção de uma plataforma que recebe esta informação e a apresenta de forma totalmente automatizada, sem a necessidade da intervenção do ser humano.

3.3. Machine-to-Machine Communications (M2M)

Machine to Machine (M2M) é um termo utilizado para descrever tecnologias que permitem computadores, sensores, actuadores e dispositivos móveis que comuniquem entre si sem a necessidade de invenção humana (Watson, Piette, Sezgen, Motegi, & ten Hope, 2004).

Chen (2013) apresenta um modelo (Imagem 1) que se divide em quatro "camadas" (*layers*) e descreve o funcionamento deste tipo de plataformas comunicação: a primeira camada refere-se à recolha de informação através de sensores aplicados às mais diversas "coisas" da IoT; a camada seguinte (segunda) trata-se da camada de transporte dessa informação fundamentalmente através da internet; a terceira camada está encarregue de processar a informação que foi recolhida e transportada; a quarta e última camada é chamada camada aplicacional e de serviços, e é onde se processa a utilização da informação em cenários como o de uma cidade inteligente (Chen, 2013).

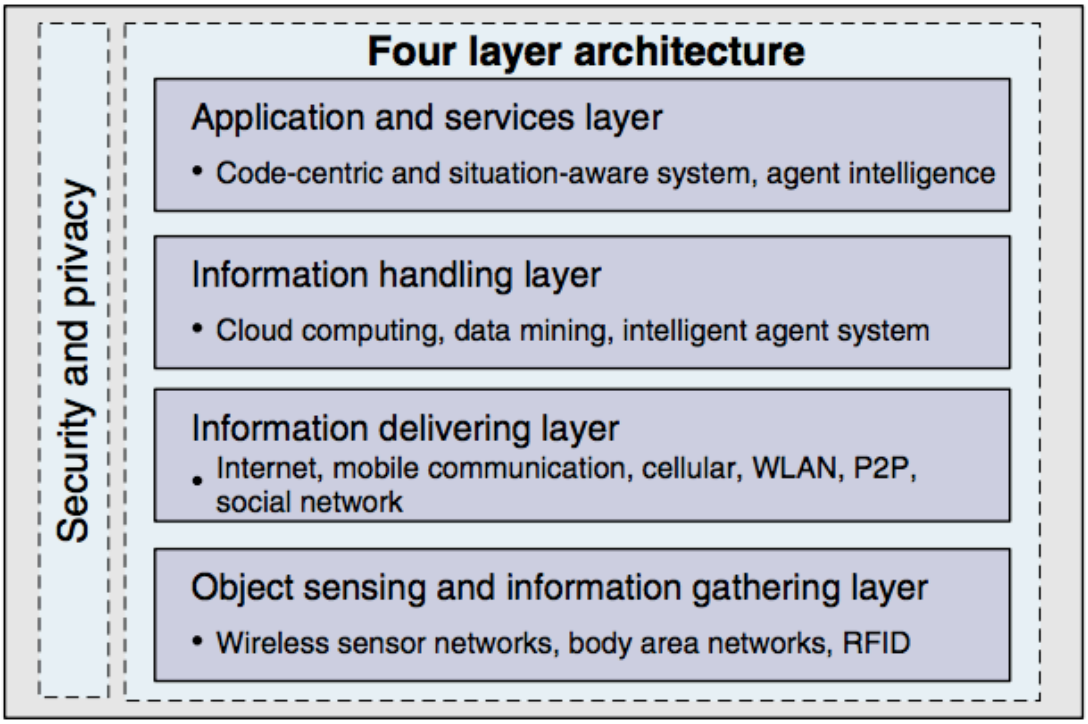


Imagem 1 | Modelo proposto por Chen (2013) para a M2M

Este modelo, ainda que generalizado, representa o conceito da comunicação M2M, que está interligado com o conceito da IoT explicado anteriormente e é um dos conceitos base no tema das cidades inteligentes.

3.4. Desenvolvimento para a plataformas *web*

Sendo o objetivo final desta investigação obter uma plataforma *web,* independentemente das suas características, existem vários conceitos e tecnologias adjacentes que devem ser referidos previamente. Este capítulo tem como objetivo fazer uma abordagem às plataformas *web*, referindo os conceitos e as tecnologias base envolvidos no desenvolvimento destas.

3.4.1. Plataformas web

Apenas há alguns anos atrás, a *World Wide Web* (Web) era utilizada para disponibilizar informação estática aos utilizadores através de ficheiros de texto simples interligados através de *hyperlinks,* contudo o cenário mudou drasticamente e hoje assistimos a exemplos de plataformas *web* bastante

complexas e dinâmicas para os mais diversos fins como por exemplo para o comércio e entretenimento (Offutt, 2002).

O termo plataforma *web*, utilizado nesta investigação para caracterizar o produto que se pretende desenvolver, pode ser substituído por aplicação *web*, ou simplesmente *website*. Offtutt (2002) utiliza o termo *"web software application"* para se dirigir a este tipo de plataformas. De acordo com o mesmo autor, esta "categoria" de plataformas/aplicações tem a característica de envolver uma grande diversidade de tecnologias, linguagens e estilos de programação complexos, utilizar ligações a bases de dados e interagir diretamente com o utilizador. Pode-se assim afirmar que este tipo de aplicações são bastante complexas e o seu desenvolvimento pode ser muito demoroso e dispendioso devido à elevada quantidade de tecnologias associadas e ao cuidado que se deve ter na apresentação da informação.

Offtutt (2002) explica o modelo de utilização e distribuição deste tipo de aplicações, fazendo uma comparação das diferentes partes envolvidas entre a primeira geração de aplicações *web* e as atuais.

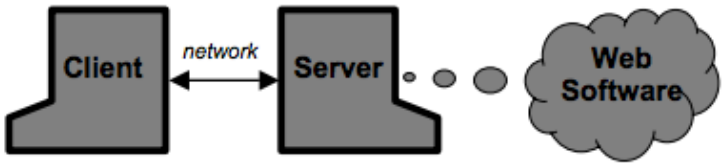


Imagem 2 | Esquema representativo do modelo da primeira geração das plataformas *web* (Offtutt, 2002)

Na Imagem 2 verifica-se um esquema representativo do modelo da primeira geração de plataformas *web,* elas são compostas por três grandes componentes – o cliente, o servidor e o *software.* Na realidade, o cliente pode ser visto como um computador que contém um *web browser* instalado, o servidor é outro computador, que recebe um pedido do cliente através da internet e transmite de volta o *software* que é interpretado pelo *web browser* para que o utilizador consiga ver a informação.

Segundo o autor, este modelo é utilizado por aplicações relativamente pequenas, não assegurando a segurança nem a escalabilidade necessárias que aplicações mais complexas exigem.

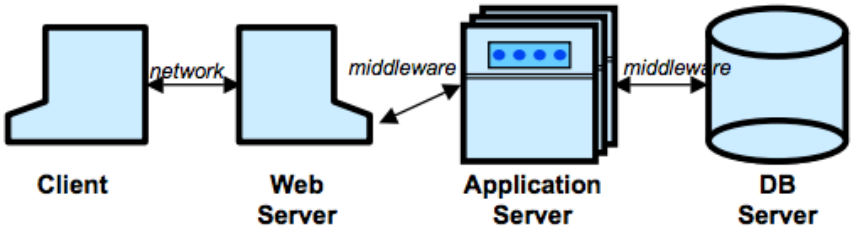


Imagem 3 | Esquema representativo do modelo das atuais plataformas *web* (Offtutt, 2002)

A Imagem 3 representa o modelo das atuais plataformas *web*. Neste modelo, e em comparação com o modela da Imagem 1, o cliente continua a ser um *web browser* utilizado pelo utilizador para visitar um *website*, e o servidor (*web server*) continua a desempenhar o papel de receber o pedido do cliente e distribuir os ficheiros necessários para apresentar a informação no *browser*. Contudo, e para aumentar a segurança, eficiência, escalabilidade e disponibilidade do serviço, o *software* da aplicação é transferido para uma nova componente, o servidor aplicacional (*application servidor)*. O servidor aplicacional pode ainda estabelecer comunicações com uma ou mais bases de dados, que são as componentes que armazenam toda a informação "dinâmica", da aplicação, como por exemplo a informação relativa a um utilizador.

Uma característica comum nas aplicações/plataformas *web* é a sua transparência para o utilizador final. Isto é, o utilizador não necessita de instalar o *software* nem ter em atenção o sistema operativo do seu computador, apenas necessita de assegurar que tem uma ligação à internet e um *web browser*, tornando-se assim o processo de utilização de uma plataforma *web* muito rápido em comparação com outros tipos de plataformas.

3.4.2. Tecnologias de suporte

É essencial fazer uma apresentação das tecnologias que dão suporte ao desenvolvimento da plataforma que é o foco desta investigação. Devido à elevada

quantidade de tecnologias, ferramentas, e linguagens de programação que suportam o desenvolvimento para a *web,* neste capítulo apenas se referem as tecnologias mais "apoiadas" e utilizadas pela comunidade de *developers* a nível internacional, e também aquelas que realmente fazem parte da componente a desenvolver desta plataforma.

De forma a apurar quais estas tecnologias que necessitam de destaque nesta investigação, o investigador baseou-se na informação recolhida em diversas plataformas *web* orientadas para os *developers*, que promovem uma discussão aberta sobre este tipo de questões e onde se podem verificar certas tendências para a utilização de uma tecnologia e a não utilização de outra, bem como *blogs* e pequenos artigos de autores que fazem balanços sobre as melhores tecnologias da atualidade nesta área. Exemplos destas plataformas são o *GitHub* (GitHub, 2015), que constitui uma plataforma de partilha de código, na qual se podem verificar as tecnologias utilizadas pelos melhores *developers* e empresas a nível mundial; o *Stackshare* (Stackshare Inc, 2015) que é uma plataforma onde se pode ver em detalhe todas as tecnologias utilizadas na construção de certa plataforma (por exemplo, é possível verificar quais as tecnologias que o *Facebook,* uma plataforma reconhecia e das mais utilizadas mundialmente, utiliza para construir o seu produto); o *Bower* (Bower, 2015) sendo esta não só uma plataforma mas sim uma tecnologia que se pode utilizar no desenvolvimento, mas que permite a procura de tecnologias e ferramentas e verificar quais as mais utilizadas e reconhecidas; por fim, diversos *blogs* como o *SmashingMagazine* (Smashing Magazine, 2015)*, David Walsh* (Walsh, 2015)*, DailyJS* (DailyJS, 2015), que são *blogs* orientados exclusivamente para o desenvolvimento *web*, mais precisamente para a componente *front-end* (vai ser explicado nos parágrafos seguintes) e que promovem a discussão sobre as melhores tecnologias e *"best practices*" no desenvolvimento. É também importante relembrar que esta investigação é realizada em parceria com uma empresa com um elevado *know-how* na área do desenvolvimento de plataformas *web,* e que apoia e orienta o investigador nestas escolhas, e portanto tem também ela um papel fundamental na escolha das tecnologias a utilizar.

Com o objetivo de clarificar melhor estas tecnologias de suporte para o desenvolvimento de plataformas/aplicações *web,* parte-se do príncipio de que estas tecnologias e ferramentas estão agrupadas em dois grandes grupos: tecnologias para desenvolvimento *front-end* e tecnologias para desenvolvimento *back-end*.

Tal como foi referido no capítulo anterior, uma plataforma *web* é servida através de um modelo onde existe o lado do cliente (*cliente side*), e o lado do servidor e das bases de dados (*server side*). Esta divisão pode ser também considerada nas tecnologias utilizadas em cada um destes lados para o seu desenvolvimento, sendo que as tecnologias *front-end* predominam no *cliente side* e as tecnologias *back-end* no *server-side*.

Numa publicação de um *blog* dedicado ao desenvolvimento *web*, o autor utiliza o *iceberg* para representar esta divisão das tecnologias (Imagem 4).

Imagem 4 | Representação da divisão entre tecnologias *frontend* e *backend*. Fonte: Skillcrush, 2012

Muito semelhante a um *website*, num *iceberg* apenas uma pequena parte é observável pelo ser humano (utilizador neste caso). Esta parte visível diz respeito, num *website* à sua interface, botões, texto e formulários, ou seja, os conteúdos

que são visíveis e interativos pelo utilizador. Esta parte de um *website* / plataforma *web* é chamada de *frontend* e, sendo apenas uma pequena parte da totalidade.

A parte de *backend* é a parte do *iceberg* que está submersa e é idealmente invisível para o utilizador – é nesta parque que é investido o maior esforço e conhecimento no desenvolvimento de uma plataforma *web.*

"The frontend is the part of a web site that you can see and interact with, while the backend is all the rest." (Skillcrush, 2012)

No contexto desta investigação, é importante focar as atenções para a "parte visível do *iceberg*", uma vez que o objetivo é desenvolver a parte *frontend* da plataforma de gestão de cidades inteligentes. Assim sendo, a componente de *backend* será menos explorada nesta investigação, ficando a empresa *Ubiwhere* encarregue de a desenvolver e fornecer todo o material necessário.

De seguida importa analisar com mais cuidado a Imagem 3 e verificar que a parte visível do *iceberg* contém três nomes: HTML, CSS e Javascript. Estas são as três tecnologias base do desenvolvimento *frontend* que interessa analisar com mais profundidade.

3.4.2.1. HTML

HTML (abreviatura para *Hyper Text Markup Language*) é uma linguagem de marcação. Uma linguagem de marcação, neste contexto, é uma linguagem que através de uma sintaxe específica indica ao *web browser* como apresentar uma página *web* (Mozilla Developer Network, 2015a). Esta linguagem utiliza um conjunto de *"tags*" (marcas) para encapsular o diferente conteúdo de uma página, sendo que cada uma destas *tags* tem um papel diferente a desempenhar. *Tags* são palavras entre os símbolos de menor (<) e maior (>), por exemplo: "", e são usualmente utilizadas em pares sendo a segunda de cada par composta também por uma barra, significando o fim do "bloco" de conteúdo. A esta *tag* chama-se "*closing tag*" (Mozilla Developer Network, 2015a).

A W3C (*World Wide Web Consortium*) é a entidade responsável por definir as normas de utilização desta linguagem e as *tags* a serem utilizadas, ou seja, os *standards* de utilização do HTML. Atualmente, esta linguagem é chamada de HTML5 na sua versão mais recente, com uma lista maior de *tags* e outras funcionalidades disponíveis para tirar maior proveito do *browser* do cliente.

Resumindo, o HTML é uma linguagem que está presente em todas as páginas *web* e permite aos seus autores publicar documentos com texto, cabeçalhos, tabelas, listas, fotos, vídeos, som, entre outros, criar formulários de pagamentos, pedidos de produtos, reservas, etc. para preenchimento dos utilizadores, e ainda incluir outras aplicações e programas diretamente no documento para possibilitar a criação de funcionalidades mais avançadas (W3C, 2015)

3.4.2.2. CSS

CSS é uma abreviatura para *Cascading Style Sheets*, e é a linguagem que descreve a apresentação de uma página *web*, incluindo as suas cores, dimensões, tipos e estilos de letra, posicionamento, entre outros, sendo que ainda permite que uma página *web* adapte a sua apresentação para diferentes tamanhos de ecrãs (W3C, 2015).

Esta linguagem diretamente relacionada com o HTML, pois o HTML trata da estrutura dos conteúdos enquanto que o CSS foca-se na sua personalização.

Através da atribuição de atributos às *tags* do HTML, é possível utilizar o CSS para se referir especificamente a um bloco de conteúdos do HTML e designar a sua cor, tamanho, etc.

Esta é uma linguagem também ela *standardizada* pelo W3C, que define as propriedades que se podem personalizar e as normas de utilização.

3.4.2.3. Javascript

Javascript é uma linguagem de programação que pode ser incluída em páginas *web* para as tornar mais interativas (Wilton-Jones, 2011). Entre outras funções, esta linguagem permite tornar uma página *web* estática em algo mais dinâmico,

por exemplo, com validações de formulários, detecção de cliques em botões para modificações de conteúdo, etc. Ou seja, por natureza, uma página *web* é estática, contendo apenas texto ou imagens e vídeos ou áudio, mas que não tem qualquer tipo de interatividade com o utilizador, isto é, não permite que o utilizador clique em certos pontos, faça transições, esconda e apresente certos conteúdos, etc.. *Javascript* é a tecnologia que permite adicionar uma "camada" de interatividade à página *web* para que a experiência do utilizador seja mais rica e interativa.

Esta linguagem é "*client-side*" – isto significa que, e referindo novamente o esquema da Imagem 2, é uma linguagem que reside e funciona do lado do computador do utilizador, mais especificadamente, no *web browser*.

Javascript é também conhecido como a linguagem de programação para a *web* (Mozilla Developer Network, 2015b), e é utilizada em praticamente todas as plataformas *web* mais recentes.

Depois desta breve introdução a cada uma das tecnologias base da componente *frontend* de um *website*, segue-se uma outra tecnologia que já não se enquadra nesta componente, mas sim na componente *backend*. Apesar desta investigação se dedicar ao desenvolvimento *frontend* da plataforma *web,* é contudo importante o seu estudo pois vai ser utilizada para desenvolver determinada componente da plataforma.

3.4.2.4. *Django*

"*Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design*" (Django Software Foundation, 2015).

Django é uma *framework* que através de um conjunto de métodos, ferramentas e regras pré-estabelecidas, permite agilizar algumas tarefas do desenvolvimento de uma aplicação *web.* A utilização desta ferramenta é imposto pela empresa que lançou o desafio desta investigação, e portanto é importante referir a sua função na plataforma.

Ao contrário das três tecnologias anteriormente referidas, esta não atua na parte do cliente (*web browser*), mas sim no servidor. Tendo em consideração uma vez

mais a Imagem 2, esta é uma ferramenta que possibilita a programação do *web server*, para que seja possível fazer os pedidos ao servidor aplicacional da informação pretendida, e tratar esta se necessário para que seja enviada para o cliente e apresentada ao utilizador.

Esta *framework* utiliza como linguagem de programação base o *Python*, que é uma linguagem de programação considerada das mais fáceis e de rápida aprendizagem (Pyhton Software Foundation, 2015).

Toda a documentação sobre esta *framework*, as suas normas de utilização e as suas inúmeras funcionalidades podem ser lidas no seguinte link:

https://www.djangoproject.com/

3.5. Testes orientados para aplicações *web*

Como já foi referido anteriormente, faz parte dos objetivos desta investigação não só desenvolver uma aplicação *web* mas também levar a cabo testes para verificar se esta cumpre os requisitos que foram inicialmente estipulados.

Para tal, é importante efetuar uma abordagem dos diferentes tipos de testes que existem, mais propriamente os que são orientados para este tipo de aplicações/plataformas *web.*

Palani (2011) refere que existem seis tipos diferentes de testes orientados para aplicações *web,* sendo eles: *Usability testing*, *User Acceptance testing*, *Performance testing*, *Security testing, Functional testing* e *Interface testing*.

3.5.1. Usability testing

Segundo o autor, testes de usabilidade são testes que pretendem averiguar se uma aplicação tem as características que tornam a navegação e interação por parte do utilizador mais cómoda e fácil: "*Usability testing plays a pivotal role with applications that are designed to make manual tasks easier*" (Palani, 2011). Importa consisderar os testes de usabilidade pois podem ser úteis no caso desta investigação, para averiguar se a estrutura da aplicação *web* é a mais correta e a

que exige menos esforço por parte dos utilizadores para efetuar as ações principais.

O mesmo autor refere ainda algumas *guidelines* a seguir aquando da realização destes testes: assegurar a existência de uma mapa do *site*; uso apropriado da cor e tamanhos; ter em conta as diferenças entre utilizadores (culturais, de idade, género); assegurar uma navegação apropriada entre páginas; evitar sobrecarregar as páginas com conteúdos; assegurar suporte para utilizadores com necessidades especiais.

3.5.2. User acceptance testing

Este tipo de testes procuram saber se a aplicação vai ao encontro das expectativas do utilizador (Palani, 2011). Isto é, uma aplicação deve estar concebida com rigor tal que permita que seja realmente "utilizável" e que permita desempenhar ações de forma eficaz e eficiente. O autor indica as seguintes *guidelines* para este tipo de testes: assegurar que a aplicação é compatível com os diferentes clientes (*browsers*); assegurar, nos formulários, que os campos obrigatórios e limites mínimos/máximos são devidamente indicados ao utilizador e que haja *feedback* no caso destes não serem respeitados; ainda nos formulários, utilizar controlos específicos para a ação que se pretende (por exemplo, se num campo do formulário é pedido o género do utilizador, então o mais indicado é apresentar uma caixa de escolha múltipla em vez de um campo de texto, não só para facilitar ao utilizador a escolha do mesmo, mas também para uniformizar as escolhas, uma vez que as palavras "masculino" e "feminino" podem ser escritas erradamente por engano).

3.5.3. Performance testing

Tal como o nome indica, os testes de performance são testes que verificam a performance da aplicação em diferentes cenários. O autor divide estes testes em 3 tipos:

3.5.3.1. *Scability testing*

Estes testes ajudam a perceber o quão adaptável é a aplicação a mudanças de *hardware* e de *software,* quer seja do cliente ou do servidor. O caso mais óbvio onde estes testes são uma mais valia são sem dúvida os testes *corss-browser* que ajudam a perceber se a aplicação tem o mesmo desempenho nos diferentes clientes disponíveis no mercado.

Com a existência de um número considerável de *browsers* e tendo em conta as suas percentagens de utilização atual por parte dos utilizadores nível mundial, é também importante verificar o desempenho do produto desta investigação nos diferentes clientes, desde a versão mais antiga à mais recente, para se garantir que todos os utilizadores tiram máximo proveito da aplicação, independentemente da escolha do *browser.*

3.5.3.2. *Load testing*

Por fim, este tipo de testes tem como objetivo verificar o comportamento da aplicação quando esta está sujeita, por exemplo, a cargas elevadas de utilizadores em simultâneo, ou quando há um fluxo de transferência de ficheiros (*upload* e *download)* entre o servidor e os clientes, acima do normal.

3.5.3.3. *Stress testing*

Muito idênticos ao *"Load Testing*", estes testes pretendem verificar como a aplicação se comporta em ambientes de elevada utilização.

Estes três tipos de testes (*scability, load* e *stress tests*) são também eles muito importantes no caso desta investigação. Como o objetivo deste trabalho é desenvolver uma aplicação que permita verificar, em tempo real, o estado de uma cidade, parte-se do principio que irá haver uma constante transferência de informação entre o servidor e o cliente (*browser*) pelo que haverá necessidade de apresentar esta informação através de formas bastante visuais e apelativas como gráficos e tabelas interativos. É então interessante verificar como estas diferentes componentes se comportam em cenários deste tipo.

3.5.4. Security testing

Testes de segurança são importantes sobretudo quando está em causa informação privada e confidencial que não pode ser exposta ao público. Por exemplo, no caso das instituições bancárias, as suas plataformas devem passar por testes de segurança intensivos e redobrados para assegurar que a informação de cada utilizador se mantém privada. Os testes de segurança têm assim um papel fulcral para verificar os mecanismos de autenticação e proteção da informação numa plataforma *web*.

No caso desta investigação, tratando-se de uma plataforma que pretende disponibilizar informação dos indicadores de determinada cidade, há que ter em conta que estes dados são sensíveis e não podem estar expostos ao público geral. Logo, os testes de segurança poderão ter um papel fulcral para verificar se a informação está realmente segura e está disponível apenas para os utilizadores que devem ter acesso a esta.

3.5.5. Functional testing

Neste tipo de testes, segundo o autor, é pretendido testar funcionalidades individuais da plataforma. Uma plataforma *web* poderá ser subdividida em diferentes módulos ou funcionalidades e os testes funcionais pretendem testar estes módulos um a um, para se ter a certeza que determinado componente está de facto totalmente operacional. Alguns exemplos de testes funcionais são: testes de base de dados; testes de configuração; testes de compatibilidade; testes de fluxo (Palani, 2011).

3.5.6. Interface testing

Por fim, os testes de interface são o sexto tipo de testes referidos pelo autor, que descreve estes testes como sendo os indicados para verificar a coerência e a interligação entre os diferentes “módulos” de uma aplicação *web*, para que a sua utilização seja mais fácil e compreensível. Como já foi referido anteriormente, uma aplicação *web* tem diversas funcionalidades e estas devem estar de alguma forma interligadas entre si de maneira a criar um fluxo de utilização para o utilizador.

Os “testes de interface” são então importantes para verificar se a aplicação cumpre estes requisitos, de maneira a simplificar a sua utilização, e devem ser feitos em qualquer aplicação *web*, sobretudo nas mais complexas, com elevado número de funcionalidades e diferentes “módulos” interligados.

3.6. Aplicações para cidades inteligentes

Neste capítulo analisam-se plataformas que tenham aspetos em comum com a plataforma a ser desenvolvida, destacando as suas funcionalidades, principalmente as que mais se aproximam das funcionalidades pretendidas para o produto desta investigação.

3.6.1. Xively by LogMeIn

Xively é uma divisão da empresa *LogMeIn Inc.*, e tem como missão transformar negócios e pessoas através da *IoT* (Xively, 2015a).

Um dos grandes problemas na concepção de plataformas baseadas na *IoT* é a diversidade de tipos de sensores, atuadores e de dados recolhidos através destes. Assim é necessário estabelecer um conjunto de protocolos e de modelos que permitam, de certa forma, uniformizar esses dados para que sejam consumidos de maneira igual pelos diversos serviços. A *Xively* oferece uma solução integrada que pretende resolver precisamente esse problema, oferecendo um conjunto de ferramentas e serviços, especialmente o chamado “*Developer Workbench”* (Imagem 5), que é um painel de controlo on-line onde é possível visualizar, gerir e utilizar os dados recolhidos da *IoT* para o desenvolvimento de novas aplicações e que se torna o mais importante a reter deste produto pois contém várias funcionalidades em comum com a plataforma que se pretende desenvolver.

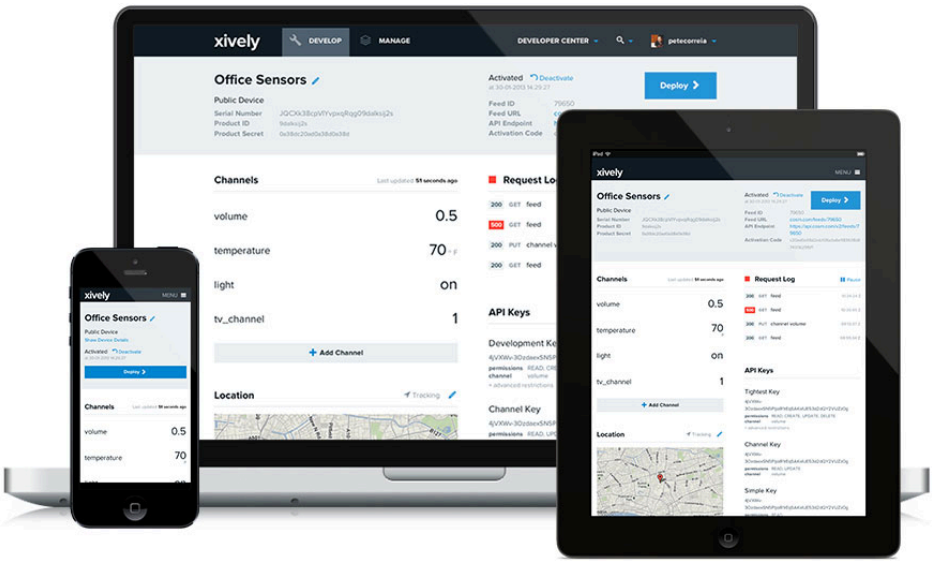


Imagem 5 | "*Developer Workbench"* da Xively

O estudo desta plataforma é também importante pois a *Xively* foi eleita a "*Best Enabling Technology"*, em 2014 e em 2013, na conferencia *M2M Battle of Platforms*, realizada em Las Vegas. (Xively, 2015b)

3.6.2. Omega Management Suite by RacoWireless

À semelhança da *Xively*, a RacoWireless é uma empresa que se foca no mercado da *IoT*, afirmando conhecê-lo melhor que ninguém. Também vencedor em uma das categorias da *M2M Battle of Platoforms* em 2013 (M2M Evolution, 2014), a *RacoWireless* apresenta uma solução bastante idêntica à que se pretende desenvolve, devido às funcionalidades que apresenta que permitem chegar a resultados idênticos, ou seja, a gestão de cidades inteligentes, tornando-se assim em mais um caso importante que vale a pena estudar.

O OMS (*Omega Management Suite)* é uma ferramenta on-line que possibilita controlar e visualizar de forma customizada os dispositivos ("coisas") que estão interligados na *IoT* em questão (Imagem 6). A *RacoWireless* garante que esta plataforma oferece uma experiência de utilização superior à da concorrência, utilizando as melhores e mais avançadas tecnologias *web* da atualidade, estando ainda preparada para ser utilizada em *tablets* e *smartphones*.

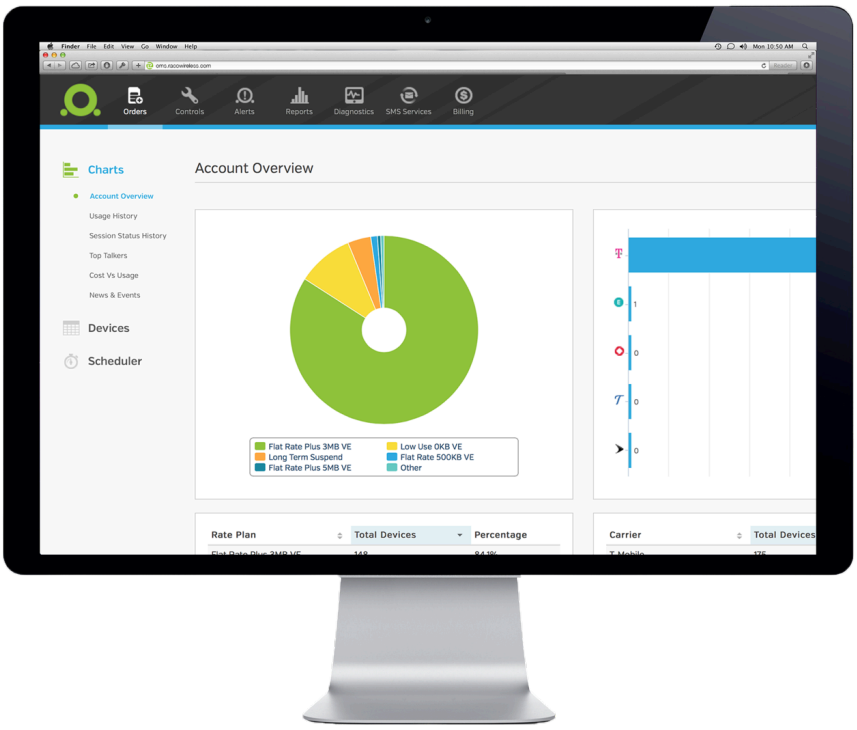


Imagem 6 | Um dos ecrãs da plataforma *"Omega Management Suite"*

Entre as funcionalidades principais desta plataforma e que vão ao encontro das funcionalidades que também se pretendem implementar no objeto de estudo desta dissertação, estão:

- Alertas de utilização: o utilizador é notificado na plataforma e via email quando um limite ("*threshold*") é atingido;
- Controlo e gestão de utilizadores: é possível gerir (criar, remover, editar) utilizadores, bem como as suas permissões, para conseguirem aceder a mais ou menos conteúdo da plataforma; (RacoWireless, 2014)

3.6.3. Freeboard

A *Freeboard* é uma plataforma que permite construir *dashboards* (Imagem 7) e visualizações de informações em apenas alguns minutos, utilizando um mecanismo de *drag & drop* na sua interface (Freeboard, 2014).

Trata-se de uma plataforma que permite ao utilizador criar uma conta pessoal e começar a monitorizar os seus dispositivos com grande facilidade e de forma muito intuitiva, porém, bastante limitada nas funcionalidades e tipos de fontes de

dados que o utilizador pode adicionar para visualizar os dados dos seus dispositivos. Contudo, este conceito de criação de *dashboards* altamente customizados pelo utilizador é uma funcionalidade que se pretende implementar no produto desta dissertação pelo que se justifica o destaque ao *Freeboard*.

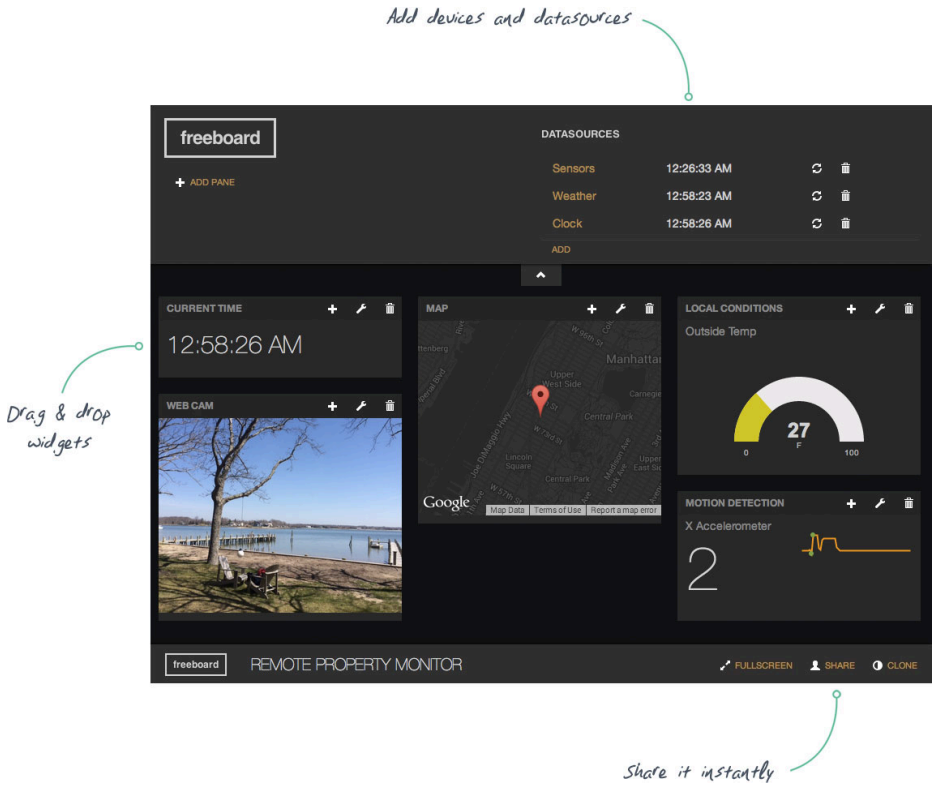
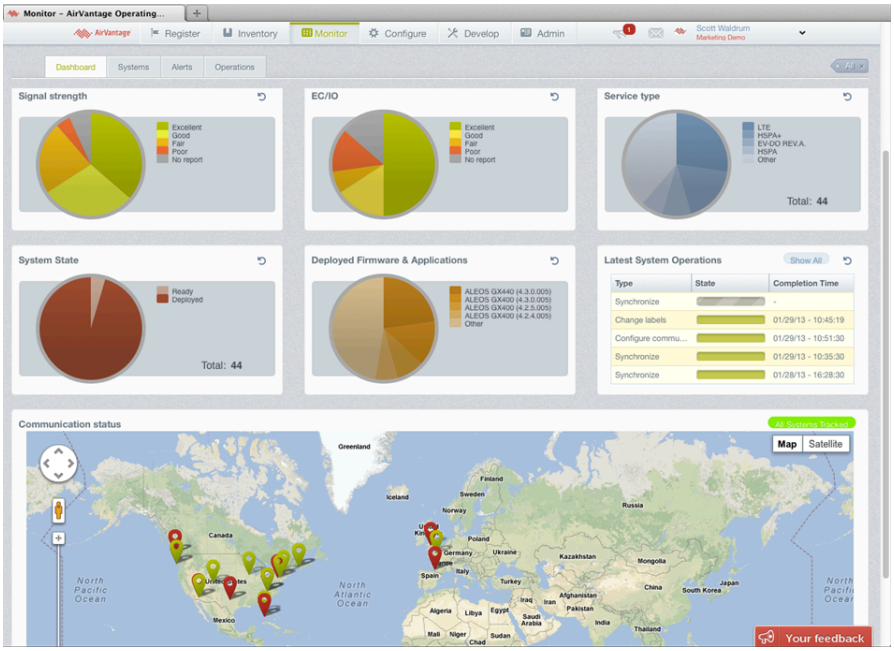


Imagem 7 | Exemplo de um *dashboard* na plataforma "*Freeboard*"

3.6.4. AirVantage Management System by Sierra Wireless

O *AirVantage Management System* (Imagem 8) é um serviço de manutenção de dispositivos que permite controlar e gerir qualquer número de dispositivos *wireless* remotamente (Sierra Wireless, 2014).

É uma plataforma bastante idêntica à oferecida pela *Xively* e pela *RacoWireless* anteriormente referidas, que se foca em disponibilizar ferramentas que possibilitam a interligação de diferentes dispositivos, de diferentes características, e oferece um painel de controlo para controlar todas as suas atividades em tempo real.



| 8 | Um dos ecrãs da plataforma "*AirVantage Management Service*"

Um dos aspetos mais importantes a reter desta plataforma é a capacidade que tem em assumir todos os dispositivos como sendo iguais, ou seja, as ações que se podem tomar com qualquer dispositivo são quase todas as mesmas, pois estes são uniformizados, quer se trate de um dispositivo GPS que debita a localização de um objeto qualquer, ou de um sensor de luz que mede a luminosidade numa rua de uma cidade. Isto é possível também através da criação de "*templates*" que se atribuem a todos os dispositivos idênticos (que sirvam para a mesma finalidade, ou que estejam relacionados de certa forma). Assim, é possível ter uma plataforma com conteúdos completamente diferentes, mas agrupados de forma e com funcionalidades em comum (Sierra Wireless, 2014). Este é um aspeto muito importante numa plataforma como esta, pois como já foi referido em pontos anteriores, os tipos de dados e dispositivos numa *IoT* são muitos e podem ser completamente díspares, logo há que encontrar formas de "*standardizar*" esses aspetos para que a usabilidade da plataforma seja a mais adequada.

Para concluir, este estudo do estado da arte segue-se uma tabela que pretende comparar as funcionalidades base de cada uma das plataformas investigadas, indicando qual delas possuí ou não um conjunto de características estabelecidas e que se julgam ser importantes neste tipo de plataforma (Tabela 2).

		PLATAFORMAS			
		Xively	*OMS*	*Freeboard*	*AirVantage*
FUNCIONALIDADES E CARACTERÍSTICAS	Criação de *dashboards* personalizados	**X**		**X**	**X**
	Manutenção de utilizadores com diferentes	**X**	**X**		**X**
	Interface moderna e apelativa	**X**	**X**	**X**	**X**
	Compatibilidade com dispositivos móveis	**X**	**X**	**X**	
	Oferta de ferramentas de desenvolvimento (*ie. APIs*)	**X**			**X**
	Suporte intensivo ao cliente	**X**	**X**		**X**
	Existência de mecanismos de segurança mais avançados	**X**	**X**		**X**
	Simplicidade e facilidade de utilização			**X**	
	Utilização gratuita			**X**	

Tabela 2 | Comparação de funcionalidades entre as diferentes plataformas

Esta tabela é importante para esta investigação, na medida em que indica de uma forma geral as funcionalidades e as características que a plataforma que se pretende desenvolver deve possuír para se destacar da concorrência e ganhar assim vantagem no mercado. Através de uma análise é possível verificar, por exemplo, que as plataformas analisadas tendem a atingir um nível considerável de complexidade que se reflete numa dificuldade na sua utilização. É portanto importante a plataforma desta investigação adopte processos e funcionalides que facilitem o uso aos utilizadores.

4. Desenvolvimento

Depois de enquadrar teoricamente o objeto em estudo e desenvolvimento desta investigação e de fazer um balanço do estado da arte na área, segue-se o processo de desenvolvimento.

É importante voltar a referir que esta investigação foi realizada em ambiente empresarial e por isso as escolhas dos caminhos a seguir no desenvolvimento da solução foram feitas em parceria com a empresa em questão, que devido à sua experiência em desenvolvimento de aplicações *web* é uma mais-valia não só para a escolha das melhores ferramentas e tecnologias a utilizar para o desenvolvimento como também para conceptualização do produto em si.

4.1. Conceito

O "*control center*", nome pelo qual o produto desta investigação será designado daqui em diante, pretende apoiar a gestão e o controlo de uma cidade inteligente, como já foi referido anteriormente, sendo agora importante referir detalhadamente como irá ser possível através de uma plataforma *web* controlar uma cidade.

Este produto é apenas mais um no meio de vários já desenvolvidos e que fazem parte do plano de negócio da empresa com a qual esta investigação está a ser desenvolvida: a empresa dispõe de aplicações que permitem o controlo de aspetos mais específicos de uma cidade, tal como uma aplicação a gestão dos lugares e parques de estacionamento e outra para a gestão da recolha do lixo.

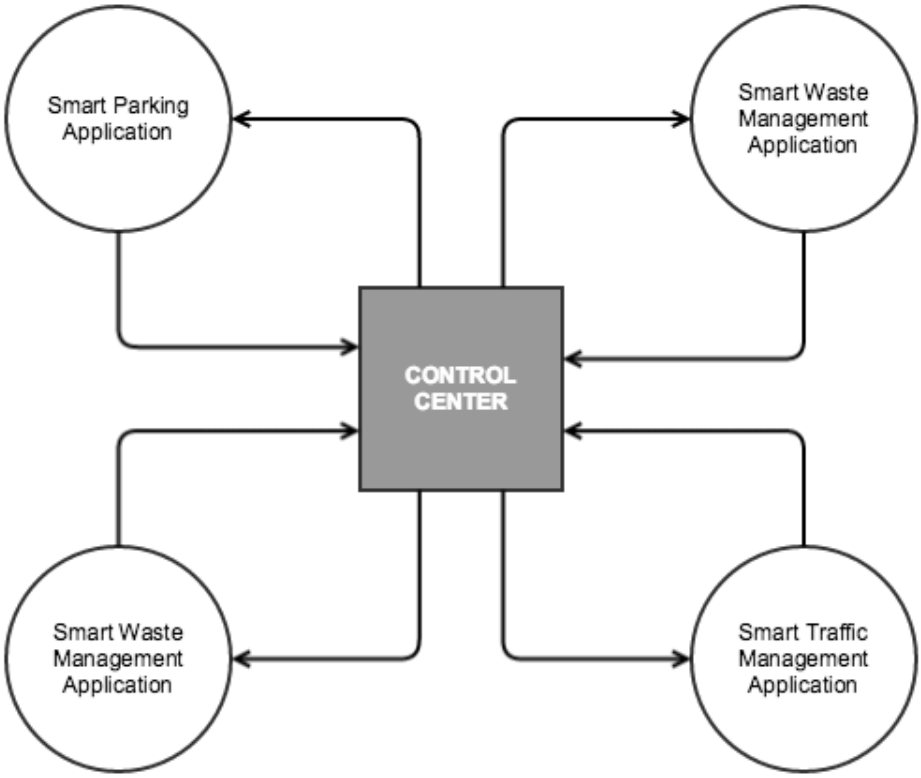


Imagem 9 | Esquema demonstrativo da relação entre o "*control center*" e as restantes aplicações

O *"control center"* é portanto uma plataforma que pretende unir todas estas aplicações individuais já existentes que tratam aspetos muito específicos de uma cidade, e utilizar os dados recolhidos por estas para permitir ao utilizador cruzar informações e ter uma visão global de toda a cidade, num único *website*. A Imagem 9 representa a ideia fundamental da plataforma trabalhada nesta investigação e para a qual a empresa conta com o investigador para desenvolver uma arquitetura de informação e os mecanismos necessários para permitir uma interação fácil, eficaz bem como uma facilitada visualização da informação.

Foi possível verificar anteriormente, principalmente no capítulo das aplicações para cidades inteligentes, que estas tendem a ser complexas e por vezes confusas para o utilizador, devido, essencialmente, à heterogeneidade dos sensores e tipos de dados possíveis numa "cidade inteligente". Um dos maiores desafios desta investigação é contrariar esta complexidade associada às plataformas de gestão de cidades inteligentes, ou aplicações genéricas

direcionadas à *IoT*, e idealizar uma plataforma unificada que trate os diferentes tipos de dados, recolhidos pelos diferentes tipos de sensores, de uma forma o mais uniforme possível. Será expectável que este processo de uniformização de processos resulte em duas grandes vantagens: a) A primeira vantagem seria um aumento da facilidade de utilização da plataforma, pois independentemente do tipo de informação que o utilizador pretende interagir, os processos serão os mesmos e consequentemente a facilidade de utilização da plataforma será maior; b) A segunda vantagem será a obtenção de uma maior escalabilidade no sistema em geral, o que significa que a aplicação estará melhor preparada para receber novos tipos de dados, e para desenvolver novas funcionalidades, independentemente do tipo de dados que se está a tratar: "*The key characteristic of a scalable application is that additional load only requires additional resources rather than extensive modification of the application itself."* (Microsoft Developer Network, 2015). Numa aplicação para a *IoT,* como a que esta investigação pretende estudar e desenvolver, a escalabilidade é uma das características mais importantes, pois a aplicação deve estar preparada para receber qualquer tipo de dados sem qualquer alteração no sistema.

4.2. Arquitetura funcional plataforma

Para resolver o principal problema das plataformas orientadas para a *IoT* e concretamente para cidades inteligentes, que foi descrito anteriormente, é necessário estabelecer um modelo de informação e uma arquitetura de sistema que permita uma interação fácil, mas ao mesmo tempo eficaz e que não limite em nenhum aspeto o tipo a informação que é possível visualizar sobre a cidade inteligente. Na Imagem 10 está representada a arquitetura da plataforma "*control center*" desenvolvida nesta investigação e de que forma se relacionam todos os seus módulos.

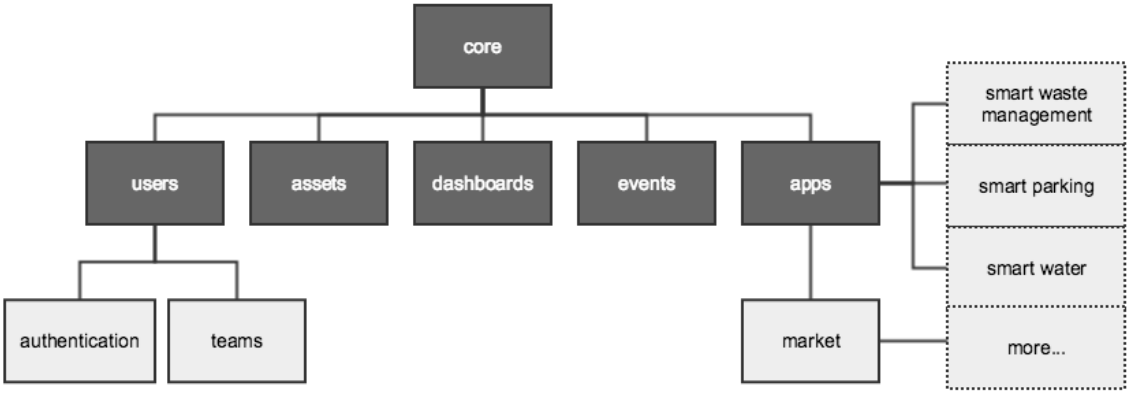


Imagem 10 | Esquema dos módulos da plataforma "*control center*"

A arquitetura é dividida em seis módulos principais: *core, users, assets, dashboards, events* e *apps*, sendo estes representados com os quadrados mais escuros na Imagem 10. Pode-se verificar ainda que o módulo de utilizadores é subdividido em dois outros módulos, o módulo de autenticação e o módulo das equipas,, e que o módulo das *apps* permite a relação entre o *"control center*" e as aplicações já desenvolvidas que se focam em aspetos concretos de uma cidade inteligente.

4.2.1. Core

O módulo "*core*" é comum a toda a plataforma e trata dos processos que são de certa forma realizados um pouco por todos os outros módulos. É este módulo que fornece os ficheiros comuns como as imagens, folhas de estilo (CSS) e *templates* que são utilizados pelos restantes módulos e é também onde se procede à ligação à componente de *backend*, através das classes e funções que permitem a ligação à API desenvolvida e fornecida pela empresa com a qual esta investigação foi desenvolvida.

4.2.2. Users

Este módulo possibilita a gestão de utilizadores no *"control center*" e também em cada uma das aplicações individuais associadas à plataforma. Isto é, o módulo *"users*" fornece as ferramentas necessárias para criar, editar e até remover utilizadores do próprio *"control center*" e também das aplicações associadas à

plataforma, para assegurar uma gestão centralizada dos utilizadores. Este módulo é também responsável pela gestão das permissões, assegurando a possibilidade de criar utilizadores com permissões diferentes para terem mais ou menos privilégios. O módulo *"users*" está ainda subdividido no módulo de autenticação e no módulo das equipas. O módulo da autenticação é responsável por autenticar o utilizador não só quando este tenta aceder à plataforma (*login*) mas também quando tenta fazer qualquer pedido de informação – como já foi referido, o módulo de utilizadores tem adjacente a noção de permissões dado que a segurança e a exposição dos dados é um assunto importante pois há informação sensível que apenas deve estar disponível para certas entidades. O módulo de autenticação pretende resolver estes problemas e assegurar que apenas as pessoas com as permissões adequadas podem ver as informações mais privadas. O módulo "*teams*" permite que haja ainda a possibilidade da criação de equipas às quais são associadas utilizadores e permissões, para que seja mais fácil gerir utilizadores com características idênticas.

4.2.3. Assets

Para tentar resolver o problema descrito anteriormente referente à heterogeneidade da informação possível de recolher numa cidade inteligente, e de forma a não limitar a informação que é possível visualizar no *"control center*" foi criado o módulo *"assets*".

A quantidade de diferentes equipamentos, sensores e dispositivos que se podem instalar numa cidade inteligente para recolher variadas informações não possibilita que haja um tratamento diferenciado para cada tipo de dispositivo. Ao invés disso, o módulo das *"assets*" visa tratar todos estes diferentes componentes como um só, permitindo registar no sistema qualquer equipamento/sensor independentemente da sua função e do tipo de informação que recolhe, sendo que o termo *"asset*" pretende designar qualquer "coisa" da *Internet of Things*. Assim potencia-se a escalabilidade do sistema pois não há nenhuma limitação no tipo de informações e sensores que podem ser contemplados numa cidade inteligente.

Este tratamento uniformizado que se dá a todos os equipamentos que se pretendem contemplar numa cidade inteligente pode trazer alguns problemas devido às enormes diferenças que podem existir entre estes. Assim pretende-se que este módulo ofereça as ferramentas necessárias para ultrapassar estes problemas, como por exemplo a interface de criação de *assets* que permite que sejam adicionados múltiplos atributos customizados aquando do registo da *asset* no sistema. Para além da interface e de criação, o módulo das "*assets*" permite também ter uma visão global sobre o estado de todos os equipamentos em tempo real, editar cada um desses equipamentos e até removê-los do sistema.

4.2.4. Dashboards

O módulo *"dashboards*" é o módulo que permite a criação dos *dashboards* customizados no "*control center*". Cada *dashboards* pode ser composto por diversos gráficos, tabelas, ou outras formas de visualização (blocos) que o utilizador pretenda. Desta forma, é possível oferecer ao utilizador diferentes formas de visualizar diferentes aspetos da cidade inteligente.

Tal como no módulo das "*assets*", o desafio neste módulo também é superar o problema da quantidade de diferentes equipamentos, sensores, e tipos de informações existentes numa cidade inteligente: é necessária uma ferramenta eficaz e simples mas ao mesmo tempo que não limite o utilizador no que diz respeito ao tipo de visualização e à informação que pode cruzar num *dashboard.*

Para simplificar o processo de criação de um *dashboard* e dos seus "blocos", este módulo fornece uma ferramenta de criação "passo-a-passo", onde o primeiro passo é escolher a forma de visualização de entre uma lista, que pode ser um gráfico de linhas, de barras, um mapa, um medidor de pressão ou uma simples caixa de texto onde irá constar um valor. De seguida é necessário escolher quais parâmetros de quais *"assets*" é que vão ser contemplados nessa forma de visualização. Por fim é apenas necessário dar um titulo e uma descrição ao bloco.

Assim, o utilizador tem a possibilidade de criar múltiplos *dashboards* customizados, cada um com diferentes blocos e diferentes formas de

visualização, que estão a mostrar informação recolhida por diferentes *"assets"* de diferentes aplicações associadas. Por exemplo, é possível ao utilizador criar um gráfico de linhas comparando os níveis de CO^2 no ar com o número de lugares de estacionamento livres ao longo do tempo numa determinada área urbana, ou então um mapa que mostra os vários contentores de lixo que necessitam de ser recolhidos, em tempo real.

4.2.5. Events

O módulo dos *"eventos*" é o módulo mais simples da plataforma, pelo menos no que diz respeito à componente em desenvolvimento (*frontend*). Este módulo, através do módulo *"core*" estabelece uma ligação à componente *backend* e recebe todos os eventos que são emitidos por todos os sensores espalhados pela cidade, servindo para mostrar ao utilizador uma listagem destes eventos em tempo real. Um evento pode ser, por exemplo, um lugar de estacionamento a ficar ocupado ou um contentor de lixo ficar cheio.

4.2.6. Applications

Como já foi referido anteriormente, o conceito deste *"control center*" é ser um painel de controlo onde é possível visualizar a informação que é recolhida em diferentes aplicações que tratam aspetos específicos de uma cidade inteligente, como por exemplo uma aplicação *"smart parking*" para recolher informações relativas aos lugares de estacionamento de um centro urbano, ou a aplicação "*smart waste*" que recolhe informação dos contentores de lixo e indica quando é que cada um necessita de ser recolhido.

O módulo *"apps*" (aplicações) estabelece a relação entre o *"control center*" e estas diversas aplicações individuais, fornecendo uma interface de visualização por cada aplicação onde o utilizador pode ver um *dashboard* com a informação mais geral e básica relativa àquela aplicação – para ver mais em detalhe os conteúdos relativos à aplicação em específico, o utilizador terá de navegar até ao *website* da aplicação, que é fornecido nessa página no *"control center*".

Este módulo tem ainda um associado outro módulo, designado como *"market*" (mercado). Este módulo está profundamente relacionado com o modelo de negocio da empresa com a qual esta investigação é desenvolvida, e serve para fornecer ao utilizador uma interface gráfica onde pode visualizar todas as aplicações disponíveis que pode associar ao seu *"control center*" para que possa controlar outros aspetos da cidade inteligente, funcionado assim como um mercado onde se podem comprar outras aplicações – estas aplicações, para serem adicionadas ao *"control center*", terão de ser registadas de acordo com um processo que assegure que sejam indicados os principais parâmetros que controla e os seus principais objetos. Por exemplo, ao registar no mercado uma aplicação *"smart parking*", esta terá de ser registada indicando os seus principais objetos, que neste caso seriam porventura os *"parking spots*", e todos os parâmetros (atributos customizados) que estes possuem, como a latitude, longitude, etc.

Na Imagem 10 é possível verificar esta relação que o módulo "*apps*" estabelece entre o *"control center*" e as aplicações externas como a aplicação *"smart parking"* ou "*smart waste management*", e também a possibilidade de haver mais aplicações associadas através do mercado.

Concluindo este capítulo, é importante referir que a arquitetura da plataforma foi pensada em conjunto com os responsáveis pelo produto na empresa com a qual esta investigação é desenvolvida. Tratando-se de um assunto que tem de traduzir as ideias e o modelo de negócio da empresa, não poderia ficar completamente ao critério do investigador desenvolver toda a arquitetura do *"control center*".

4.3. *Mockups* da plataforma

Tal como foi referido no capítulo das Finalidades e Objetivos desta investigação, faz parte do trabalho do investigador a prototipagem dos ecrãs do *"control center*". Aqui a missão do investigador é interpretar as *user stories* definidas pela empresa e construir ecrãs que as satisfaçam, servindo futuramente como base para os *designers* da empresa construírem os ecrãs finais da plataforma.

Sendo o *"control center"*, dito de uma forma muito geral, um painel de controlo de uma cidade inteligente, que permite controlar os diversos dispositivos (sensores, etc.), utilizadores, aplicações, *dashboards* e eventos (Imagem 10) relacionados, é expectável que este venha a ter um número considerável de interfaces onde é possível criar, editar e listar cada um destes. Logo, é importante manter todas estas interfaces acessíveis facilmente para não comprometer a usabilidade da plataforma.

Assim, decidiu-se que a plataforma adotaria uma interface tipo *"dashboard"*. Esta interface é característica pelo seu menu lateral sempre presente – este menu reproduz de certa forma a arquitetura da plataforma apresentada na Imagem 10, permitindo um acesso rápida a cada um dos componentes, ideal para plataformas que necessitem de mostrar uma quantidade relativamente grande de informação: "a well-designed dashboard can save huge amounts of time, helping people to quickly identify the numbers that matter, in order to make insightful observations or to compile reports." (Lake, 2013)

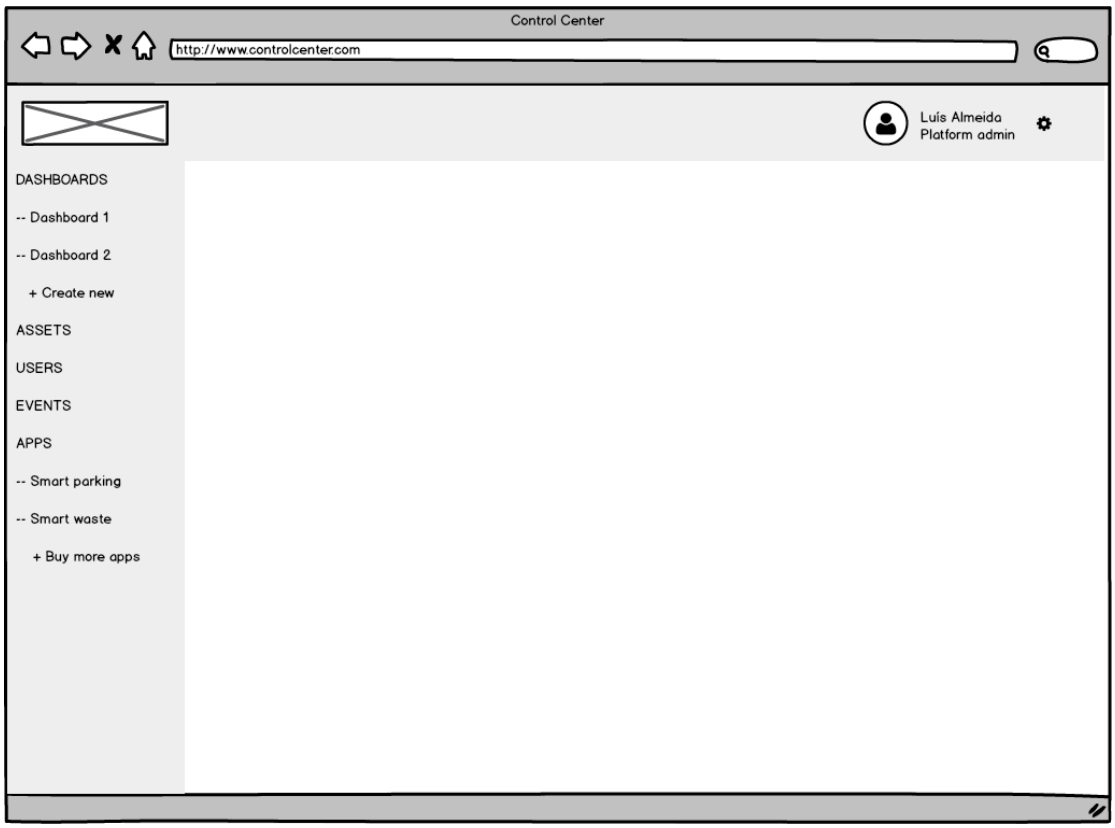


Imagem 11 | Mockup da interface base do "*control center*"

Esta interface simplifica a navegação ao utilizador, pois a barra do lado esquerdo está sempre visível e sempre igual independentemente da página que esteja ativa, e reproduz a arquitetura da plataforma dando acesso às principais áreas em poucos cliques. Para além disso, esta barra lateral transmite ainda a ideia de hierarquia de conteúdos, pois o botão de *Dashboards* expande para mostrar todos os *dashboards* que o utilizador tem criados, e ainda um outro botão para criar um novo *dashboard.* O mesmo acontece com o botão das *Apps,* onde é possível expandir para ver todas as aplicações associadas ou ainda ir para o *market* para associar novas aplicações.

Ao clicar em cada um dos botões da barra lateral de menus, o conteúdo respetivo irá aparecer no centro da interface, na área em branco da Imagem 11. É importante referir ainda a barra no topo da interface base, que irá conter a informação relativa ao nome do utilizador, e o nome da permissão que lhe foi dada aquando da sua criação, como foi explicado no capítulo da arquitetura da plataforma, no módulo dos *Users.*

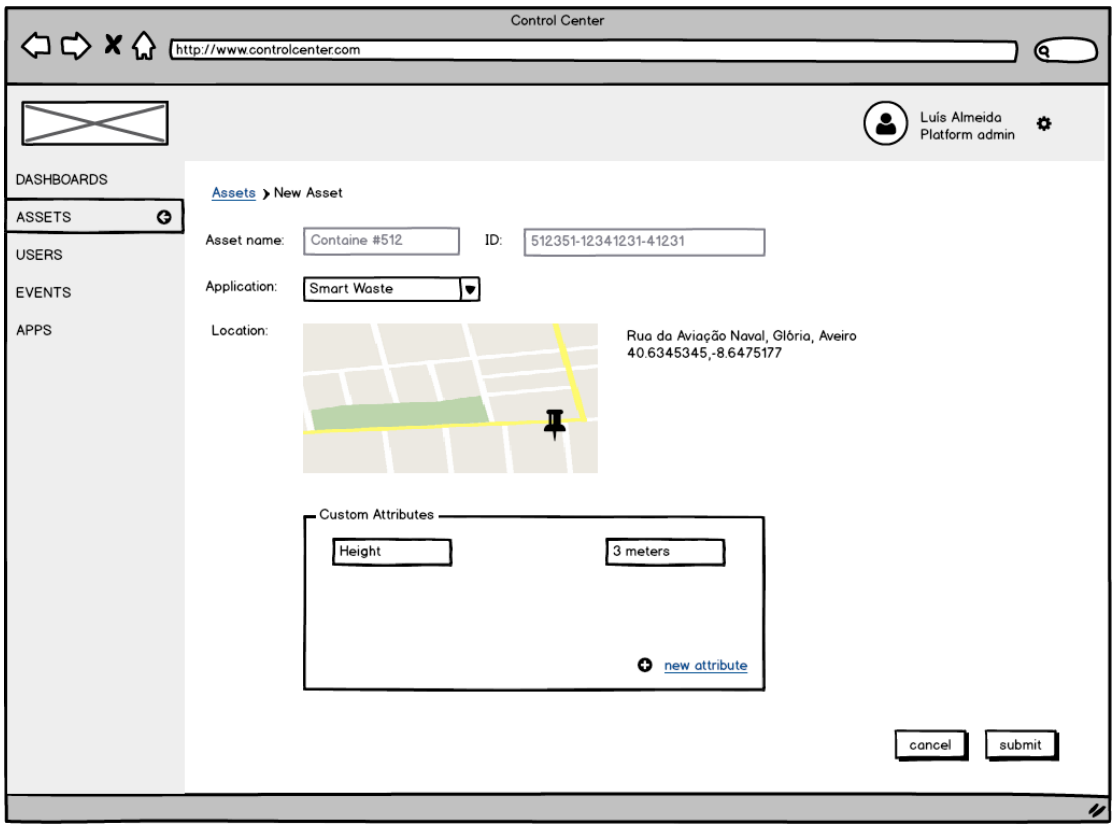


Imagem 12 | *Mockup* da interface de criação de "*assets*"

Na Imagem 12 está representado o mockup da interface que permite ao utilizador registar *"assets*" no *"control center*" e também nas respetivas aplicações. Tal como foi referido anteriormente, o módulo das *"assets*" visa criar qualquer "coisa*" na IoT* , e como tal, é importante não limitar o tipo de dispositivo que se pode adicionar através desta interface. A principal preocupação neste ecrã foi a possibilidade de adicionar atributos customizados, pois como foi referido anteriormente na explicação do módulo "*assets*" na arquitetura da plataforma, as diferenças que existem no tipo de dispositivos que uma cidade inteligente pode ter pode trazer problemas na criação de interfaces que possibilitem ao utilizador o registo destes no sistema. Como é possível verificar na Imagem 12, esta interface permite *dar* um nome à "*asset*", especificar o ID pelo qual vai ser reconhecida na componente *backend,* especificar a sua localização no centro urbano, e também atribuir os "atributos customizados", que podem ser tantos quantos o utilizador pretenda. É importante referir também a obrigatoriedade de especificar a qual das aplicações é que esta *"asset*" pertence. Mais uma vez, o *"control center*" é apenas um painel de controlo que unifica as restantes aplicações individuais já desenvolvidas e, portanto, ao criar uma *"asset*" no *"control center*" o que se está na verdade a fazer é a criar uma *"asset*" numa das aplicações. É por isso necessário especificar qual aplicação a que pertence – neste caso, através de uma lista de aplicações disponíveis, o utilizador apenas terá de selecionar aquela para qual a *"asset*" vai se associada.

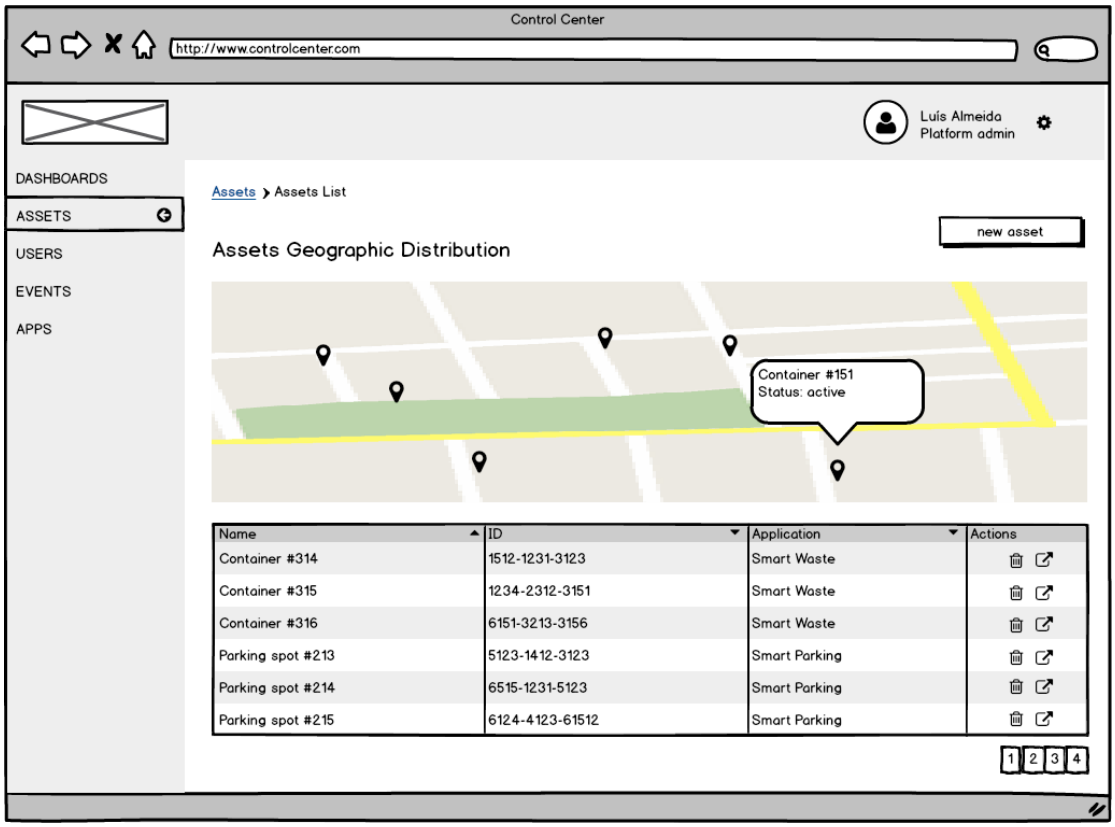


Imagem 13 | *Mockup* da interface de listagem de *assets*

Para ter uma vista geral de todas as *assets,* foi também desenvolvido um *mockup* de listagem, representado na Imagem 13. Neste *mockup* evidencia-se o mapa que representa a distribuição geográfica de todas as *assets* registadas nas diferentes aplicações, na qual o utilizador pode clicar nos diferentes pontos e ver de que *asset* se trata e o seu estado. É também possível através desta interface rapidamente ver a *asset* em detalhe na sua própria página e até removê-la do sistema.

Um dos objetivos estabelecidos pela empresa seria tornar esta interface o mais simples e uniforme possível, portanto todos os ecrãs de criação, quer seja de *assets* ou de utilizadores, são idênticos aos ecrãs de edição e de detalhe, sendo que nos ecrãs de edição a informação relativa ao objeto que se está a tratar está desde logo preenchida em cada um dos respetivos campos.

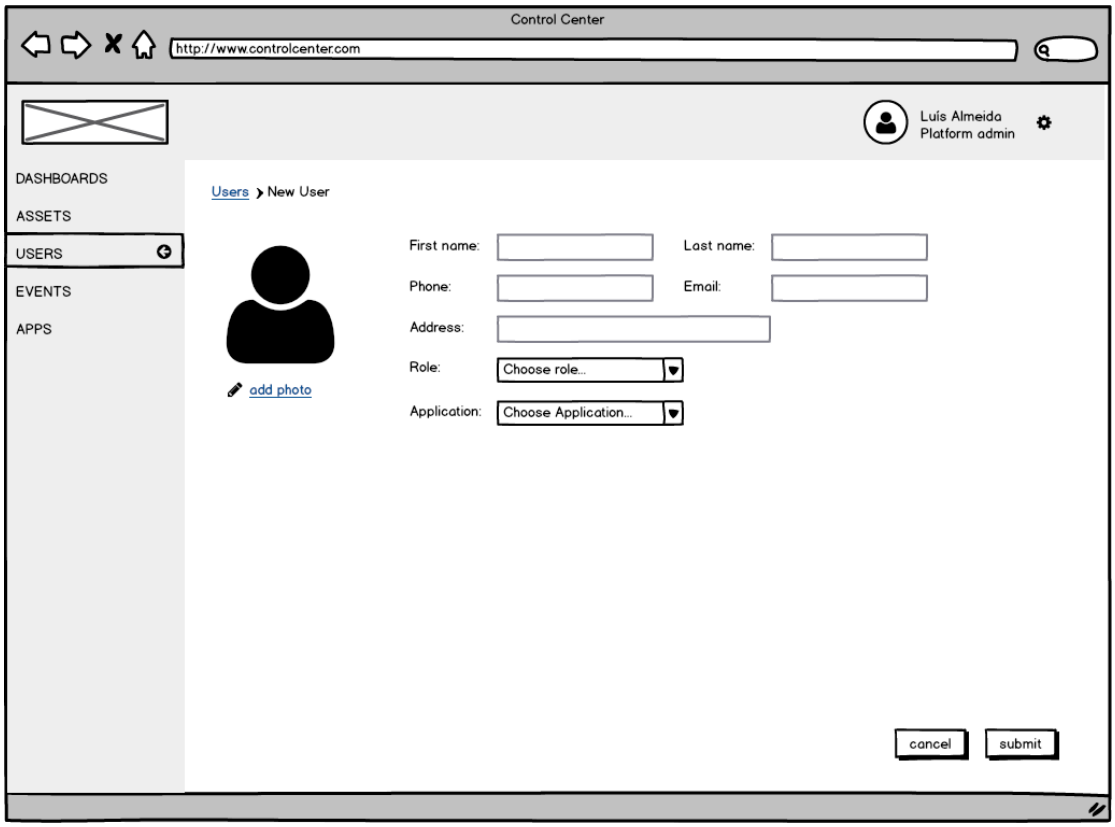


Imagem 14 | *Mockup* da interface de criação de utilizadores

À semelhança do módulo *"assets"*, também o módulo dos utilizadores necessita de fornecer uma forma de adicionar utilizadores, não só no próprio *"control center"* como também a cada uma das aplicações associadas.

Na imagem 14 é possível verificar o *mockup* de criação de utilizadores do *"control center"*: nesta interface é necessário especificar o primeiro e último nome do utilizador, o seu número de telefone, o email, a morada, a permissão/*role*, e a aplicação a qual pertence. Como já foi referido anteriormente na explicação do módulo dos utilizadores no capítulo da arquitetura da plataforma, este módulo permite a gestão de utilizadores e das permissões associadas para que se consigam dar mais ou menos privilégios dentro da plataforma. Ao indicar o *role* do utilizador aquando da sua criação, atribui-se uma permissão especifica que pode dar mais ou menos acessos a conteúdos mais sensíveis que se queiram proteger. Tal como na criação das *"assets"* é crucial indicar com qual das aplicações é que o utilizador é associado.

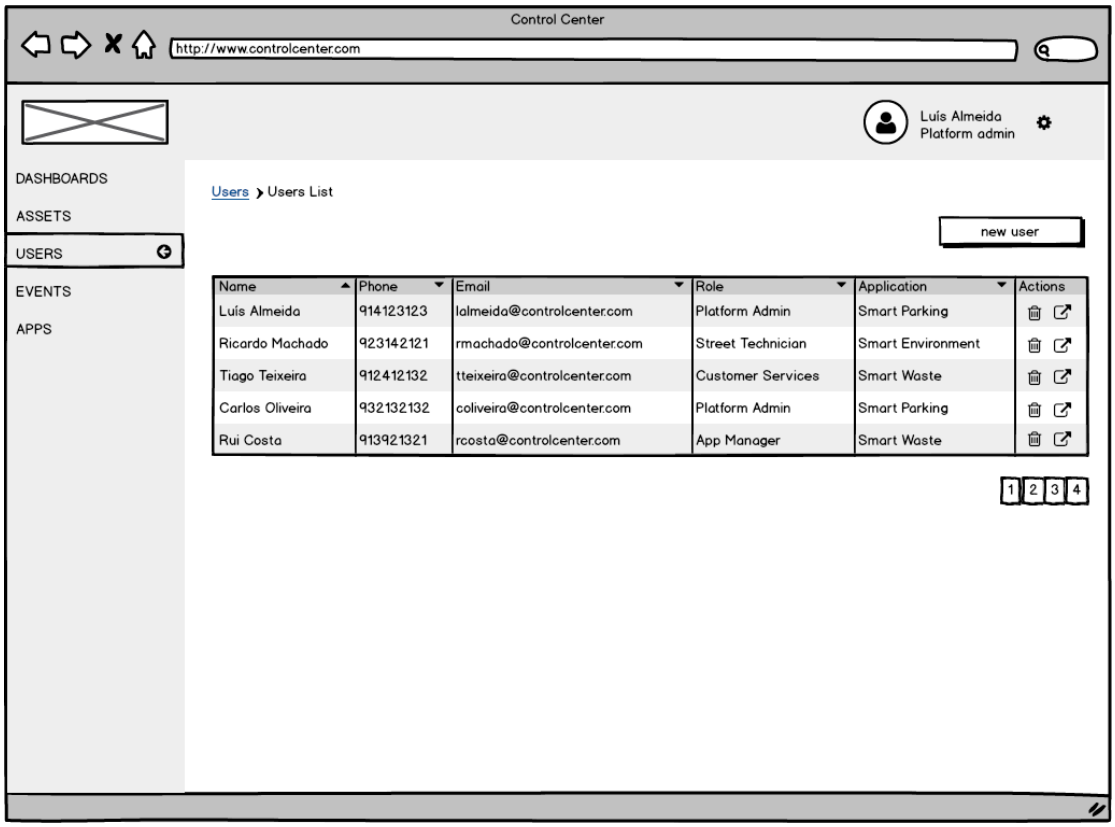


Imagem 15 | *Mockup* da interface de listagem de utilizadores

É necessária também uma interface onde seja possível listar todos os utilizadores e de forma fácil ver as suas informações principais. Na Imagem 15 é possível verificar o *mockup* que dá resposta a esta necessidade e que permite ao utilizador ver todos os outros utilizadores registados nas diferentes aplicações, através de uma listagem simples e que permite ao mesmo tempo ter ações rápidas como apagar um determinado utilizador ou ir para a página de detalhe do mesmo.

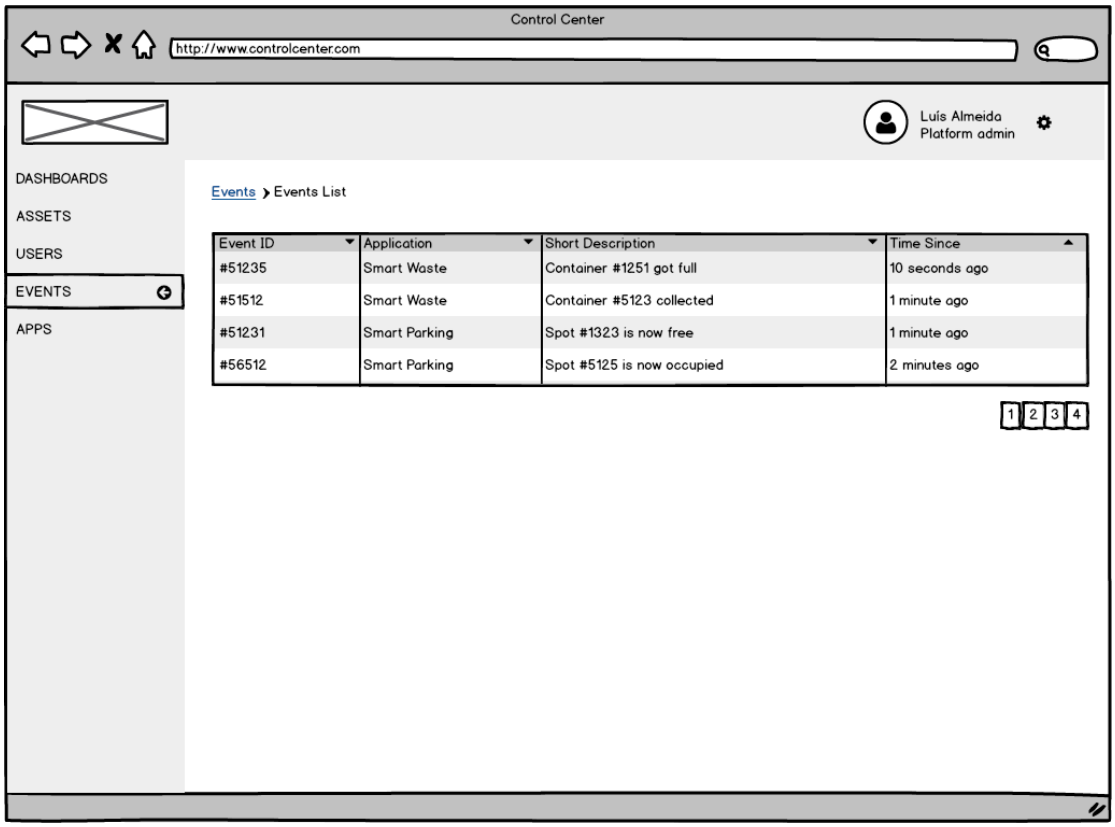


Imagem 16 | *Mockup* da interface de visualização dos eventos

A visualização dos eventos é a funcionalidade que exige uma interface mais simples, pois não há quaisquer ações associadas, tratando-se de uma listagem simples apenas a indicar o ID do evento, a aplicação associada, uma pequena descrição e há quanto tempo se sucedeu. Esta mesma interface está representada na Imagem 16 e é uma listagem idêntica à de utilizadores e à das *"assets*".

No que diz respeito ao módulo dos *dashboards*, o grande desafio é, como referido anteriormente, a ferramenta de criação das formas de visualização (blocos).

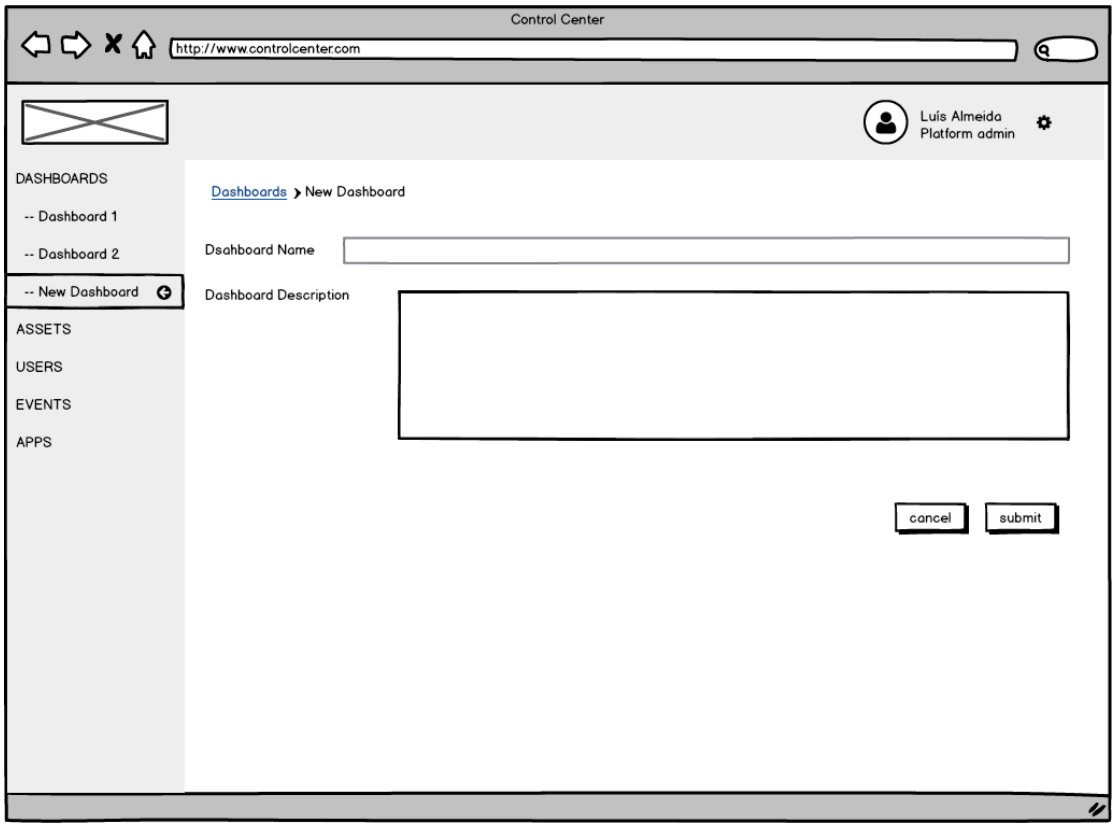


Imagem 17 | *Mockup* da interface de criação de *dashboards*

Em primeiro lugar o utilizador tem de criar um *dashboard* onde vão estar as diferentes formas de visualização, ou blocos. Ao clicar em *Dashboards* na barra lateral da interface, o menu dos *dashboards* expande-se e apresenta um botão por cada *dashboard* criado e um outro para criar um novo *dashboard,* como está representado na imagem 17.

Ao clicar em "*New Dashboard*" o utilizador acede à interface de criação que é muito simples necessitando apenas de introduzir um nome e uma descrição.

Imagem 18 | *Mockup* do estado de um *dashboard* sem qualquer bloco

Depois de criar o *dashboard*, o utilizador terá então acesso à página do mesmo através do menu lateral, contudo, este ainda não tem quaisquer blocos criados (Imagem 18), e terá de proceder à criação destes através da ferramenta de criação passo-a-passo explicada anteriormente no capitulo da arquitetura da plataforma.

Ao clicar no botão *"new block"* o utilizador pode acede à interface de criação de blocos (Imagem 19). Esta interface abre em forma de modal para não afastar o utilizador do contexto do *dashboard,* ou seja, o ecrã do *dashboard* continua sempre visível por detrás desta nova interface.

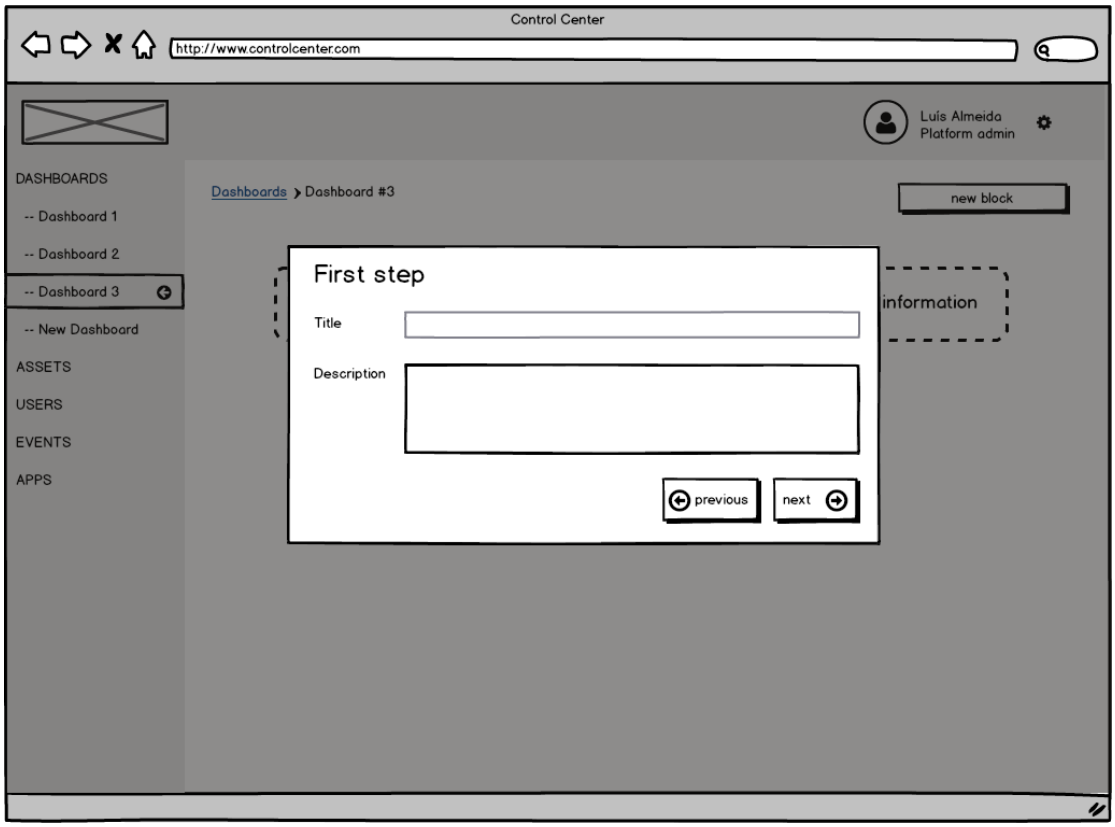


Imagem 19 | *Mockup* da interface do primeiro passo da ferramenta de criação de blocos para os *dashboards*

Pode-se ainda verificar, na Imagem 19, que os botões *previous* e *next* servem para o utilizador navegar para o próximo passo da ferramenta de criação de blocos ou recuar para o anterior.

No seguinte passo (Imagem 20) o utilizador depara-se com a interface de escolha da forma de visualização a partir de um conjunto de opções. Estas opções são pré-definidas e necessitam de desenvolvimento sempre que seja desejado acrescentar uma nova forma de visualização a este painel de escolhas uma vez que cada uma necessita de dados diferentes, para ser possível a sua construção.

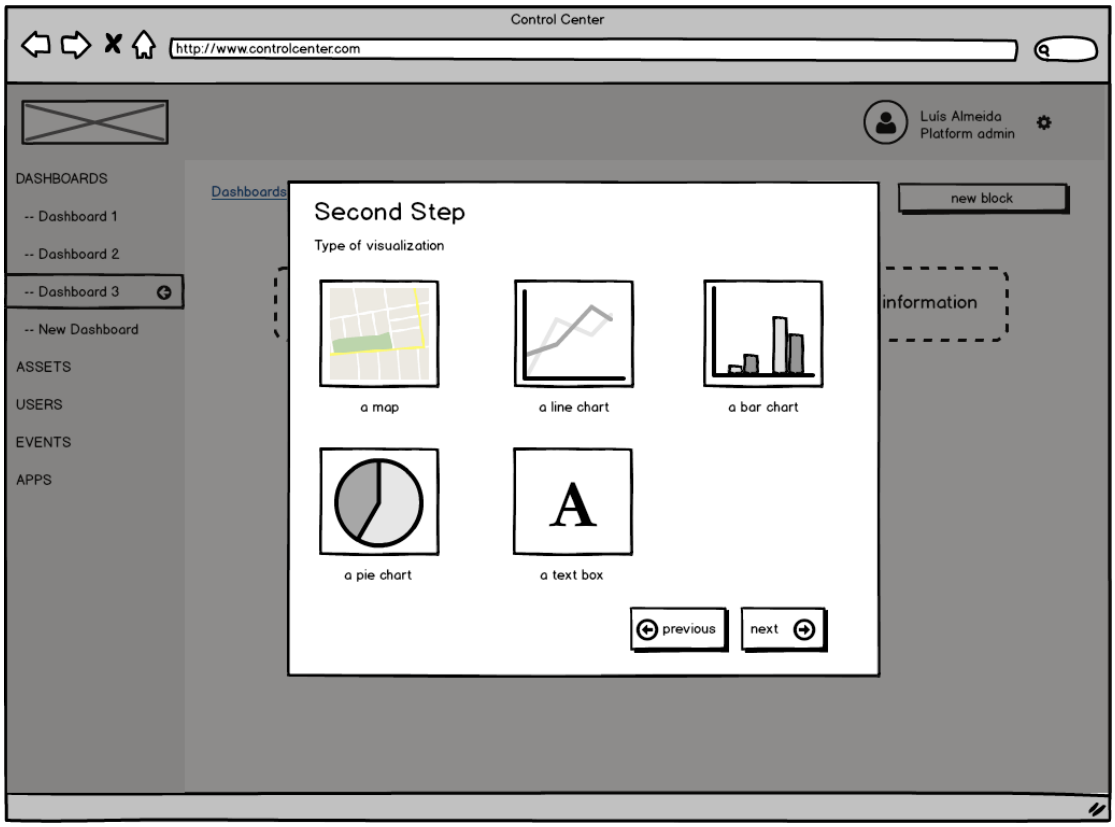


Imagem 20 | *Mockup* da interface do segundo passo da ferramenta de criação de blocos para os *dashboards*

Por exemplo, um mapa necessita de dois conjuntos de valores para a latitude e longitude dos pontos, enquanto que um gráfico de linhas necessita de um conjunto de dados mais complexo, e portanto o passo seguinte (Imagem 21) é diferente para cada uma das formas de visualização disponíveis. Contudo, a forma de seleção dos valores a serem contemplados é idêntica para todos os casos.

Como é possível verificar na imagem 21 (abaixo), o terceiro passo, que, tal como já foi referido, irá ser diferente consoante a forma de visualização escolhida, permite ao utilizador selecionar os valores em 3 passos. Primeiro é necessário escolher a aplicação em que o valor se encontra. Depois de escolher a aplicação, a caixa seguinte (do meio), irá apresentar os serviços registados para esta aplicação. Como foi referido anteriormente na explicação do módulo das *Applications* e do *Market*, para cada aplicação que é registada no *"control center"* é necessária a especificação dos principais objetos e dos seus atributos, sendo

aqui designados como serviços. Por exemplo, ao selecionar na primeira caixa a aplicação *smart waste,* aparece entre os serviços disponíveis os *containers*, e por fim, na terceira caixa, estarão os atributos disponíveis deste objeto específico. Neste caso, interessa selecionar as latitudes e longitudes dos mesmos.

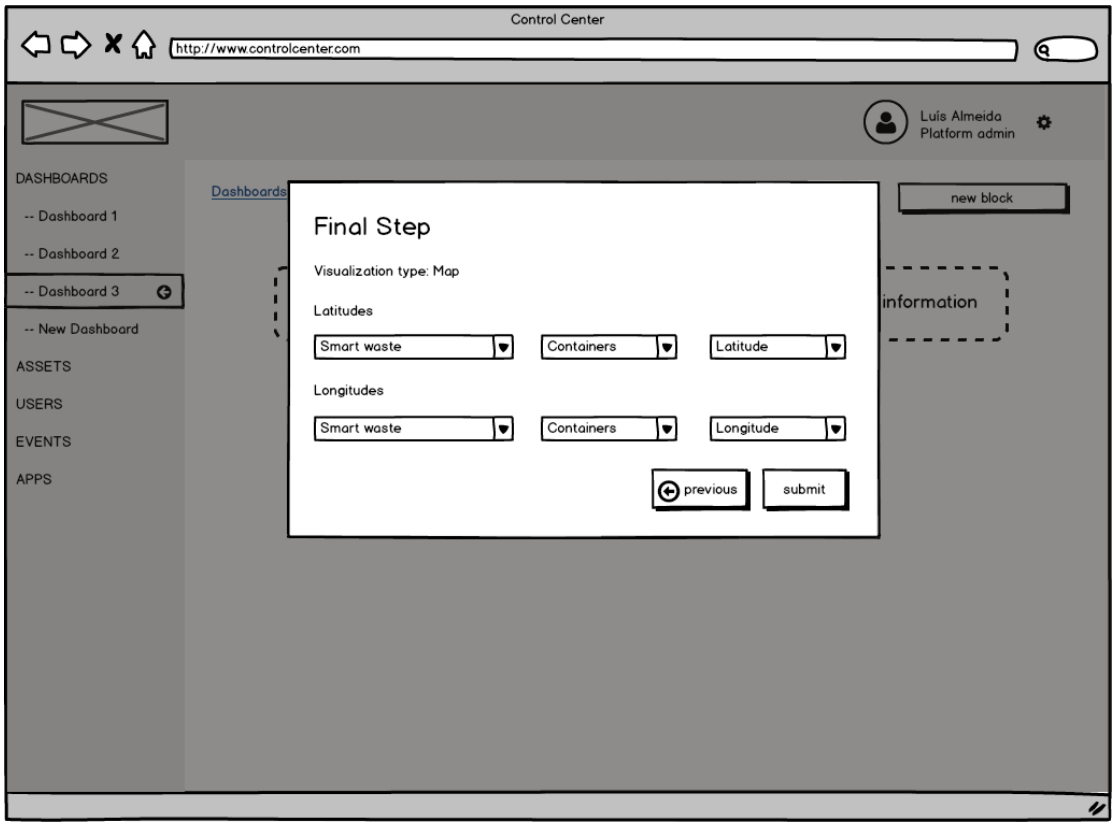


Imagem 21 | *Mockup* da interface do último passo da ferramenta de criação de blocos para os *dashboards*

Desta forma, assegura-se uma uniformização na maneira como são escolhidos os dados, e fica ao critério do utilizador escolher os valores certos para as formas de visualização certas, não limitando as suas escolhas.

Contudo, a solução encontrada para a ferramenta de criação de blocos para os *dashboards* limita o utilizador na medida em que oferece apenas algumas formas de visualização, e sempre que for necessário acrescentar mais uma forma, existirá trabalho de desenvolvimento. Porém, esta é uma limitação impossível de contornar neste cenário: cada forma de visualização requer uma formatação da estrutura e da própria informação muito específica, bem como de *frameworks* externas para auxiliar a sua implementação, logo é inconcebível para esta

investigação desenvolver um ferramenta que não limite o utilizador nas formas de visualização, principalmente devido aos limites impostos pelos prazos de entrega.

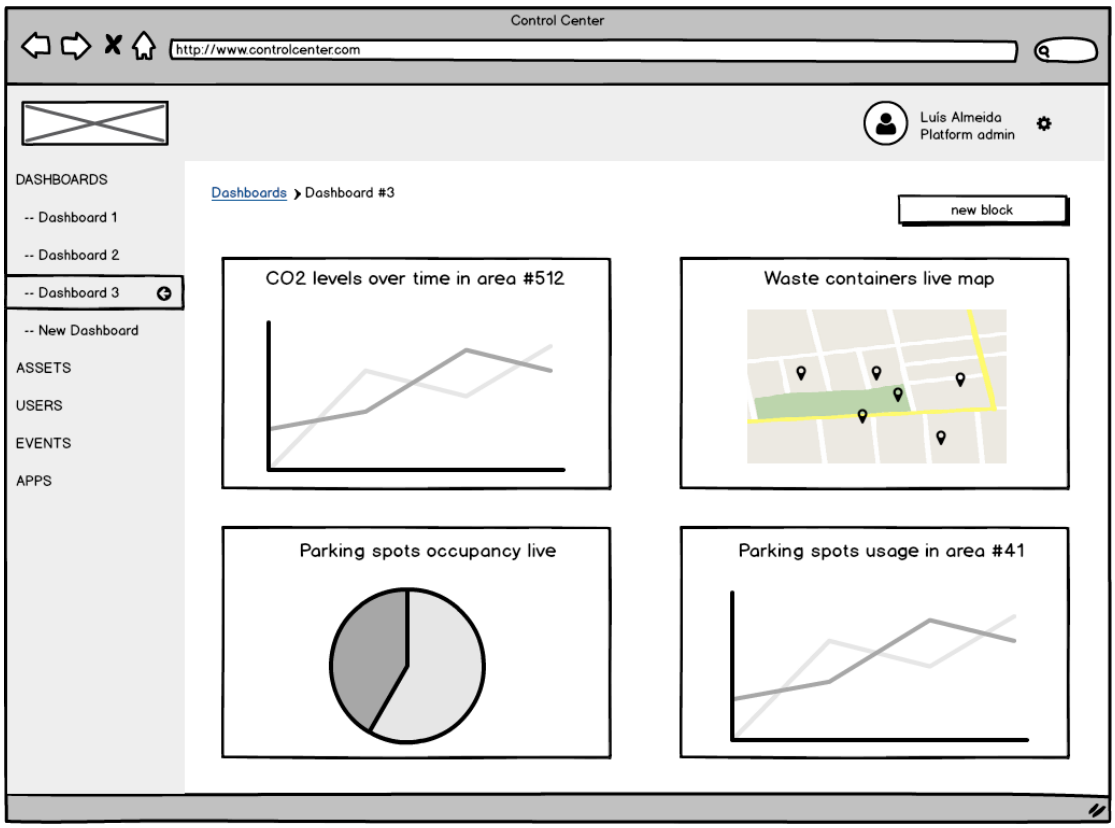


Imagem 22 | *Mockup* do aspeto de um *dashboard* com blocos

Finalmente, na Imagem 22 (acima) é possível verificar o aspeto de um *dashboard* já com alguns blocos criados, cada um com o título que o utilizador especificou no primeiro passo da ferramenta de criação de blocos (Imagem 19). Para tornar esta página mais interativa, cada um dos gráficos foi desenvolvido para responder à interatividade do utilizador – ao passar o cursor do rato em cima de uma das linhas dos gráficos de linhas, por exemplo, apresenta-se o valor naquele ponto, enquanto que num mapa, o utilizador pode clicar nos diferentes marcadores para ver de que objeto se trata e o seu estado atual.

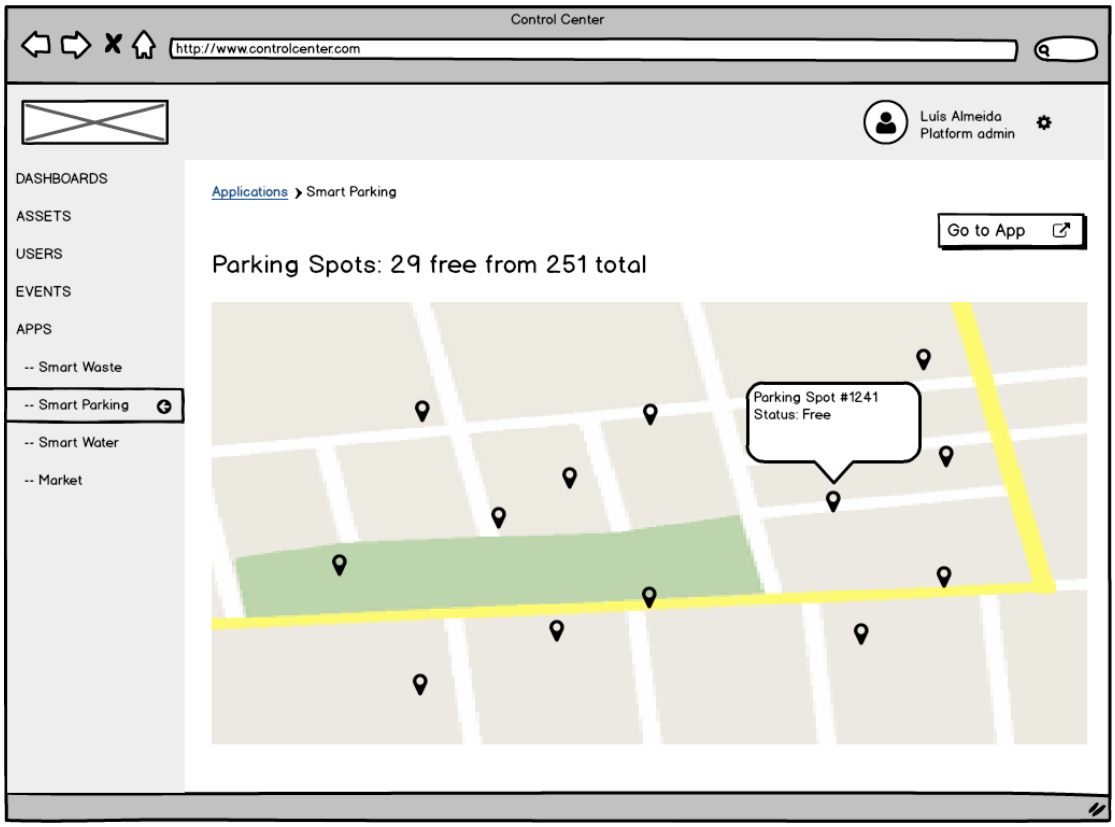


Imagem 23 | *Mockup* da interface da página de apresentação de uma aplicação associada ao "*control center*"

No que diz respeito às páginas de cada uma das aplicações, tal como já foi referido, tratam-se de interfaces para visualizar apenas os dados mais importantes – para o utilizador ver mais em detalhe os conteúdos relativos a determinada aplicação terá de navegar para a plataforma específica da aplicação, clicando no botão no canto superior direito que indica *Go to App* (imagem 23).

Os conteúdos que irão aparecer nesta página de apresentação da aplicação no *"control center"* serão especificados aquando do registo da aplicação no sistema.

4.4. Esquema da arquitetura física

Enquanto que no capítulo da arquitetura da plataforma o principal foco foi explicar os módulos do *"control center"*, a responsabilidade e o que representa cada um deles, de forma a aprofundar também o conceito do produto desta investigação, neste capítulo, procede-se à explicação da arquitetura física do sistema, ou seja,

as entidades e componentes envolvidas que tornam possível a distribuição do *"control center"* no *browser* de cada cliente (Imagem 24).

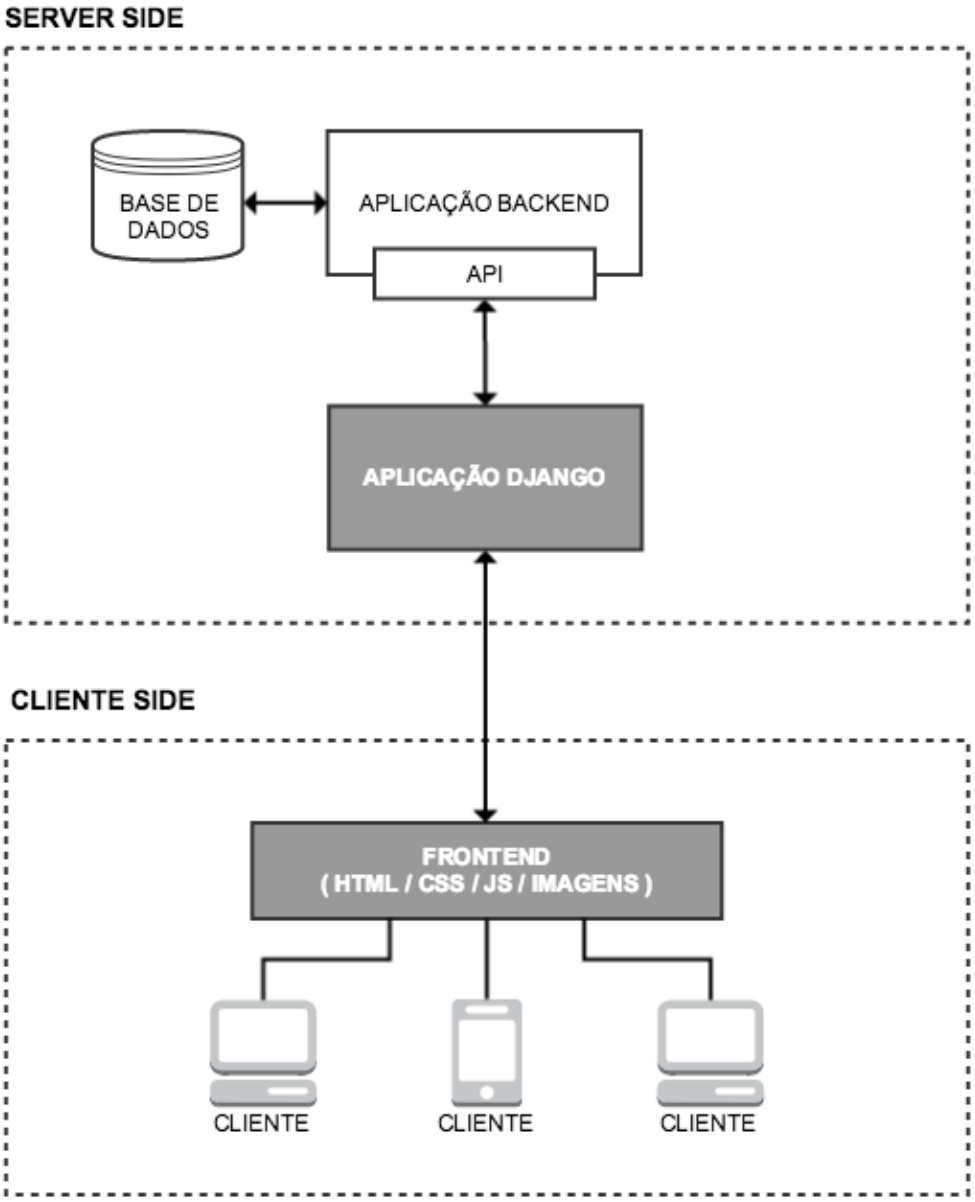


Imagem 24 | Esquema da arquitetura física do sistema

Tal como foi referido anteriormente no capítulo do enquadramento teórico, uma plataforma *web* é constituída essencialmente por duas grandes componentes – a componente *server side* e a componente *cliente side.* A plataforma *web* em desenvolvimento nesta investigação enquadra-se neste esquema, e tal como se pode verificar na imagem 24, a arquitetura está divida nestas duas componentes.

Esta investigação é desenvolvida em contexto empresarial pelo que a função do investigador não é arquitetar a plataforma mas sim desenvolver determinadas componentes, potenciando o seu conhecimento ciêntifico na área. Neste caso, o papel do investigador é desenvolver, de acordo com a imagem 24, apenas as componentes representadas com rectângulos com fundo cinzento (aplicação *Django* e *Frontend*).

A componente *server side* (servidor) do sistema do *"control center*" é composta pela base de dados, pela aplicação *backend*, uma API e a aplicação *Django*.

A base de dados é o conjunto de informação armazenada e organizada que vai sendo gerada ao longo do tempo. Sempre que o utilizador, por exemplo, cria uma *asset*, esta é armazenada na base de dados. A base de dados tem de estar preparada para receber enormes quantidades de dados, por vezes em curtos intervalos de tempo e, portanto, tem de haver o cuidado necessário para que esteja operacional a maior pecentagem de tempo possível. Esta componente está a cargo de outros colaboradores da empresa que são responsáveis pela instalação, configuração e manutenção da base de dados e dos seus conteúdos. E o diagrama ER? Em que parte contribuiste??

A aplicação *backend* é aqui designada como sendo a plataforma que modela e serve os conteúdos da base de dados para a aplicação *Django* – esta componente tem uma outra componente associada: a API (*Application Program Interface*), representada também ela na imagem 24 No caso desta investigação, o que a componente da aplicação *backend* representa é um conjunto de modelos, classes e métodos que permitem interagir com a informação que está na base de dados, sendo que, maioritariamente, por motivos relacionados com o modelo de negócio da empresa, esta componente disponibiliza uma API para estabelecer a comunicação entre a aplicação *Django* e a base de dados. Na prática, quer seja a aplicação *Django* desenvolvida pelo investigador, ou qualquer outra aplicação a ser desenvolvida que necessite de acesso aos conteúdos da base de dados, apenas terá de comunicar com a API para ter acesso aos conteúdos, eliminando assim a necessidade de desenvolver múltiplas aplicações *backend* específicas

para cada aplicação. O desenvolvimento de APIs é, atualmente, uma prática recorrente no desenvolvimento de *software web*, devido à possibilidade de estas serem utilizadas por diferentes aplicações, independentemente da sua natureza, agilizando assim o trabalho do programador, principalmente nas componentes *backend* que necessitam constantemente de aceder à base de dados (Lane, 2015).

A aplicação *Django* é uma das componentes desenvolvidas inteiramente pelo investigador. Esta componente está no domínio do servidor, ou seja, faz parte do *server-side,* querendo isto dizer que o utilizador final nunca terá acesso ao seu conteúdo – é importante perceber que a aplicação *Django*, do ponto de vista do cliente, funciona como uma *"black box",* ou seja, o cliente faz os pedidos ao servidor pela informação, mas nunca tem conhecimento dos processos desenrolados antes de receber a resposta. É nesta componente que é processada toda a lógica associada aos pedidos e tratamento da informação proveniente da API, processos relacionados com a autenticação e com a persistência de dados na sessão do utilizador. Na prática, sempre que o utilizador abre uma qualquer parte do "*control center*", este faz um pedido HTTP ao servidor, mais concretamente a esta aplicação *Django*, que interpreta o pedido, constrói a resposta e envia de volta para o utilizador uma página HTML com toda a informação necessária. É evidente que esta componente tem uma grande importância na arquitetura do *"control center*" pois é aqui que está grande parte da lógica e do desenvolvimento do investigador.

Atualmente a dependência numa componente *server side* como a utilizada no *"control center*" é discutível: Spike Brehm, *Frontend Engineer* na *Airbnb,* uma das plataformas *web* mais utilizadas globalmente para aluguer de imóveis, escreveu um artigo intitulado "*Isomorphic Javascript: The Future of Web Apps*", no qual explica de que forma é que as tecnologias *web*, atualmente, permitem reduzir a necessidade de desenvolver componentes *server-side*, sendo toda a lógica processada do lado do cliente, através de *Javascript*, uma tecnologia explicada anteriormente no capítulo do Enquadramento Teórico. A constante evolução da

capacidade de processamento dos *browsers* dos clientes, a grande quantidade de *frameworks* para o desenvolvimento com tecnologias *client side*, nomeadamente o *Javascript*, e o cada vez mais frequente uso de APIs fazem com que já não haja uma necessidade de depender tanto na componente *server side,* utilizando o cliente para a maioria dos processos relacionados com os pedidos, tratamento e disponibilização da informação (Brehm, 2013).

Contudo, no caso particular desta investigação, e como se pode verificar pelo esquema da arquitetura física da plataforma representado na imagem 24, optou-se por alocar o máximo de processamento e de lógica na parte do servidor (*server side*), por dois grandes motivos que foram acordados entre o investigador e a empresa: i) O primeiro motivo vai ao encontro do facto de existir a já referida "caixa negra" que assegura que o cliente nunca irá ter acesso aos processos mais importantes do sistema e por consequente aumenta a segurança da plataforma no geral. ii) O segundo motivo está relacionado com a possível carga excessiva que seria colocada nos computadores de cada cliente, uma vez que pode ser necessário tratar uma grande quantidade de informação, e, ao não saber se o cliente possuí um computador com maior ou menor capacidade de processamento, o facto deste processamento residir no servidor reduz a probabilidade de clientes com computadores menos potentes terem dificuldades em utilizar a plataforma.

4.5. Tecnologias e ferramentas utilizadas

O capítulo do desenvolvimento desta investigação termina com a especificação das tecnologias e ferramentas utilizadas na construção do *"control center"*. É importante especificar quais as tecnologias que foram escolhidas, porquê e em que ocasiões foram aplicadas – desta forma, esta investigação contribui com um conjunto de *guidelines* para outros investigadores e *developers* seguirem e construírem as suas próprias plataformas que encaixem no perfil da plataforma aqui em estudo.

Como já foi referido anteriormente, o papel do investigador é de desenvolver apenas a componente *frontend* e a aplicação *Django* descrita na arquitetura da

plataforma, pelo que as tecnologias que são descritas nos parágrafos seguintes correspondem apenas às tecnologias utilizadas para a produção destes componentes em específico.

Começando pela aplicação *Django*, tal como o nome indica e como já foi referido, é uma aplicação construída utilizando uma *framework* chamada *Django*, que assenta na linguagem de programação *Python*. Esta *framework* segue a filosofia *"batteries-included",* querendo isto dizer que os seus criadores idealizaram esta ferramenta para oferecer aos programadores tudo o que precisam para desenvolver uma aplicação *web*, sem haver necessidade de diversas outras ferramentas (Makai, 2015).

A razão pela qual *Django* é uma tecnologia utilizada nesta investigação para o desenvolvimento do *"control center*" prende-se com o facto de ser necessária uma *framework* para desenvolver a aplicação que interage diretamente com o *backend* (API) do sistema, tal como está representado na imagem 24. A utilização de uma *framework* para desenvolver esta aplicação permite que o seu desenvolvimento seja mais rápido e apoiado, pois existe já uma base na qual o programador se pode apoiar e um conjunto de ferramentas disponíveis para agilizar a implementação de diversas funcionalidades. Havendo esta necessidade*,* foi escolhida a *Django* por várias razões: a primeira vai ao encontro do facto da empresa com a qual esta investigação é realizada utilizar esta ferramenta em vários outros projetos e ter já um vasto *know-how* para apoiar o investigador na resolução de problemas que possam decorrer durante o desenvolvimento. A segunda razão vai ao encontro das tendências da escolha de ferramentas para desenvolvimento de aplicações *web* atualmente – esta *framework* é conhecida pela sua estabilidade, performance e pela cada vez maior comunidade que a utiliza e apoia o que origina um elevado número de material como tutoriais e *packages* externos que agilizam ainda mais o desenvolvimento (Makai, 2015).

Como esta ferramenta assenta no padrão de arquitetura de *software* MVC (*Model View Controller*) (Django Software Foundation, 2015) é importantíssimo que a arquitetura da plataforma esteja bem definida antes de começar o

desenvolvimento propriamente dito e, por isso, o modelo apresentado na Imagem 10 tem grande impacto na utilização desta *framework*.

É também importante referir que a utilização da *framework Django implica que* toda a plataforma terá de assumir uma estrutura de pastas e ficheiros bem definida para permitir o bom funcionamento da ferramenta, estrutura essa que está representada na figura abaixo.

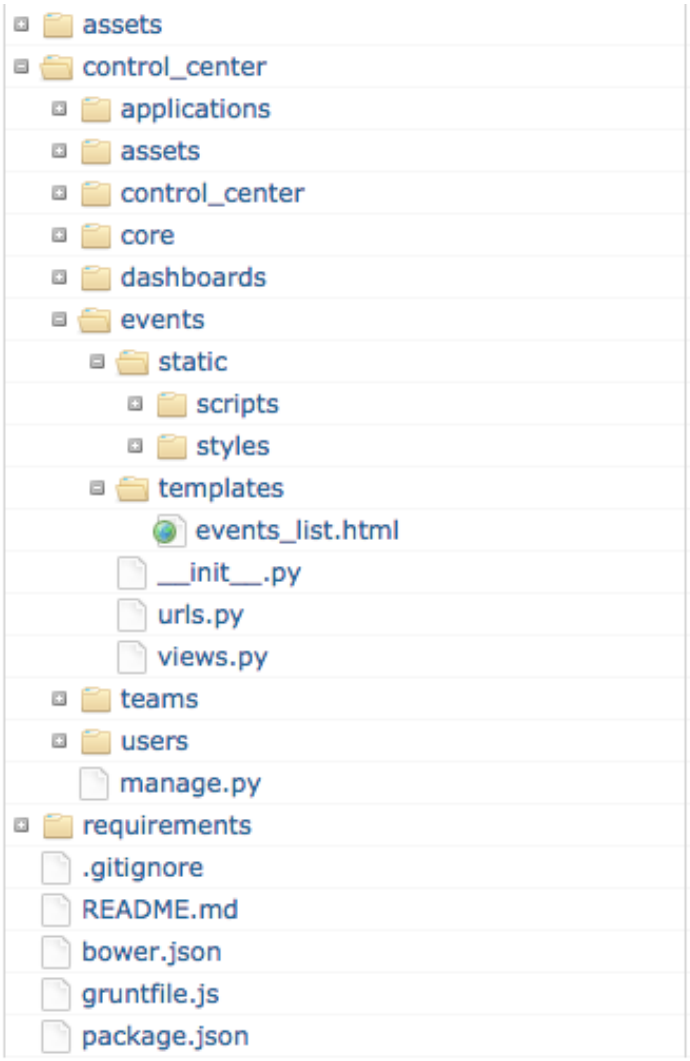


Imagem 25 | Estrutura de pastas e ficheiros do "*control center*"

Na Imagem 25 está representada a organização das pastas e ficheiros do todo o trabalho desenvolvido pelo investigador. Esta hierarquia corresponde à arquitetura dos módulos da plataforma: dentro da pasta *"control_center"* estão as pastas *applications, assets, dashboards, events, users, teams*, todos eles designados

como módulos na arquitetura da plataforma representada na Imagem 10. Dentro de cada uma destas pastas a organização dos seus ficheiros é igual, sendo que existe sempre uma pasta *static* e uma pasta *templates*. A *static* dividida em *styles* e *scripts*, e cada uma delas serve para armazenar os ficheiros CSS e Javascript próprios daquele módulo. Na pasta *templates* estão todos os ficheiros HTML que são utilizados para produzir as interfaces próprias do módulo. Na Imagem 25 pode-se verificar esta organização para na pasta *events* dentro do *control_center*, que representa como estão organizados os ficheiros do módulo de eventos da plataforma *"control center*". Para além destes pastas e ficheiros especificados anteriormente, estão, também, dentro de cada pasta de cada módulo, um ficheiro *urls.py* e um outro *views.py.* Estes ficheiros fazem parte da *framework Django* e têm um papel fundamental para o funcionamento da aplicação: *urls.py* é um ficheiro no qual são especificados todos os URLs que estão disponíveis para o utilizador aceder, dentro daquele módulo específico. Por exemplo, o utilizador pode aceder a *http://www.controlcenter.com/events/list* para aceder à listagem de eventos, e a *http://www.controlcenter.com/events/41512* para aceder à interface de detalhe do evento que é identificado pelo número 41512. A organização de URLs numa aplicação *web* é um aspeto importante a ter em conta quando se desenvolve uma plataforma. Na prática, esta técnica é conhecida como *URL dispatching*, sendo uma das várias ferramentas que a *framework Django* oferece para a construção de plataformas *web.* Tim Berners-Lee, criador da *World Wide Web,* escreveu um artigo onde refere a importância da aplicar esta técnica nas plataformas *web:* é importante ter um conjunto de *URLs* que estão disponíveis na plataforma para o utilizador aceder, e que, independentemente do ficheiro que esteja a ser acedido, o padrão no link para chegar ao seu conteúdo deve ser igual e constante ao longo do tempo, caso contrário, utilizadores que tenham guardado o *URL* no seu *browser* para aceder diretamente ao conteúdo de uma página, poderão não o conseguir fazer simplesmente porque um nome de um ficheiro foi alterado, o que causa frustração (Berners-Lee, 1998).

Assim sendo, o *"control center*" tem um padrão de URLs muito bem definido, de acordo com a arquitetura da plataforma especificada na Imagem 10, que permite

chegar às listagens, edições e detalhes dos mais variados objetos de cada módulo através de um padrão uniformizado da seguinte forma: *dominio.com/modulo/list* para ver a listagem; *dominio.com/modulo/123/* para ver o detalhe de um objeto; *dominio.com/modulo/123/edit/* para aceder à interface de edição; *dominio.com/modulo/132/delete/* para remover determinado objeto. Ou seja, o padrão de URLs do *"control center*" pode-se resumir com a seguinte fórmula: *dominio.com/<módulo>/<identificador_do_objeto>/<ação>*. Se o utilizador editar manualmente o URL para aceder a uma qualquer página que não exista, ou a um determinado identificador de objeto que não esteja disponível, a *framework Django* deteta e redireciona automaticamente o utilizador para uma interface específica a explicar que o URL que tentou aceder não existe.

O ficheiro *views.py* é um dos ficheiros base da *framework Django* que tem também ele um ficheiro muito importante no *"control center*". Cada módulo tem um destes ficheiros, que tratam toda a informação antes de ele chegar ao utilizador: como já foi explicado anteriormente, optou-se por tratar toda a informação do lado do servidor, antes que esta chega ao utilizador, e portanto, é necessária uma lógica que prepare os dados e que os disponha na página HTML de forma eficiente. No ficheiro *views.py* é feito o pedido à API desenvolvida pela empresa, é recebida a informação, é tratada e enviada para um dos ficheiros HTML que estão na pasta *templates* do módulo. Na prática, o utilizador acede, por exemplo, à listagem de eventos do *"control center*" e através do ficheiro *urls.py*, a *framework* sabe o que o utilizador tentou aceder, e "aciona" a função correspondente do *views.py.* Nessa função, é realizado um pedido à API pela informação necessária, realizado o processamento desta, e finalmente enviada para o cliente onde é apresentada com base no respetivo *template*

O modelo deste funcionamento está representado abaixo na Imagem 26, onde é possível verificar como é que a informação chega ao cliente após o seu pedido à aplicação *Django*.

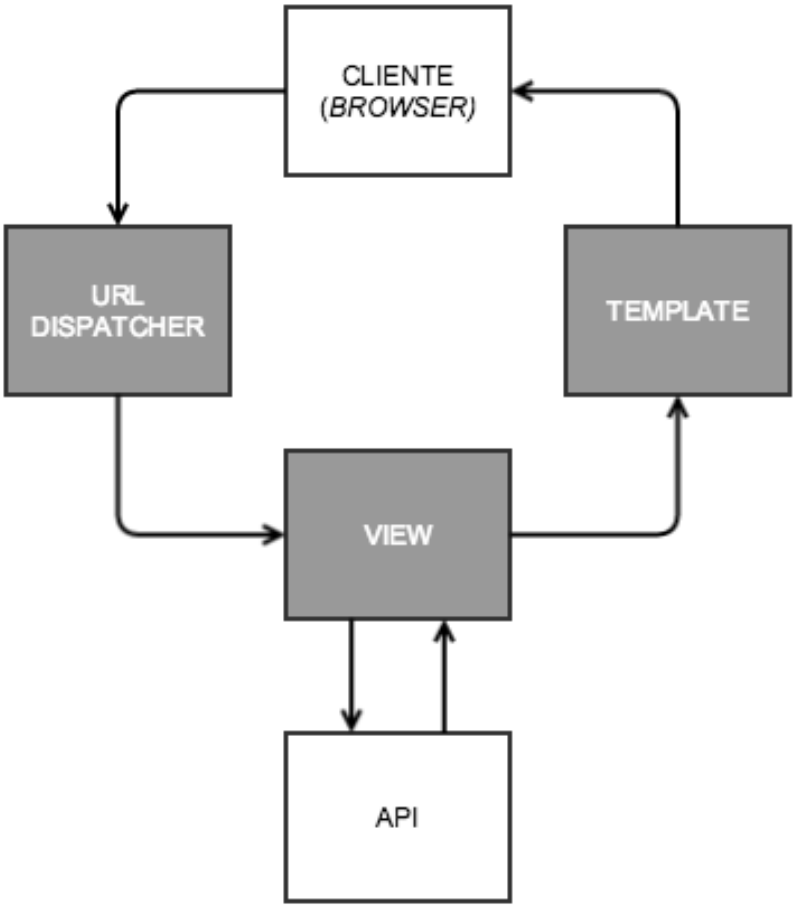


Imagem 26 | Modelo de funcionamento da *framework Django* no "*control center*"

Para além das pastas que representam cada um dos módulos, dentro da pasta *"control_center"* (de notar que, apesar de na Imagem 10 o módulo *teams* aparece dentro do módulo *users,* na estrutura de ficheiros optou-se por estes permanecerem lado a lado, e o módulo de autenticação está dentro da pasta *users*), existe ainda uma outra pasta denominada *"control_center"* que contém o ficheiro *settings.py.* Este ficheiro contém todas as definições gerais da *framework Django*: é neste ficheiro que são especificados quaisquer *packages* externos utilizados, definições gerais para a plataforma como por exemplo o URL da API, entre outras definições que podem ser encontradas no *website* da documentação oficial da *framework*.

No lado do cliente (módulo *frontend* representado na Imagem 24), e por forma a explicar a restante estrutura de ficheiros da Imagem 25, o investigador tem o

papel de desenvolver também o *software* que é utilizado do lado do computador do cliente. Ou seja, se até agora se explicou o funcionamento da aplicação *Django*, como esta faz os pedidos à API e trata a informação para a entregar ao cliente, explica-se agora a tecnologia utilizada no *browser* e que permite a visualização e a interação por parte do utilizador.

Nesta componente são utilizadas três tecnologias base, muitas vezes utilizadas na construção de aplicaões *web:* HTML, CSS e Javascript.

Como já foi referido em capítulos anteriores desta investigação, nomeadamente no capítulo do enquadramento teórico, estas três tecnologias são as tecnologias de suporte ao desenvolvimento da componente *frontend* de quase todos os *websites*, sendo que cada uma das três tem um papel específico no que diz respeito a mostrar a informação ao utilizador – muito resumidamente, a HTML especifica a estrutura de uma página *web*, o CSS especifica os estilos dessa estrutura, e a *Javascript* permite adicionar uma camada de interação mais rica a essa estrutura.

De acordo com a estrutura de ficheiros do *"control center*" representada na Imagem 25, pode-se verificar onde estão, na prática, estas três tecnologias: em cada uma das pastas *templates* de cada módulo estão os ficheiros. Estes ficheiros são integrados na *framework* Django, que disponibiliza técnicas para reproduzir a informação dinâmica consoante o pedido feito pelo utilizador, por exemplo, quando o utilizador navega no *"control center*" para a página de detalhe de um determinado *asset*, o template HTML tem de estar preparado para reproduzir a informação daquele *asset* em específico, e isso é conseguido através de blocos de código inseridos entre duas chavetas (designados como *template tags*), como está exemplificado na imagem 27:

```
<div id="asset-detail-ctn">
    <div class="row spaced">
        <div class="col-sm-12 col-md-12 col-lg-12">
            <div class="pull-left">
                <h3>{% trans 'Asset Details' %} #{{ asset.id }} - {{ asset.name }}</h3>

            </div>
            <div class="text-right">
                <a href="{% url 'assets:edit' asset_id=asset.id %}">
                    <button class="border-btn btn-smaller" type="button">{% trans 'edit asset' %}</button>
                </a>
            </div>
        </div>
    </div>
```

Imagem 27 | Exemplo da utilização do HTML integrado na *framework* Django

Este é um pedaço de código escrito em HTML que utiliza a tecnologia das *template tags* para imprimir o identificador e nome do *asset*, num exemplo muito simples da utilização desta técnica. É através destas *template tags* que é possível ter um *template* que pode ser aplicado para todos os casos, como por exemplo, o *template asset_detail.html* que é utilizado para visualizar todas as páginas de detalhe de todas as *assets*.

No que diz respeito ao CSS, os ficheiros correspondentes estão nas pastas *styles* dentro de cada pasta *static* de cada módulo. O facto de estarem todos dentro de uma pasta designada por *static* é uma exigência da *framework Django*. Tal como já foi explicado anteriormente, estes ficheiros têm a missão de dar estilo à estrutura HTML. Contudo, nesta investigação é utilizada uma outra tecnologia para produzir estes ficheiros CSS: ao invés de escrever diretamente CSS, é utilizada uma tecnologia chamada LESS para produzir os ficheiros CSS. Esta ferramenta é designada tecnologicamente como um *Preprocessor* (The core less team, 2015). LESS disponibiliza um conjunto de técnicas como a possibilidade da criação de variáveis e de funções que permitem agilizar a escrita do CSS. Na prática, é possível definir uma variável com uma cor e esta é utilizada nas mais diversas ocasiões, eliminando assim a necessidade da repetição de cores e outros estilos nos diferentes ficheiros CSS.

Imagem 28 | Explicação do funcionamento do *LESS* (Tock, 2013)

Na Imagem 28 (Tock, 2013) está especificado um exemplo da utilização de LESS: é possível definir uma espécie de função (*mixin*), que recebe um parâmetro e produz o resultado final em CSS. É importante referir que estes ficheiros LESS não têm lugar numa aplicação *web,* isto é, eles não são utilizados diretamente na aplicação, mas sim os ficheiros CSS que produz e, por isso, é designada um *preprocessor.* Ao verificar novamente a Imagem 25, que retrata a estrutura de ficheiros do *"control center*", a primeira pasta *"assets*" serve para armazenar estes ficheiros, ficando assim de fora da estrutura principal do *"control center*", ou seja, fora da pasta *"control_center*".

Quanto à Javascript, os ficheiros correspondentes estão, tal como no caso dos ficheiros CSS, dentro das pastas *static* de cada um dos respetivos módulos, contudo ainda dentro de uma outra pasta, denominada *scripts*. À semelhança dos ficheiros CSS, os ficheiros Javascript são referenciados no *template* HTML correspondente, e para agilizar a sua escrita foi utilizada uma outra tecnologia bastante utilizada nas aplicações *web* atuais: *jQuery.* Esta tecnologia é descrita como uma biblioteca de Javascript bastante versátil e extensível que facilita vários aspetos no desenvolvimento de aplicações *web*, como por exemplo a manipulação da estrutura HTML, animações, controlo de eventos, entre outros (The jQuery Foundation, 2015). A utilização desta tecnologia é "quase obrigatória" em qualquer aplicação *web* atual devido às características que estas adotam: são aplicações dinâmicas, com muita interação associada, e para isso é necessário

uma utilização cada vez mais intensiva da *Javascript*, e o *"control center*" não é exceção.

Para apoiar o investigador na implementação dos *layouts* desenvolvidos foram utilizadas ainda duas outras ferramentas: *bootstrap* e *font awesome*.

Começando pelo *bootstrap*, esta é uma ferramenta que agiliza a construção da estrutura HTML: através de um conjunto de estilos (CSS) disponibilizados pela ferramenta, é possível construir muito rapidamente a estrutura de um *website*, e para além disso, esta fica ainda preparada para ser visualizada tanto nos ecrãs mais pequenos como os dos *smartphones*, como nos ecrãs maiores, passando pelos *tablets* (Bootstrap, 2015). Esta característica tem o nome técnico de *"responsive*" e significa assim que um *website* deve ajustar a sua estrutura e a forma como os conteúdos são apresentados de acordo com o tamanho e a orientação do ecrã através do qual está a ser visualizado, para permitir uma melhor experiência ao utilizador e eliminar a necessidade de desenvolver múltiplas versões das plataformas (Knight, 2011). Exemplificando, é possível verificar, através da Imagem 11, que a interface do *"control center*" contém um menu lateral esquerdo sempre presente. Este menu, nos dispositivos com ecrã mais pequeno irá ocupar demasiado espaço, espaço esse que seria fundamental para apresentar a informação mais importante e, portanto, a melhor estratégia será esconder esse menu nestes ecrãs mais pequenos e torna-lo visível apenas quando o utilizador pressiona um botão, tal como se demonstra na imagem seguinte (Imagem 29). O ícone no canto superior direito do ecrã do *smartphone* permite que se expanda um menu com todos os botões que teria o menu lateral esquerdo.

Imagem 29 | Exemplo da apresentação da interface do "*control center*" num dispositivo mais pequeno

Fontawesome é também uma ferramenta para ajudar no desenvolvimento da interface de uma plataforma *web.* Esta ferramenta oferece uma vasta coleção de ícones que podem muito facilmente serem adicionados à interface de um *website* através de CSS. Este ícones podem ainda ser customizados na sua cor e tamanho (Font Awesome, 2015). Esta ferramenta facilita o trabalho do investigador na medida em que a interface pode ter uma iconografia rica e personalizada sem que seja necessária a exportação de uma imagem por cada ícone necessário na plataforma. Para além disso, estes ícones são disponibilizados pela ferramenta em forma de vetor, significando isto que as suas linhas são bem definidas, em qualquer tamanho e em qualquer resolução de ecrã.

Por fim, falta especificar as tecnologias utilizadas pelo investigador que não têm impacto direto no produto final, mas que foram fundamentais para o seu desenvolvimento. Estas são três e são cruciais no desenvolvimento de qualquer

projeto inserido numa equipa e que tenha um tamanho considerável, como é o caso do *"control center*": *git, grunt* e *bower*.

O *GIT,* é um sistema de controlo de versões (*VCS – Version Control System), open-source* e gratuito (Git, 2015). Sendo esta investigação realizada em contexto empresarial e em conjunto com outros programadores, é fundamental que haja uma ferramenta que permita controlar mudanças feitas no código dos diferentes ficheiros e que permita juntar facilmente mudanças feitas por pessoas diferentes. Nesta investigação, esta ferramenta foi utilizada não só pelas razões apontadas acima, mas também porque todos os projetos realizados na empresa em questão, independentemente das suas características, são desenvolvidos com o apoio desta.

O Grunt é um *"task runner*" que permite automatizar uma série de procedimentos típicos do desenvolvimento *frontend* de plataformas *web* (Grunt, 2015). Um *task runner* permite configurar um conjunto de tarefas que são acionadas automaticamente para realizar ações que se tornam repetitivas. No caso prático desta investigação, o *Grunt* é utilizado para compilar os ficheiros LESS em ficheiros CSS e coloca-los na pasta certa, sempre que são realizadas alterações, eliminando assim a necessidade que existiria do investigador compilar manualmente os ficheiros sempre que houvesse modificações. O *Grunt* pode ser utilizado para inúmeros outros casos, como por exemplo, para compactar ficheiros JS e CSS para que estes ocupem menos espaço e sejam mais rápidos a carregar no *browser* do cliente.

O *Bower* é um sistema de gestão de "*packages"* para a *web* (Bower, 2015). Através do *Bower* é possível ao investigador facilmente "instalar" no *"control center*" bibliotecas externas, plugins, *frameworks,* etc. Por exemplo, o *bootstrap* é uma ferramenta que pode ser adicionada a um projeto através do *bower*, com uma simples linha de comando, poupando assim o trabalho de fazer *download* dos ficheiros correspondentes e de os colocar manualmente no projeto. É mais uma ferramenta que se torna particularmente útil quando o projeto necessita de um número considerável de bibliotecas/*frameworks*/*plug-ins* externos.

Concluindo este capítulo, é importante referir que as tecnologias especificadas anteriormente e que foram utilizadas no desenvolvimento do *"control center*" são quase todas elas consideradas atualmente essenciais no desenvolvimento de plataformas *web* por parte da comunidade de *developers* e que pode ser facilmente confirmado ao verificar os mais conhecidos fóruns, blogs e *websites* dedicados ao tema. O objetivo do investigador, neste capítulo, foi o de especificar quais foram as tecnologias escolhidas e como foram aplicadas neste projeto, por forma a contribuir com informação importante para outros investigadores e programadores sobre que tecnologias utilizarem em projetos com características semelhantes ao aqui apresentado.

5. Testes funcionais e análise dos resultados

Depois do capítulo do desenvolvimento, é altura de confrontar de novo o capítulo das finalidades e objetivos estabelecidos para esta investigação, no qual é possível verificar que faz parte dos objetivos acordados entre o investigador e a empresa com a qual esta investigação é realizada, a realização de testes ao produto desenvolvido.

Foi possível também constatar, no capítulo do enquadramento teórico, que existem diversas tipologias de testes orientados para aplicações *web*, cada uma com características diferentes em que se pretende testar aspetos mais específicos de uma plataforma.

Contudo, importa especificar o contexto no qual se enquadra o desenvolvimento desta plataforma, através de uma perspetiva mais relacionada com os procedimentos de desenvolvimentos adotados pela empresa: sendo esta investigação realizada em contexto empresarial, a empresa que intervém tem um papel fundamental na forma como é abordado o desenvolvimento e, por consequente, nos testes à plataforma. A empresa em questão adopta a metodologia de desenvolvimento *"agile*" em todos os seus projetos, e o desenvolvimento do produto desta investigação foi também ele enquadrado nesta metodologia e, portanto, influenciado pelos seus princípios.

Em 2001 foi introduzido o termo *"Agile*" pela primeira vez num contexto de desenvolvimento de *software*, no *"Manifesto for Agile Software Development*", também conhecido como *"Agile Manifesto*" que indica quatro princípios base para esta metodologia de desenvolvimento:

"Individuals and interactions over processes and tools"; "Working software over comprehensive documentation"; "Customer collaboration over contract negotiation"; "Responding to change over following a plan" (Beck et al., 2001).

É possível verificar, através destes princípios, que esta metodologia aponta para um desenvolvimento flexível, suportado na constante comunicação e colaboração entre os intervinientes, preocupado não só em desenvolver mais rapidamente

mas sobretudo em manter a qualidade deste, sendo assim uma atitude a adoptar e sob a qual um projeto é encarado (Mark, 2011).

Os testes foram realizados neste contexto, significando que a plataforma foi testada por um colaborador da empresa especializado profissionalmente em testar *software*, mais propriamente plataformas *web.*

Os testes foram realizados em ambientes totalmente controlados pelo colaborador em questão, ou seja, com acesso não apenas à componente *frontend* e à aplicação *Django* desenvolvida nesta investigação, mas também a todas as outras componentes representadas na Imagem 24 que representa a arquitetura física da plataforma, possibilitando assim ao colaborador (*"tester*") verificar, no caso de alguma anomalia, se o problema é de facto causado pelas componentes desenvolvidas pelo investigador ou por outras desenvolvidas pela empresa.

Assim sendo, e tendo em consideração a natureza e a finalidade desta plataforma, foram realizados três tipos de testes: os funcionais, os de performance e os de usabilidade. Estes três tipos de testes foram analizados no capítulo do Enquadramento Teórico desta investigação, mais propriamente no subcapítulo dos Testes orientados para aplicações *web*.

Os testes funcionais, no caso prático desta investigação e da plataforma aqui desenvolvida, pretendem apurar se todas as funcionalidades produzem os resultados esperados, não só na componente desenvolvida pelo investigador, mas também nas restantes componentes, principalmente na base de dados (por exemplo, se a criação/edição de um *asset*/utilizador é realizada com sucesso e os dados são gravados corretamente na base de dados);

Quanto aos testes de performance, estes prendem-se com a capacidade da plataforma estar pronta a apresentar ao utilizador grandes quantidades de dados minimizando a influencia na sua experiência. Para realizar estes testes o colaborador responsável pela realização dos testes prepara um ambiente com um número significativo de dados (*assets*, utilizadores, *dashboards*, entre outros) e verifica como responde a plataforma *web.*

Por fim, os testes de usabilidade têm o objetivo de verificar se os mecanismos desenvolvidos pelo investigador para possibilitar a interação do utilizador com a plataforma para o desenvolvimento de tarefas tais como a criação/edição/listagem de objetos, validações de formulários, criação de *dashboards* e dos respetivos blocos, funciona de forma adequada e de forma a maximizar a experiência do utilizador. Um exemplo prático deste teste na plataforma desenvolvida nesta investigação, é o teste realizado à ferramenta de criação de blocos para os *dashboards:* sendo uma ferramenta complexa ao nível das tarefas que deve permitir realizar e dos resultados que deve permitir obter, é importante que esta ferramenta receba a atenção adequada aquando dos testes para que possam ser detetadas possíveis falhas no seu funcionamento.

Aquando da fase de testes, todos os problemas detetados (denominados como *"bugs*") são registados na plataforma *"Asana*", uma ferramenta de gestão de *"bugs*" utilizada pela empresa para este efeito (Asana, 2015). Nesta ferramenta, o investigador pode verificar o registo de todos estes problemas, a sua discrição, a causa e que ações deve realizar para o replicar. A partir destes dados é possível partir para a resolução do problema, e após a sua resolução marcar o respetivo *"bug*" como resolvido, sendo este o fluxo de funcionamento de detecção e resolução de todas as anomalias detetadas na fase de testes.

Este registo de problemas na plataforma da empresa pode então ser considerado, neste caso, como o resultado dos testes realizados à plataforma, e está presente na secção de anexos desta investigação.

Os *bugs* detetados na plataforma foram todos eles problemas de fácil resolução relacionados com questões de usabilidade na plataforma, como por exemplo, problemas gerados nos formulários de criação de objetos, nos quais é necessária uma validação da informação introduzida pelo utilizador. Devido à metodologia de desenvolvimento dentro da empresa, que motiva a rapidez na detecção e resolução de *bugs*, estes foram resolvidos imediatamente após serem detetados, num curto espaço de tempo, e marcados como resolvidos na plataforma *Asana*.

5.1. Resultado final com interface melhorada

Após a realização dos testes e da correção dos problemas identificados, é possível apresentar agora o aspeto final das interfaces principais da plataforma.

É importante referir que apenas os *mockups* apresentados no capítulo *Mockups* da plataforma constituem o trabalho do investigador, sendo que as seguintes interfaces foram desenvolvidas com base nesses mesmos *mockups* por parte da equipa de *designers* da empresa.

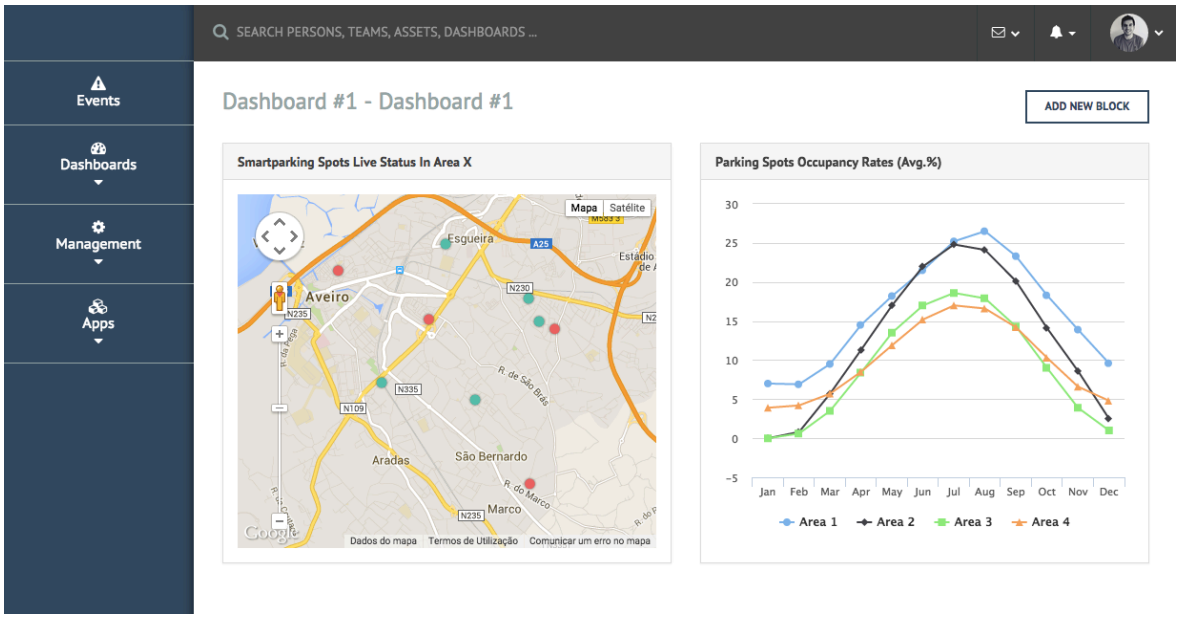


Imagem 30 | Aspeto final da interface de um *dashboard*

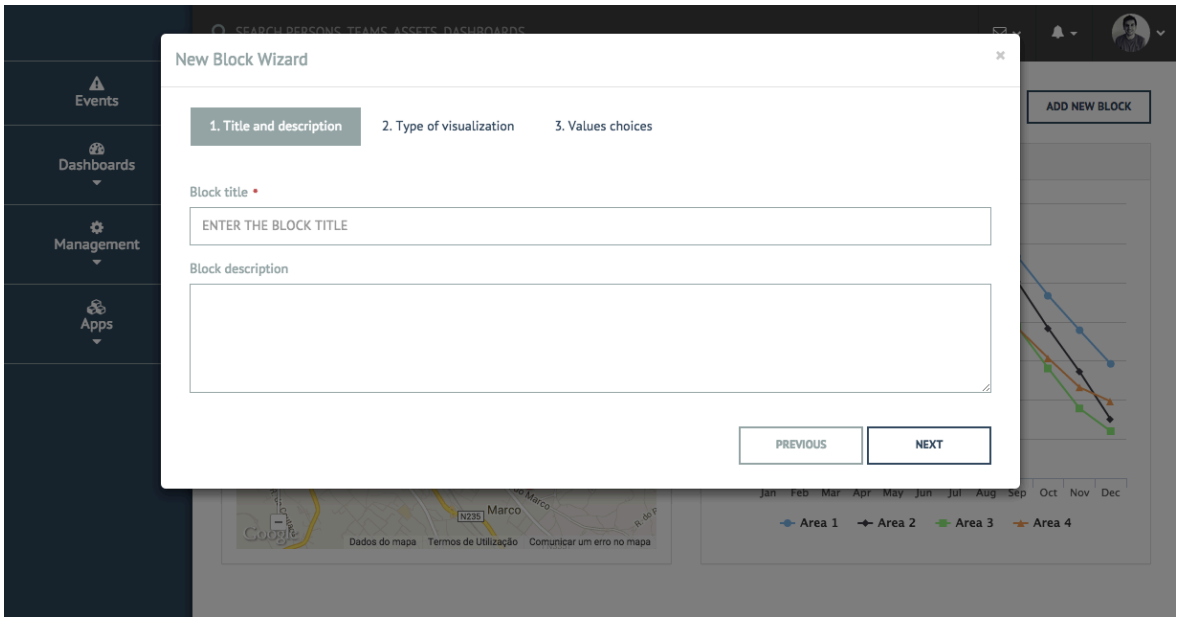


Imagem 31 | Aspeto final do primeiro passo da interface da ferramenta de criação de blocos para os *dashboards*

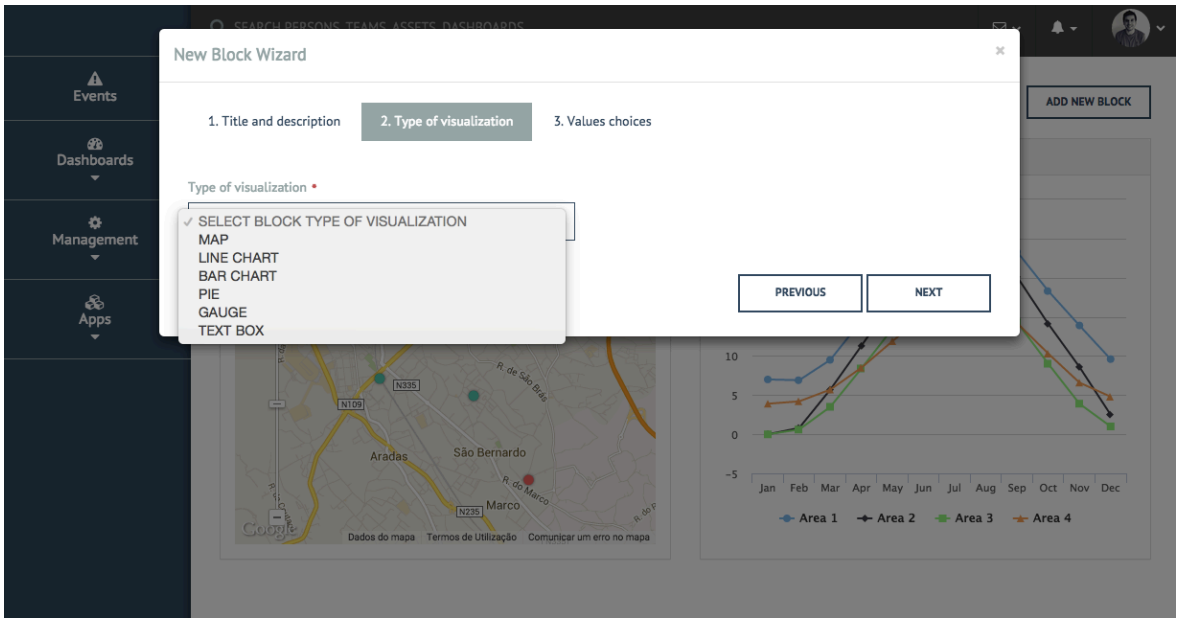


Imagem 32 | Aspeto final do segundo passo da interface da ferramenta de criação de blocos para os *dashboards*

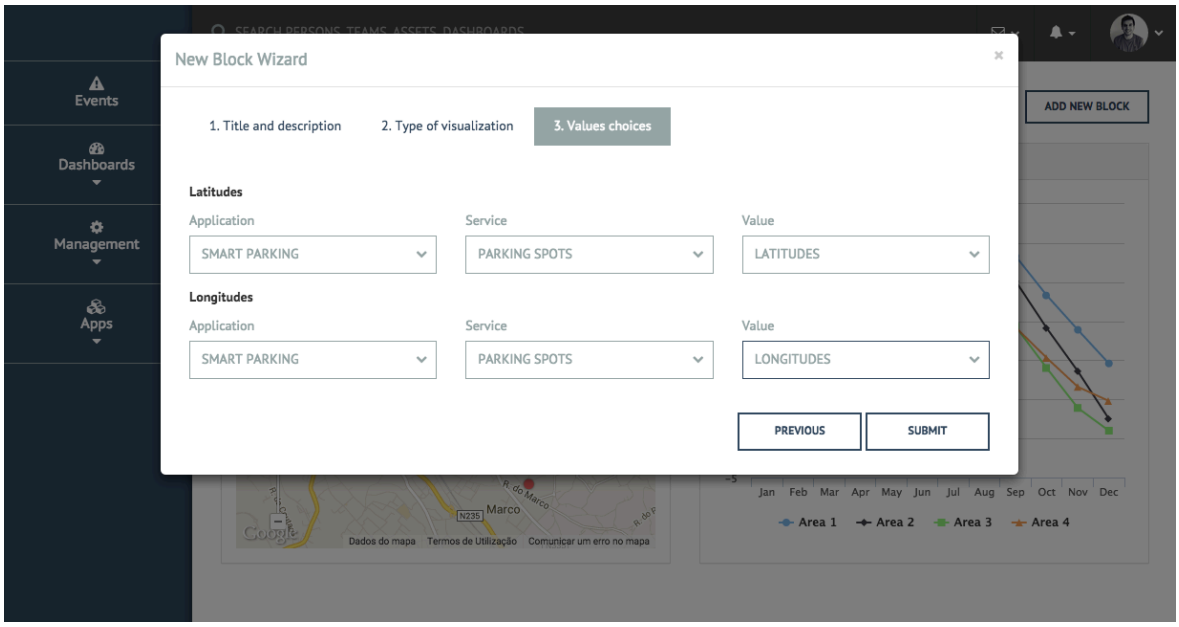


Imagem 33 | Aspeto final da interface do último passo da ferramenta de criação de blocos para os *dashboards*

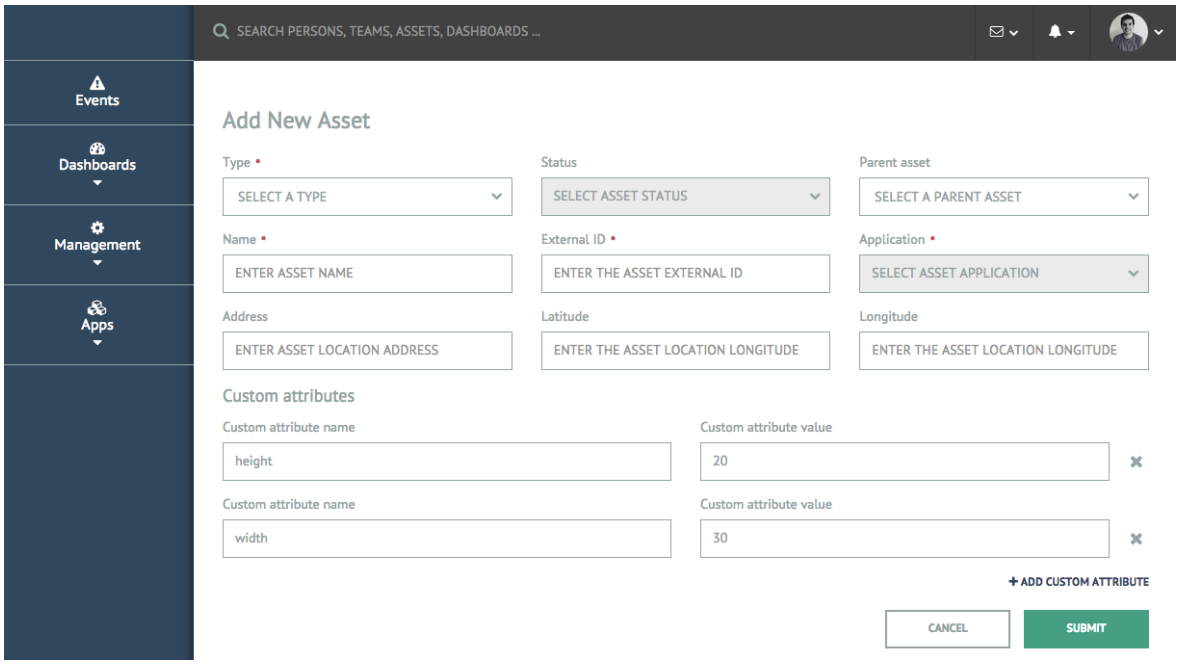


Imagem 34 | Aspeto final da interface de criação de *assets*

6. Conclusões

Existe um número cada vez maior de pessoas a optar por viver em centros urbanos, o que causa a diminuição na capacidade das entidades governadoras de prestarem os serviços básicos aos seus habitantes, como por exemplo, a recolha do lixo ou a gestão do tráfego rodoviário. Este facto abre as portas para o desenvolvimento de aplicações na área das TICs que sejam capazes de ajudar os governos a manterem a qualidade e eficácia na prestação destes serviços.

Esta investigação, em parceria com a empresa Ubiwhere, insere-se nesta problemática e teve como objetivo o desenvolvimento de uma plataforma para apoiar as entidades governadoras na gestão das suas cidades. Ao fundamentar este desenvolvimento e a justificar cada uma das escolhas realizadas ao longo do seu percurso, foram criadas um conjunto de linhas orientadoras (*guidelines*) que podem servir de auxílio para o desenvolvimento de outras plataformas idênticas: o facto de nesta investigação ser explicada em pormenor a arquitetura (lógica e física) da plataforma, todos os módulos que a constituem, como se relacionam e que papel desempenham, as tecnologias utilizadas, a justificação das suas escolhas e até pormenores mais técnicos de desenvolvimento, constitui material para que outros investigadores ou *developers* o considerem no desenvolvimento dos seus estudos e/ou plataformas.

Numa plataforma com as características semelhantes à aqui desenvolvida, a escalabilidade é crucial para que esta se adapte e consiga receber qualquer tipo de dados proveniente de qualquer tipo de sensor, e consiga apresentar a informação da forma mais adequada ao utilizador. É portanto fundamental referir que, de entre todas as funcionalidades, tecnologias e características que esta plataforma foi adquirindo, há três que têm uma importância e valor acrescido como resultado desta investigação: a ferramenta de criação de *dashboards,* o conceito de *assets* e os atributos customizados, e o conceito das aplicações externas e das ligações que estas podem estabelecer com o *"control center*".

Estas três funcionalidades/conceitos, em conjunto, permitem ao sistema ter capacidade de se adaptar a uma elevada quantidade de casos de uso, que é o

pretendido para uma plataforma que tem como o objetivo ajudar na gestão de um centro urbano onde são inúmeros os aspetos passíveis de serem monitorizados.

O facto desta investigação ter sido realizada em contexto empresarial foi também uma mais-valia para o investigador. O facto de estar num ambiente profissional, rodeado por outros membros do projeto, com o apoio constante destes, e seguindo as metodologias de desenvolvimento *"agile"* adotadas pela empresa, tornou o processo de desenvolvimento muito mais eficiente e eficaz. A fase de testes foi também bastante agilizada com a colaboração da empresa, nomeadamente de um colaborador especializado em realizar testes em plataformas *web.*

É importante, por fim, confrontar a pergunta de investigação inicialmente colocada: "Que aspetos tecnológicos considerar no desenvolvimento de uma plataforma *web* de gestão de uma cidade inteligente?". Nesta fase final da investigação é possível constatar que esses aspetos foram apurados e descritos ao longo desta investigação, não só através das escolhas das tecnologias feitas e das respetivas justificações, das arquiteturas lógica e física que foram apresentadas, das técnicas de desenvolvimento explicadas, mas sobretudo dos três conceitos descritos anteriormente, constituindo estes os aspetos diferenciadores e que maior valor dão à plataforma desenvolvida nesta investigação.

Finalmente, é também importante referir que foi submetido um *paper* baseado nesta investigação para ser publicado e apresentado na conferência CENTERIS 2015 – *Conference on ENTERprise Information Systems*, sendo que este foi avaliado e aceite pelos revisores. O *paper* tem o título "*Smart City management platform*" e está disponível no Anexo 1.

7. Bibliografia

Arrowsmith, L. (2014). Smart Cities to Rise Fourfold in Number from 2013 to 2025. Retrieved December 17, 2014, from https://technology.ihs.com/507030/smart-cities-to-rise-fourfold-in-number-from-2013-to-2025

Asana. (2015). Asana · Teamwork without email. Retrieved July 2, 2015, from https://asana.com/

Ashton, K. (2009). That “Internet of Things” Thing. *RFiD Journal*. Retrieved from http://www.itrco.jp/libraries/RFIDjournal-That Internet of Things Thing.pdf\npapers3://publication/uuid/8191C095-0D90-4A17-86B0-550F2F2A6745

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Agile Manifesto. Retrieved from http://agilemanifesto.org/

Berners-Lee, T. (1998). Cool URIs don’t change. Retrieved May 24, 2015, from http://www.w3.org/Provider/Style/URI

Bhatta, B. (2010). Analysis of Urban Growth and Sprawl from Remote Sensing Data. In *Analysis of Urban Growth and Sprawl from Remote Sensing Data* (pp. 17–37). doi:10.1007/978-3-642-05299-6

Bootstrap. (2015). Bootstrap · The world’s most popular mobile-first and responsive front-end framework. Retrieved May 30, 2015, from http://getbootstrap.com/

Bower. (2015). Bower - A package manager for the web. Retrieved May 30, 2015, from http://bower.io/

Brehm, S. (2013). Isomorphic JavaScript: The Future of Web Apps. Retrieved May 7, 2015, from http://nerds.airbnb.com/isomorphic-javascript-future-web-apps/

Caragliu, A., Del Bo, C., & Nijkamp, P. (2011). Smart Cities in Europe. *Journal of Urban Technology*. doi:10.1080/10630732.2011.601117

Chen, M. (2013). Towards smart city: M2M communications with software agent intelligence. *Multimedia Tools and Applications*, *67*, 167–178. doi:10.1007/s11042-012-1013-4

Coutinho, C. P. (2011). *Metodologia de Investigação em Ciências Sociais e Humanas*. Almedina. Retrieved from http://www.almedina.net/catalog/product_info.php?products_id=14814

DailyJS. (2015). DailyJS. Retrieved June 3, 2015, from http://dailyjs.com/

Department of Economic and Social Affairs. (2014). *World Urbanization Prospects: The 2014 Revision Highlights*. United Nations Pubns.

Django Software Foundation. (2015). The Web framework for perfectionists with deadlines | Django. Retrieved January 25, 2015, from https://www.djangoproject.com/

Font Awesome. (2015). Font Awesome, the iconic font and CSS toolkit. Retrieved May 30, 2015, from http://fortawesome.github.io/Font-Awesome/

Freeboard. (2014). freeboard - Dashboards For the Internet Of Things. Retrieved December 17, 2014, from https://freeboard.io/

Git. (2015). Git --everything-is-local. Retrieved May 30, 2015, from https://git-scm.com/

GitHub. (2015). GitHub. Retrieved from https://github.com

Grunt. (2015). Grunt: The JavaScript Task Runner. Retrieved May 30, 2015, from http://gruntjs.com/

Hernández-Muñoz, J. M., Vercher, J. B., Muñoz, L., Galache, J. A., Presser, M., Hernández Gómez, L. A., & Pettersson, J. (2011). Smart cities at the forefront of the future internet. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6656*, 447–462. doi:10.1007/978-3-642-20898-0_32

Hollands, R. G. (2008). Will the real smart city please stand up? Retrieved from http://www.tandfonline.com/doi/abs/10.1080/13604810802479126#preview

Knight, K. (2011). Responsive Web Design: What It Is and How To Use It. Retrieved May 30, 2015, from http://www.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/

Lake, C. (2013). 24 beautifully-designed web dashboards that data geeks will love | Econsultancy. Retrieved July 6, 2015, from https://econsultancy.com/blog/62844-24-beautifully-designed-web-dashboards-that-data-geeks-will-love/

Lane, K. (2015). API 101. Retrieved May 6, 2015, from http://101.apievangelist.com/

M2M Evolution. (2014). Battle of the Platforms | M2M Evolution. Retrieved December 17, 2014, from http://www.m2mevolution.com/conference/battleoftheplatforms.aspx

Makai, M. (2015). Django - Full Stack Python. Retrieved May 19, 2015, from http://www.fullstackpython.com/django.html

Mark, J. (2011). Agile Web Development That Works. Retrieved June 2, 2015, from http://sixrevisions.com/web-development/agile/

Microsoft Developer Network. (2015). Scalability. Retrieved April 22, 2015, from https://msdn.microsoft.com/en-us/library/aa292172%28v=vs.71%29.aspx

Mozilla Developer Network. (2015a). Introduction to HTML. Retrieved January 24, 2015, from https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Introduction

Mozilla Developer Network. (2015b). JavaScript | MDN. Retrieved January 25, 2015, from https://developer.mozilla.org/pt-PT/docs/Web/JavaScript

Offutt, J. (2002). Quality attributes of Web software applications. *IEEE Software*, *19*, 25–32. doi:10.1109/52.991329

Oliveira, L. (2006). Metodologia do desenvolvimento: um estudo de criação de um ambiente de e-learning para o ensino presencial universitário. *Unisinos*, *10*, 69–77. Retrieved from http://repositorium.sdum.uminho.pt/handle/1822/8129

Palani, G. S. (2011, March 29). Summary of web application testing methodologies and tools. Retrieved from http://www.ibm.com/developerworks/library/wa-webapptesting/

Pyhton Software Foundation. (2015). About Python™ | Python.org. Retrieved January 25, 2015, from https://www.python.org/about/

Quivy, R., & Campenhoudt, L. Van. (1998a). Manual de investigação em ciências sociais. *Vasa*, 1–34. Retrieved from http://medcontent.metapress.com/index/A65RM03P4874243N.pdf\nhttp://www.fep.up.pt/docentes/joao/material/manualinvestig.pdf

Quivy, R., & Campenhoudt, L. Van. (1998b). Manual de investigação em ciências sociais. *Vasa*, 1–34.

RacoWireless. (2014). Omega Management Suite. Retrieved December 17, 2014, from http://www.racowireless.com/products/omega-management-suite/

Richey, R. C., Klein, J. D., & Nelson, W. A. (2004). Developmental research: Studies of instructional deisgn and development. In *Handbook of Research for Educational Communications and Technology* (pp. 1099–1130).

Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., & Oliveira, A. (2011). Smart cities and the future internet: Towards cooperation frameworks for open innovation. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6656*, 431–446. doi:10.1007/978-3-642-20898-0_31

Sierra Wireless. (2014). Sierra Wireless - AirVantage Management Service (AVMS). Retrieved December 18, 2014, from http://www.sierrawireless.com/productsandservices/AirVantage_M2M_Cloud/Management_Service.aspx

Skillcrush. (2012). Frontend vs. Backend | Skillcrush. Retrieved January 23, 2015, from http://skillcrush.com/2012/04/17/frontend-vs-backend-3/

Smashing Magazine. (2015). Smashing Magazine – For Professional Web Designers and Developers. Retrieved June 3, 2015, from http://www.smashingmagazine.com/

Stackshare Inc. (2015). Discover and discuss the best dev tools and cloud infrastructure services | StackShare. Retrieved June 3, 2015, from http://stackshare.io/

The core less team. (2015). Less.js. Retrieved May 24, 2015, from http://lesscss.org/

The jQuery Foundation. (2015). jQuery. Retrieved May 24, 2015, from https://jquery.com/

Tock, N. (2013). Using LESS with WordPress. Retrieved May 24, 2015, from http://www.noeltock.com/web-design/wordpress/using-less-with-wordpress/

W3C. (2015). HTML & CSS. Retrieved January 24, 2015, from http://www.w3.org/standards/webdesign/htmlcss

Walsh, D. (2015). David Walsh - JavaScript, HTML5 Consultant. Retrieved June 3, 2015, from http://davidwalsh.name/

Watson, D. S., Piette, M. A., Sezgen, O., Motegi, N., & ten Hope, L. (2004). Machine to Machine (M2M) Technology in Demand Responsive Commercial Buildings. Pacific Grove, CA. Retrieved from http://escholarship.org/uc/item/3286r367#page-4

Wilton-Jones, M. (2011). Introduction to JavaScript. Retrieved January 25, 2015, from http://www.howtocreate.co.uk/tutorials/javascript/introduction

Xively. (2015a). About Xively. Retrieved December 17, 2014, from https://xively.com/about/

Xively. (2015b). Awards - Xively. Retrieved December 17, 2014, from https://xively.com/awards/

8. Anexos

Anexo 1 – *Papper* submetido para a conferencia *"CENTERIS – Conference on ENTERprise Information Systems"*

Conference on ENTERprise Information Systems / International Conference on Project MANagement / Conference on Health and Social Care Information Systems and Technologies, CENTERIS / ProjMAN / HCist 2015 October 7-9, 2015

Smart city management platform

First Author[a], Second Author[b], Third Author[a,b,]

[a]*First affiliation, Address, City and Postcode, Country*
[b]*Second affiliation, Address, City and Postcode, Country*

Abstract

At this moment, 54% of the whole global population lives in large urban centers, and the trend is to increase. Alongside with this growth of urban population, emerges a decrease in the efficiency providing the basic public services to the population, like transportation, education, security, sanitation, by the responsible entities. The current state of the communication technologies allows the possibility to restructure the infrastructures that provide the basic services to the population, increasing its efficiency and sustainable, leading to the concept of *smart city* – though the installation of sensors and actuators, it is possible to collect real time information about different actuation areas, like the occupancy rate of all the car parking spots in the city or how filled are the public waste containers in certain street, allowing the responsible entities to make conscious and grounded decisions. Therefore, it is needed a platform where one can intuitively visualize and analyze all this information. This investigation pretends to study the development process and associated technologies of such a platform, in order to obtain a function high fidelity prototype and a set of guidelines to develop identical platforms.

Peer-review under responsibility of SciKA - Association for Promotion and Dissemination of Scientific Knowledge.

Keywords: Smart city, web platform, developent guidelines, management, control, dashboard.

1. Introduction

In the past few years, there has been witnessing an increase of the percentage of world population living in urban areas. In 2014, more then a half of the world population (54%) lives in urban areas, and it is expected this number to rise to 66% by 2050[1].

As the urban population increases, the capacity of the responsible entities to provide public services to the population, such as sanitation, security and transportation, starts to decrease[2]. A restructure of the infrastructures that provide these public services is needed in order for them to become more effective, efficient and sustainable for the responsible entities.

The information and communication technologies (ICTs) may play a very important role in this restructuration when used to aid the public service, being this called as "smart solutions"[3].

According to several investigations, a city is considered "smart" when, in its essence, uses the ICTs to develop an infrastructure to provide the public services to the population[4]. Therefore, we can today witness several use cases of the ICTs in urban centers and its benefits: in the transportation area, sensors are used to control the traffic congestion, control the parking spots occupancy, and to manage the public transportation in general; in the waste management area, the ICTs may be useful to check where the trash needs to be collected in the first place, to establish priorities and allocate human resources when collecting it; the public security

may also benefit from the ICTs, with the use of motion capture cameras, for example[4]. These are only some of the many examples where the ICTs may play a very important role when it comes to manage and control an urban center.

Schaffers et al.[5] points three fundamental steps for a city to become smart (or intelligent): in the first place, a city needs to assure the existence of the necessary communication infrastructures (e.g. optical fiber, wireless networks), making possible the connectivity between the different persons and devices; secondly, it is necessary the strategic setup of sensors and actuators to measure important information in real time about what's happening in the city; lastly, the development of platforms and applications that allow the treatment and visualization of the information captured by the sensors. This investigation focus on this last step and aims to develop a web platform for the city responsible entities to access and see in real time information captured by the sensors and actuators spread by the urban center, so that conscious and grounded decisions could be made in order to increase the efficiency and sustainability providing public services to the population. After the prototype is developed, performing both usability and functionality tests are also part of this investigation objectives.

2. State of the art

In this stage, it is important to analyze similar platforms to the platform this investigation is about. In this chapter it is briefly explained some of the platforms currently on the market and its key functionalities.

2.1. Xively by LogMeIn

One of the biggest problems when developing platforms regarding the *Internet of Things* is diversity of sensors, actuators, types of information, connections and so on. It is important to establish common protocols, procedures and models to ensure, in a certain way, the standardization of all the information, making it easier to build tools like the one this investigation is aiming to develop.

Xively offers an integrated solution that has the objective to solve the problem described above, with a set of tools and services, grouped on a framework called "*Developer Workbench"*. This framework is an on-line dashboard that allows the management, control and visualization of the different data from the *Internet of Things*[6], in a way that it is possible to control everything inside this platform despite of the differences between the types of sensors and actuators. *Xively* is a trendy solution for this type of problems and will be an important component of the solution developed under this research.

2.2. Omega Management Suite by RacoWireless

Similar to the *Xively* platform, the *Omega Management Suit* tackles the problem related to the diversity of data in the *Internet of Things*, offering a web platform that consists on customizable dashboards where the user can visualize in different ways the information captured in real time. This platform has an interesting and innovating design and user experience, and is ready to be used not only through a desktop computer, but also by tablets and smartphones7.

This solution has, as key functionalities, the usage alerts, which consist in notifications to the users of the platform when certain predefined thresholds values has been reached, and the detailed users control and management, that allows the creation of different roles and access rights and assign those to different user accounts[7].

2.3. *Freeboard*

This platform consists in a website that allows the creation of customized dashboards. *Freeboard* is far simpler than the *Xively* and the *Omega Management Suit* platforms: as it slogan is "*Ridiculously simple dashboards for your devices"* [8], the *Freeboard* allows the users to create their own *Internet of Things* dashboards just in minutes with an intuitive user experience and design, throughout a simple interaction paradigm like drag & drop. Although, *Freeboard* is much more limited and doesn't allow to perform complex tasks like the *Xively* or the *Omega Management Suit* do.

The research team also aims to achieve this simplicity in the platform that will be developed, so it became important to study *Freeboard* in order to understand how the platform was designed to simplify something that is so complex.

Table 1. Comparison of key characteristics between the different platforms in study

	Platforms		
Key characteristics	*Xively*	*Omega Managemenet Suit*	*Freeboard*
Creation of custom dashboards	x		x
Users management	x	x	
Modern and intuitive interface	x	x	x
Mobile ready	x	x	x
Public development tools (APIs)	x		
Intensive client support	x	x	
Advanced security systems	x	x	
Simple user experience	x	x	x
Free			x

3. The solution

After the study of the current platforms on the market being used to the same purpose, there has been achieved a first idea of what the web platform of this investigation would look like.

It is now important to refer that this investigation is ongoing in partnership with a company, which is specialized in the development of web and mobile solutions, specifically for the *Internet of Things* and smart cities market. The company is making efforts to grow on these markets and already have their own solutions for smart parking and smart waste management.

The platform, herein referred as "control center", was designed as a platform that would "consume" information from the already developed solutions such as the mentioned above. In other words, the control panel would work as the central platform of a smart city, gathering information from different applications and representing it the best way possible to the user. Thus the user will have an overview on the city status through different indicators.

Although the smart cities and the *Internet of Things* platforms tend to be very complex, one of the main focuses of this work was to keep the "control center" architecture as simple as possible. The architecture is represented in *Fig. 1*.

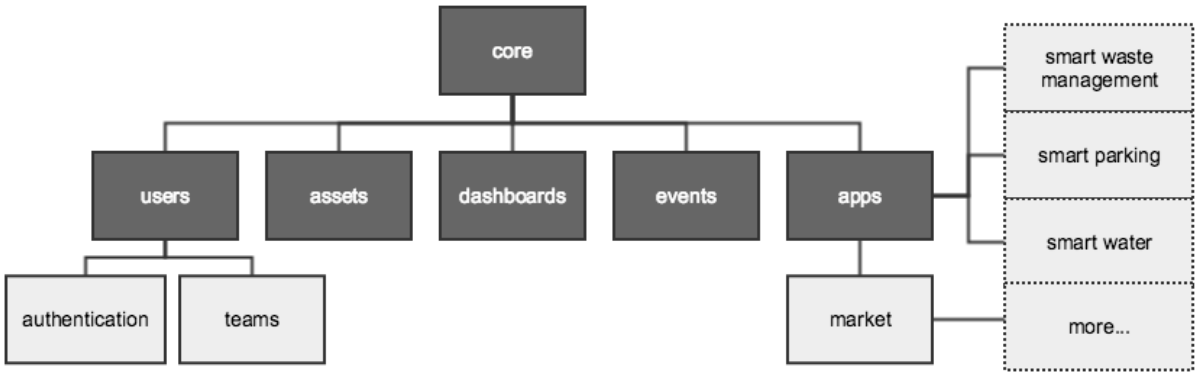


Fig. 1. "Control center" platform architecture

The "control center" is divided into 6 main modules: core, assets, users, dashboards, events and apps (represented as darker boxes in Fig. 1).

The core module is where the base templates, classes and static files (images, scripts and style sheets) that will be used across the platform by all the other modules are placed.

Assets module controls all the assets related processes. An asset represents a "thing" from the *Internet of Things*. It can be a proximity sensor, a humidity sensor, a camera, an RFID tag, etc. This module allows the user to create, edit and delete assets, as well as a list and a details view. As explained before, the main objective of the "control center" is to control all the other specific platforms (smart parking, smart waste, etc.), so when creating an asset in the control panel, the user must specify to which app the asset is related: this protocol is the same for the users' module.

The users' module allows users management: this module allows the creation and edition of users for the "control center" itself or to a specific app. A user can be created with a specific role associated, so that it haves more or less permissions on the platform. For instance, a user can be created with the administrator role and be able to create, edit and delete other users and assets, but a regular user and can only see the detail or list all the users or assets. The user module is subdivided into 2 other modules: the authorization and the team's module. The authorization module handles all the authentication processes of the platform (login and logout methods and interfaces), while the team's module allows the creation of users groups, so that permissions can be better organized (i.e. creating an administrators group with high-level permissions and assigning users to it).

The events module is the smaller module of the entire platform, as it only has the objective to allow the user to see in real time events happening in all the different apps.

The dashboards module is the most important module of the "control center" because it allows the user to create customized dashboards on the control center. A user can have multiple dashboards, and a dashboard can have multiple elements: an element on a dashboard can be a graph, a live map, a gauge, or others elements that could facilitate data visualization. To simplify the creation of dashboards and its elements (or blocks), the user needs to go through a step-by-step creation wizard, where in the first place it is necessary to choose the form of visualization, which can be a line graph, a bar graph, a map, a gauge or a text box, in the second step it's required to choose which parameters from which assets of some app are going to be contemplated, and finally the title of the block. This way, a user can create customized dashboards with different blocks and different types of visualizations, pulling information from the different apps and even mixing this information. For example, it is possible to create a line graph comparing the levels of CO^2 in the air with the number of available parking spots over time in a certain urban area, or even a live map representing in real time which waste containers need to be collected.

Finally, the Applications module handles the connection between the specific smart city applications and the "control center". The user can see a general page for each app that is connected to the "control center", where it is possible to see the basic information about it (it is provided a link to visit the each of the platforms if the user wants to reach specific information). The applications module has also a sub module to handle purchases of more apps available in the solutions marketplace. This is related mostly with the business plan of the company.

A complex web platform such as the one under development in this investigation is divided into two main components: the *front-end* and the *back-end*. In resume, "The frontend is the part of a web site that you can see and interact with, while the backend is all the rest."[9]

This can be compared to an iceberg (Fig. 2), in which the visible part of it represents the *front-end* and the part under water, which is transparent to the user, represents the *back-end* side. Each of these parts has core technologies, tools, technologies and concepts associated that are crucial to each development of each part.

Fig. 2. Analogy between an iceberg and the *front-end* and *back-end* parts of a web platform (Source: *Skillcrunch*)

In this investigation, the investigator's mission is to develop the *front-end* component of the "control center", since other company developers are developing the *back-end* component.

4. Next steps

At the time of this writing, the platform has its core, assets, users and events module fully developed and working integrated with the *back-end* side. The dashboards and apps modules are still under development. As next steps for this investigation, it is need to develop one of the main functionalities of the platform, and the most challenging one: a mechanism to allow the creation of the customized blocks on the dashboard. This is a complex task, because the platform has to handle the possibility to create customized dashboards mixing different types of information, and displaying it in different types of visualization screens. Due to the fact that information available from the different apps might be really diverse and with different degrees of complexity, the mechanism that will allow the creation of these dashboards must be ready to handle multiple scenarios.

After the development part of the project is fully done, it will be time to perform functionality and usability tests. These tests will be handled by the other employees in the company, and will give feedback to the investigator about possible functionality or usability issues in the platform. The functionality tests are expected to see if the platform reacts the right way to the actions performed by the users (i.e. if an asset is created the right way using the platform asset creation tool), and the usability tests are expected to give feedback about the platform interface (mainly, if the platform is well organized so it doesn't confuse the users while navigating and trying to perform tasks). It will be also given a list of tasks for the testers to follow and perform to ensure all the main functionalities are tested. The tests results are going to dictate the need or not to do repairs in the platform.

5. Final remarks

Smart cities are an important research area and still under a lot of development. The complexity and quantity of the information that can be captured in an urban center define how complex and confusing might be an application that aims to control and manage a smart city.

This investigation has a main objective to build a platform that allows the control and management of an urban center, in a simple and intuitive way and without limiting the user in terms of what he wants to do or to see. This gets a challenging thing to do as in an urban center there are lots "things" that are worth tracking and controlling and the most obvious case of this objective in this investigation context is the mechanism to create customized dashboards, which is a step-by-step wizard that allows the user to build a dashboard choosing the type of visualization mechanism (a bar, line or pie chart, a map, a gauge, or others) and the information that he wants to track.

It can be said that this investigation sets, in a certain way, a guideline for others who want to develop smart city oriented platforms in terms of: platform's architecture presented and explained; the technologies choice; the mechanism that allows the creation of multiple and customized dashboards in a simpy and user friendly way; These guidelines may help other investigations, and developers who are struggling trying to make a platform for urban center management both simple and functional.

Acknowledgements

Acknowledgements and Reference heading should be left justified, bold, with the first letter capitalized but have

References

1. Department of Economic and Social Affairs. (2014). World Urbanization Prospects: The 2014 Revision Highlights. United Nations Pubns.

2. Bhatta, B. (2010). Analysis of Urban Growth and Sprawl from Remote Sensing Data. In Analysis of Urban Growth and Sprawl from Remote Sensing Data (pp. 17–37). doi:10.1007/978-3-642-05299-6.

3. Caragliu, A., Del Bo, C., & Nijkamp, P. (2011). Smart Cities in Europe. Journal of Urban Technology. doi:10.1080/10630732.2011.601117.

4. Hernández-Muñoz, J. M., Vercher, J. B., Muñoz, L., Galache, J. A., Presser, M., Hernández Gómez, L. A., & Pettersson, J. (2011). Smart cities at the forefront of the future internet. Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6656, 447–462. doi:10.1007/978-3-642-20898-0_32

5. Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., & Oliveira, A. (2011). Smart cities and the future internet: Towards cooperation frameworks for open innovation. Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6656, 431–446. doi:10.1007/978-3-642-20898-0_31

6. Xively. (n.d.). About Xively. Retrieved December 17, 2014, from https://xively.com/about/

7. RacoWireless. (n.d.). Omega Management Suite. Retrieved December 17, 2014, from http://www.racowireless.com/products/omega-management-suite/

8. Freeboard. (n.d.). freeboard - Dashboards For the Internet Of Things. Retrieved December 17, 2014, from https://freeboard.io/

9. Skillcrush. (2012). Frontend vs. Backend | Skillcrush. Retrieved January 23, 2015, from http://skillcrush.com/2012/04/17/frontend-vs-backend-3/

Anexo 2 – Imagem da listagem de problemas da plataforma *"control center"* que foram reportados na plataforma *"Asana"*