Universidade Lusófona de Humanidades e Tecnologias

Licenciatura em Informática de Gestão

Relatório Trabalho fim de curso: WIFI Viewer

Prof. Orientador: Dr. Inês Oliveira

Nome	**Nº**
Luís Ferreira	21002427
José Miguel Veríssimo	21000490

Índice

Índice de Figuras

Resumo

Este relatório detalha o processo de conversão de um protótipo de visualização do padrão de utilização dos *hotspots WiFi* da Universidade Lusófona de Humanidades e Tecnologias realizado em *Flash*, para o dotar de capacidade de adaptação a outros cenários e locais, melhorar a sua flexibilidade, expansibilidade e facilidade de *deployment.* Foi dado enfâse à *forward-compatibility* através da implementação com recurso a tecnologias que permitam a utilização da aplicação em dispositivos móveis e *browsers.* Foram utilizadas as tecnologias *HTML5* e *JavaScript* para alcançar estes objectivos, apoiados por bibliotecas *open-source* de forma a garantir o mínimo de entropias e resistência à sua adopção.

Foi ainda desenvolvido um *Webservice* que devolve os dados da utilização, que serve não só como prova de conceito e suporte para a aplicação mas também porque lhe confere um aspecto muito aproximado da realidade simulando a sua implementação num contexto real

Abstract

This report details the conversion process of the Universidade Lusófona de Humanidades e Tecnologias WiFi hotspot usage visualization prototype, built with Flash, to make the prototype suitable to other sets and scenarios, improve its flexibility, extensibility and ease of deploy. Great emphasis was given to make the prototype mobile and forward compatible, by using HTML5 and Javascript as the technology stack, supported by open source libraries, to reduce the entropy as much as possible and enhance the adoption rate and speed.

During the implementation arose the necessity to build a webservice to serve the usage data, and that is presented as a proof of concept, and application support as well as to approach the development process to a real production scenario.

1. Introdução

O WIFIViewer é um protótipo *Web*, desenvolvido em *Flash* que tem como propósito mostrar o padrão de utilização de *Hotspots WIFI* do campus da Universidade Lusófona de Humanidades e Tecnologias (ULHT). A representação da utilização dos *Hotspots* é feita através da exibição de um mapa do campus da ULHT com os seus edifícios, sobre os quais são dispostos ícones que representam a utilização dos *Hotspots* em cada edifício.

Tecnicamente, a aplicação actual está a apresentar uma imagem estática do campus da ULHT sobre a qual são desenhados os ícones através da utilização de funções de *ActionScript.*

Objectivamente, o protótipo desenvolvido, tendo em conta as tecnologias utilizadas está muito limitado quando à sua adaptação a novas realidades. Ou seja, caso se pretenda aplicar este protótipo a uma outra área (por exemplo, outra faculdade), o código teria de ser refactorizado na totalidade devido, não só ao facto de estar a ser utilizada uma imagem *raster* mas sobretudo devido ao posicionamento dos ícones estar a ser feito sobre a representação estática dos edifícios nessa imagem. Adicionalmente o facto da tecnologia *Flash* não ser suportada por algumas plataformas móveis de referência também torna o protótipo limitado.

Assim, mantendo o propósito original de mostrar o padrão de utilização dos *Hotspots WiFi*, com a versão que vamos desenvolver, o protótipo ficará dotado de capacidade para, através da parametrização dos elementos da área envolvente, dos detalhes arquitectónicos adjacentes e dos edifícios que se pretendem apresentar, desenhar e construir automaticamente um novo "Mapa" de forma dinâmica, totalmente configurável, mantendo um comportamentos padrão para interacções do utilizador e exibição dos dados.

De forma a garantir *forward compatibility* do protótipo, o mesmo será implementado com recurso a tecnologias *bleeding edge, que é conseguido* com o recurso à especificação *HTML5*, em particular com utilização da *Tag Canvas* manipulada através de *JavaScript* sem recurso a imagens o que torna o *payload* de tráfego na rede significativamente mais leve, acelerando o carregamento e acesso à informação. Este

aspecto é uma preocupação neste tipo de sistemaas especialmente relevante quando o acesso é feito sobretudo em plataformas móveis.

Estas alterações irão conferir ao protótipo a capacidade de ser aplicado a outras realidades, com elevado nível de consistência, robustez e sobretudo facilidade de implementação.

Os principais objectivos a que nos propomos nesta nova versão do WiFi Viewer que vamos apelidar de "WiFi Viewer 2" baseiam-se no que a aluna Patrícia Simões identificou no seu relatório como "Futuras Evoluções" e que citamos abaixo:

"Após o desenvolvimento da aplicação e posterior discussão no S-BRAIN surgiram as seguintes sugestões de futuras evoluções:

Conversão para HTML5

Configuração, por exemplo via XML, dos ícones de dispositivos usando um conjunto de imagens externas carregadas dinamicamente. Actualmente é feito de forma estática no ficheiro config.as que referencia símbolos na biblioteca. Esta alteração é fácil de fazer para um programador.

Configuração da imagem do mapa usando uma imagem externa carregada dinamicamente. Actualmente é uma imagem estática. Esta alteração é fácil de fazer para um programador.

Configuração, por exemplo via XML, da localização dos edifícios no mapa. Actualmente é feito de forma estática no ficheiro config.as. Esta alteração é fácil de fazer para um programador.

Além disto, a aplicação actual tem um temporizador que vai ler o ficheiro devices.xml várias vezes (ver config.as) a fim de simular o comportamento entre duas leituras sucessivas. É possível fixar facilmente o intervalo de tempo em que a aplicação vai ler o ficheiro, bem como permitir que esta aceda a mais do que um ficheiro do mesmo tipo, por exemplo: dados da noite e dados do dia. Esta alteração é fácil de fazer para um programador."

No entanto, e como se trata de uma aplicação que se destina a representar informação, será dado um tratamento completamente diferente na área de *data visualization*,

reflectindo algumas perspectivas de análise dos dados da utilização dos *Hotspots*, que nos parecem mais relevantes e interessantes.

2. Enquadramento Teórico

A versão original do protótipo que nos serviu de base para este projecto foi, conforme já referido, desenvolvido com recurso a tecnologias com algumas limitações quanto à compatibilidade entre diferentes plataformas, quanto à sua aplicação a diferentes contextos e quanto à sua facilidade de configuração.

Tendo em conta estas limitações e em função dos objectivos propostos, a abordagem que optamos para o desenvolvimento da solução passou por refactorizar todo o protótipo mantendo o seu propósito original mas dando novas perspectivas de análise aos utilizadores da aplicação.

Toda a aplicação teve que ser repensada para permitir a escalabilidade, a configuração simples e ágil e dotar a aplicação de interactividade.

Assim, foi escolhido um *stack* tecnológico que dá garantias de *forward compatibility*, e foi desenhada uma arquitectura modular que garante a facilidade na expansão do protótipo. Foram também aplicadas técnicas e seguidas recomendações de engenharia de *software* que garantem consistência do código e reflectem as boas práticas da indústria.

Os detalhes de implementação serão explorados em maior detalhe na secção 3.

2.1. Revisão bibliográfica

Neste ponto abordamos com mais algum detalhe as plataformas e tecnologias de referência estudadas e utilizados durante essencialmente a fase de desenvolvimento do projecto.

- **HMTL5** – (Hypertext Markup Language, versão 5) é uma linguagem para estruturação e apresentação de conteúdo para a World Wide Web, e interpretado por todos os *browsers*.

No contexto desta aplicação, para além de *markup* trivial, exploramos a utilização do elemento SVG, introduzido nesta versão.

- **JAVASCRIPT** – JavaScript é uma linguagem de programação interpretada pelo *browser* baseada em ECMAScript pela ECMA international nas especificações ECMA-2623 e ISO/IEC 16262. É uma linguagem orientada a objectos com forte suporte à programação funcional. Baseada em protótipos, possui tipagem fraca e suporte a funções *high-order* e *closures*. 95% deste projecto é realizado nesta linguagem, o que passa a responsabilidade do processamento para o lado do cliente, diminuindo o payload das comunicações entre o servidor e o browser.

- **JSON** – JSON (com a pronúncia ['dʒejzən]), um acrónimo para "JavaScript Object Notation", é um formato leve para transporte de dados. JSON é um subconjunto da notação de objectos de JavaScript, mas a sua simplicidade tem vindo a torná-la mais popular no transporte de dados, especialmente como uma alternativa ao XML. É tipicamente usada em *webservices* onde a fonte dos dados é confiável e previsível, e onde a perda dos recursos de processamento XSLT no lado cliente para manipulação de dados ou geração da interface não é relevante, isto é, onde é a aplicação cliente que trata da camada de apresentação dos dados. Algumas linguagens que suportam JSON incluem ActionScript, C/C++, C#, ColdFusion, Java, JavaScript, OCaml, Perl, PHP, ASP 3.0, Python, Rebol, Ruby, Lua, Progress, entre outras.
Todos os objectos externos ao projecto (*layers* gráficos, configuração, etc...), bem como as chamadas e respostas do serviço são efectuadas em JSON.

- **SVG** – É a abreviatura de Scalable Vector Graphics que pode ser traduzido do inglês como gráficos vectoriais escaláveis. Trata-se de um *subset* que utiliza a linguagem XML para descrever de forma vectorial desenhos e gráficos bidimensionais, quer de forma estática, quer incluindo animação. Umas das principais características dos gráficos vectoriais, é que não perdem qualidade ao serem ampliados. O SVG é *open source* e foi criado pela World Wide Web Consortium, responsável pela definição de outros padrões, como o HTML e o XHTML. SVG é suportado por todos os *browsers* modernos de forma nativa ou através de bibliotecas JavaScript. O suporte nativo no Microsoft Internet Explorer só é possível a partir da versão 9 (antes era utilizado o formato VML para o mesmo efeito).
O projecto utiliza o elemento SVG para todo o conteúdo gráfico, não tendo sido utilizada qualquer imagem *raster*.

2.2. Bibliotecas JavaScript utilizadas

- **jQuery**

jQuery é uma biblioteca javascript leve, *cross-browser* que se identifica pelo mote *"write less, do more"*. Esta biblioteca torna algumas tarefas comuns, mas que requerem muitas linhas de código para implementar, em métodos que podem ser invocados com apenas uma linha de código. Também estão implementados métodos simplificados para chamadas AJAX (asynchronous javascript), e para a manipulação do DOM (Document Object Model). É actualmente a biblioteca javascript mais popular e extensível, e é apoiada pelas maiores *software houses* como a Google e Microsoft, que têm equipas que contribuem para o seu desenvolvimento.

No contexto do nosso projecto o jQuery está a ser utilizado para manipular o DOM. Isto é, permite-nos manipular de forma mais expedita os elementos da aplicação e para as chamadas Ajax, como por exemplo na leitura do serviço que fornece os dados da utilização dos *hotspots WiFi*.

- **Raphael.js**

Raphael.js é uma biblioteca javascript, cross-browser que se foca no desenho vectorial em *websites*. A biblioteca utiliza SVG para a maioria dos *browsers*, mas suporta o formato VML para versões mais antigas do Internet Explorer. É a biblioteca javascript para a manipulação de gráficos vectoriais mais utilizada, e sendo parte do Sencha Labs, está optimizada para a utilização em dispositivos *touch*.

No contexto do nosso projecto esta biblioteca está a ser utilizada para o desenvolvimento dos componentes gráficos da aplicação, uma vez que permite uma abstracção dos ajustes necessários para suportar os principais *browsers*.

- **RequireJS**

O require.js é um sistema de gestão de dependências e de conflito de nomes, mecanismos que não estão presentes nativamente na especificação do Javascript, o que torna a gestão de dependências uma tarefa complexa e passível de introdução de bugs na aplicação.

No contexto do nosso projecto, a biblioteca RequireJS permite-nos gerir as dependências para termos o conceito de *NameSpace/Package* inexistente no JavaScript e a separação em módulos (componentes reutilizáveis), o que tornará mais simples a manutenção e futuras evoluções.

3. Método

Uma vez que se tratava de um projecto com uma envergadura considerável, foi necessário separar o desenvolvimento em fases, até porque dado as condicionantes de prazo era necessário adoptar uma metodologia mais ágil no que concerne a pesquisa e implementação.

Assim foi preciso definir claramente os requisitos para que a implementação fosse a mais célere e eficaz.

Como principais requisitos técnicos consideramos os seguintes:

- Utilização da Linguagem HTML5 e Javascript
- Configuração simples
- Facilmente escalável e adaptável a outros contextos
- Utilização de *standards* da indústria ao invés de formatos proprietários
- Redução do *payload* de dados no carregamento da aplicação
- Utilização de ferramentas *open source* e sem licenciamento

Como principais requisitos funcionais, da nossa análise resultaram os seguintes:

- Compatibilidade da aplicação Web com as principais plataformas mobile (Android/IOS/Windows Phone)
- Interactividade e *User Experience*
- Clareza da informação apresentada

Foram ainda definidos alguns pressupostos para a aplicação

- Acesso e disponibilização de um servidor Web para a aplicação (Apache, nginx, IIS, tomcat, jBoss, entre outros)
- Acesso a um *webservice* que devolve os dados relativos à utilização dos *Hotspots WiFi*, no formato descrito em anexo.

Com base neste levantamento, o desenvolvimento iniciou-se com a análise de várias soluções técnicas para a manipulação do elemento *canvas* do HTML5, tendo a escolha recaído sobre a biblioteca Raphael.js, nomeadamente por abstrair o desenvolvimento dos detalhes de adaptação aos diversos *browsers*, e também por permitir o suporte a *browsers* mais antigos. Neste processo foi ainda decidido, e uma vez que o projecto terá um número considerável de dependências, a utilização da biblioteca Require.js para fazer essa gestão de dependências.

Foi também detectado que seria necessário criar um serviço *dummy* que providenciasse dados para alimentar a aplicação, uma vez que a ULHT não providencia esse serviço. A escolha do *stack* tecnológico a utilizar nesse serviço recaiu sobre a linguagem Ruby, com a *framework* Ruby On Rails, servido via Phusion Passenger como módulo de Apache num servidor Ubuntu a correr numa instância Micro da Amazon Cloud Services. A motor de base de dados utilizado foi o MySQL.

A escolha deste stack tecnológico para o serviço, foi puramente uma questão de agilidade, mantendo a perspectiva dos componentes *open-source*, e ferramentas que permitissem a agilidade necessária. Trata-se de uma componente que não faz parte da especificação do projecto, mas que ainda assim requereu algum esforço de implementação para dar garantias que os dados não seriam totalmente aleatórios, e assim se pudesse observar o resultado final muito próximo do que será ter a aplicação em ambiente de produção.

Foi também necessário providenciar um servidor Web que servisse a aplicação, e a mesma foi colocada numa instância partilhada num servidor Red Hat a correr Apache.

Através destas decisões de projecto pretendeu-se aproximar o desenvolvimento aos ambientes profissionais, e com isso demonstrar a fiabilidade da solução.

3.1. Planeamento e projecto

No desenvolvimento deste projecto, nas suas diversas fases, estiveram envolvidos 3 recursos:

Listagem de Recursos:

- **Gestora Projecto** – Dr. Inês Oliveira

- **Programador Sénior 1** – Miguel Veríssimo
- **Programador Sénior 2** – Luís Ferreira

A participação dos recursos nas diversas actividades inerentes ao desenvolvimento do mesmo estão visíveis no diagrama de Gantt e na tabela de utilização de recursos que colocamos em anexo.

3.2. Arquitectura da Solução

A arquitectura da aplicação é relativamente simples, uma vez que através do recurso ao Javascript, todo o processamento é feito no cliente (*browser*), sendo que com a versão implementada o desenho é o seguinte:

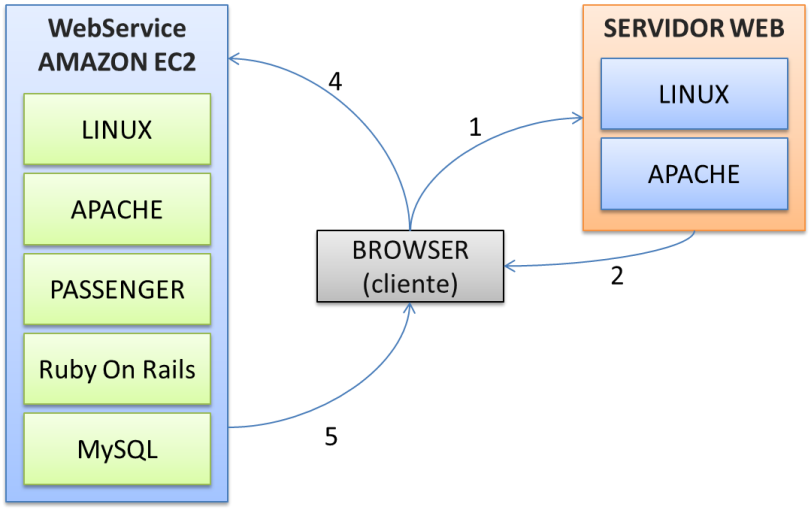


Ilustração 1 - Arquitectura da Solução

3.3. *Workflow* da Aplicação

No sentido de evidenciar e ilustrar o fluxo aplicacional da solução, descreve-se de seguida esse workflow, com referência à ilustração 1.

O fluxo aplicacional é o seguinte:

1. O cliente (*browser*) solicita a página da aplicação com um pedido GET ao URL que aponta para o servidor onde a aplicação está alojada.
2. O servidor responde com a página HTML que contém as referências para os outros componentes da aplicação (ficheiros estáticos CSS, JavaScript e JSON).

3. O cliente repete o passo 1 para cada componente, e o servidor devolve o componente pedido, até toda a aplicação estar carregada.
4. No momento da inicialização da aplicação (após todos os componentes terem sido carregados do servidor Web), é iniciada uma *thread* que a cada minuto faz um pedido ao webservice, dos dados referentes ao último minuto de utilização dos *hotspots WiFi*.
5. O *webservice* responde com os dados solicitados e a aplicação é actualizada.
6. Sempre que o utilizador clica sobre um edifício, ou sobre o componente que mostra o resultado global, é feito um pedido ao servidor (4) sobre a utilização dos *hotspots WiFi* desse edifício (ou o total global) na última semana.
7. O serviço responde com os dados solicitados (5).

3.4. Arquitectura da Aplicação

Sendo uma aplicação essencialmente gráfica, foi necessário considerar as camadas de apresentação (*layers*), para que o desenho seja exibido pela ordem correcta (do mais baixo para o mais alto: terreno, edifícios, aspectos arquitectónicos e paisagísticos, posição das bolhas que indicam a utilização, posição do indicador global). Foi também necessário distinguir se o *layer* é estático ou dinâmico (por exemplo, o terreno é estático, e os edifícios são dinâmicos), através duma propriedade da configuração do *layer*. Também se define a posição do *layer* e o mapeamento com as classes que possuem os métodos específicos para o tipo de objecto.

Assim o ficheiro de configuração possui a informação de cada *layer*, bem como os URLs dos serviços de dados.

A aplicação é composta de diversas classes, como se pode observar no *source code* em anexo, mas podemos destacar algumas classes:

- **Config.js** – É onde se detalha a configuração da aplicação.
- **App.js** – É onde se carregam os *layers* e onde se instanciam os objectos que compõem a aplicação, bem como onde se inicia a *thread* que actualiza os dados a cada minuto.
- **MultiLevelBuilding.js** – É a classe mais complexa e *core* da aplicação, uma vez que os edifícios são desenhados isometricamente a partir de dados como comprimento, largura e número de pisos. A cada objecto desta classe é ainda *binded* um objecto

(WifiHotspotAccessData.js) que é responsável por pedir, receber, tratar e apresentar os dados relativos à utilização dos *hotspots WiFi* desse edifício, na última semana.

- **WifiHotspotAccessData.js** – É a classe responsável por invocar o serviço de dados referente à última semana de utilização dos *hotspots WiFi* para o contexto em que foi invocada, e de tratar e apresentar esses dados ao utilizador.
- **BuildingStatsPresenter.js** – É a classe responsável por desenhar o balão que detalha a utilização dos hotspots WiFi no último minuto, para um edifício específico.

3.5. Detalhes de implementação relevantes

Dada a natureza *Single-threaded* do *JavaScript* e ao considerável número de *assets* que têm que ser carregados no arranque da aplicação, optámos por fazer os *requests* desses *assets* de forma assíncrona, recorrendo ao mecanismo de *promises* da biblioteca jQuery de forma a garantir que todos os *assets* são carregados, e que a aplicação apenas "arranca" quando os dados estiverem do lado do cliente.

Foi necessário declarar um objecto que guarda alguma informação de estado, uma vez que cada entidade é responsável pela sua parte do processamento e desenho. Ao invés do fluxo imperativo normalmente utilizado onde o programador controla o estado da aplicação e o *chaining* de eventos, numa implementação que recorre a métodos assíncronos é necessário gerir de forma cautelosa e inteligente os *callbacks* das funções para prevenir dessincronização e estados inválidos. Para o efeito cada função de desenho dos processos que correm em *background*, verifica se há algum elemento a bloquear o interface, e assim desenha (ou não) as suas componentes.

Houve particular atenção para a implementação correcta dos objectos, de forma a eliminar o código dos métodos da definição do objecto, e adicioná-los ao *protoype*, o que aumentou a complexidade com a gestão do acesso das funções de *callback* devido aos mecanismos de *closure* e *high-order function* do Javascript.

4. Resultados

O resultado do trabalho foi uma aplicação quem cumpre sigilosamente os requisitos definidos, adicionando dimensões de visualização dos dados que não estavam presentes

no protótipo, e que na nossa opinião, tornam a sua utilização muito mais interessante e útil.

Do ponto de vista gráfico, procurámos ir ao encontro, com o maior detalhe possível, da representação do campus que já existia no *website* da ULHT:

Ilustração 2 - Interface da aplicação

Para evitar o ruído visual, introduziu-se um mecanismo que mostra ou oculta a informação da distribuição da utilização, que pode ser alternado clicando em "*show distribution*":

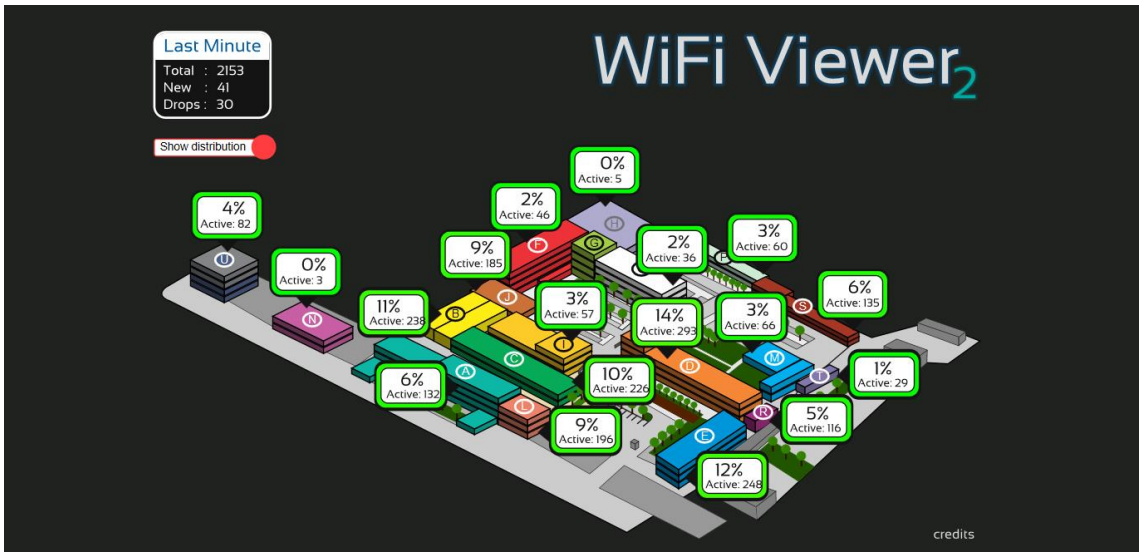


Ilustração 3 - Interface da aplicação - distribuição da utilização dos hotspots activa

Foi ainda introduzida a possibilidade de visualizar o detalhe da utilização (global ou por edifício) durante a última semana:

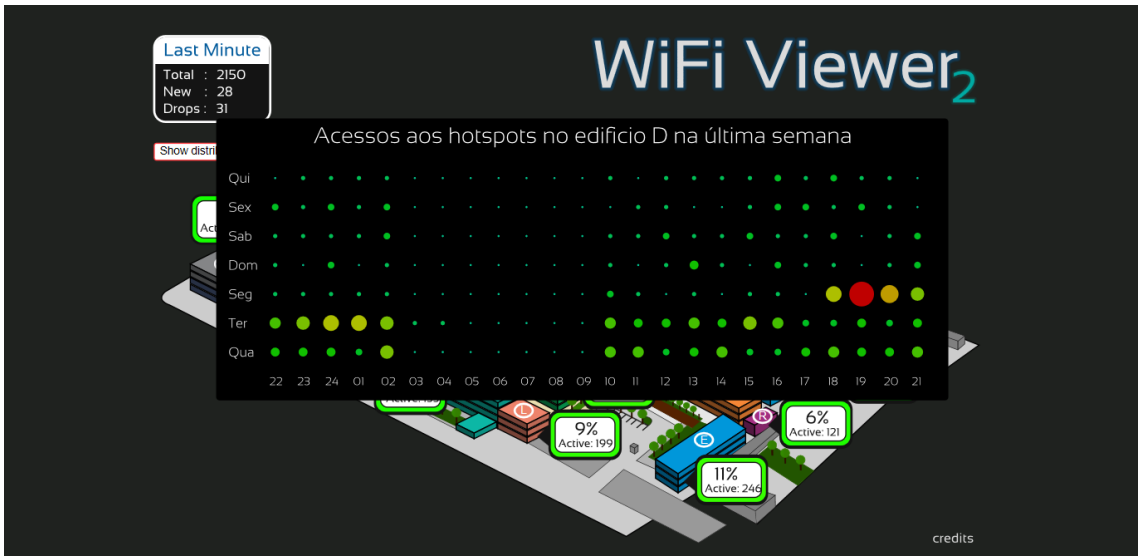


Ilustração 4 – Interface da apliacação – Detalhe da utilização dos hotspots no edifício D, na última semana

A aplicação é *responsiva*, o que significa que se ajusta ao tamanho do ecrã, o que facilita a sua utilização em qualquer dispositivo, embora seja necessário um tamanho mínimo de 740x496 px, abaixo do qual os objectos não serão escalados porque se perderia a legibilidade.

Toda a implementação foi no sentido de garantir que se estavam a utilizar as técnicas e os processos mais adequados às necessidades listadas, de forma a tornar a solução robusta o suficiente para entrar em produção, condição essa que julgamos estar cumprida.

5. Conclusões e Trabalho Futuro

Os desafios introduzidos com este projecto foram de um nível de exigência considerável tendo em conta todo o *know-how* técnico necessário para o efectuar com rigor, cumprindo todos os requisitos e objectivos propostos. Foi necessário mais dedicação em termos de tempo do que que aquilo que expectávamos inicialmente devido essencialmente à utilização de linguagens de programação e conceitos que não foram abordados no curso da licenciatura, e dado o contexto muito específico da aplicação. Para adquirir conhecimento nestes ambientes foi muitas das vezes necessário testar, desenvolver e implementar pequenas *features*/*samples* para reduzir ao máximo a curva de aprendizagem. A partir da altura que os conceitos básicos da linguagem foram adquiridos, todo o projecto correu como esperado. Durante o desenvolvimento do projecto foram desenvolvidos conhecimentos técnicos, quer a nível teórico. Quer por

exploração de tecnologias com a qual ainda não tínhamos tido contacto, o que evidencia as competências adquiridas e demonstradas durante todo o curso.

Como trabalho futuro, identificamos cinco pontos:

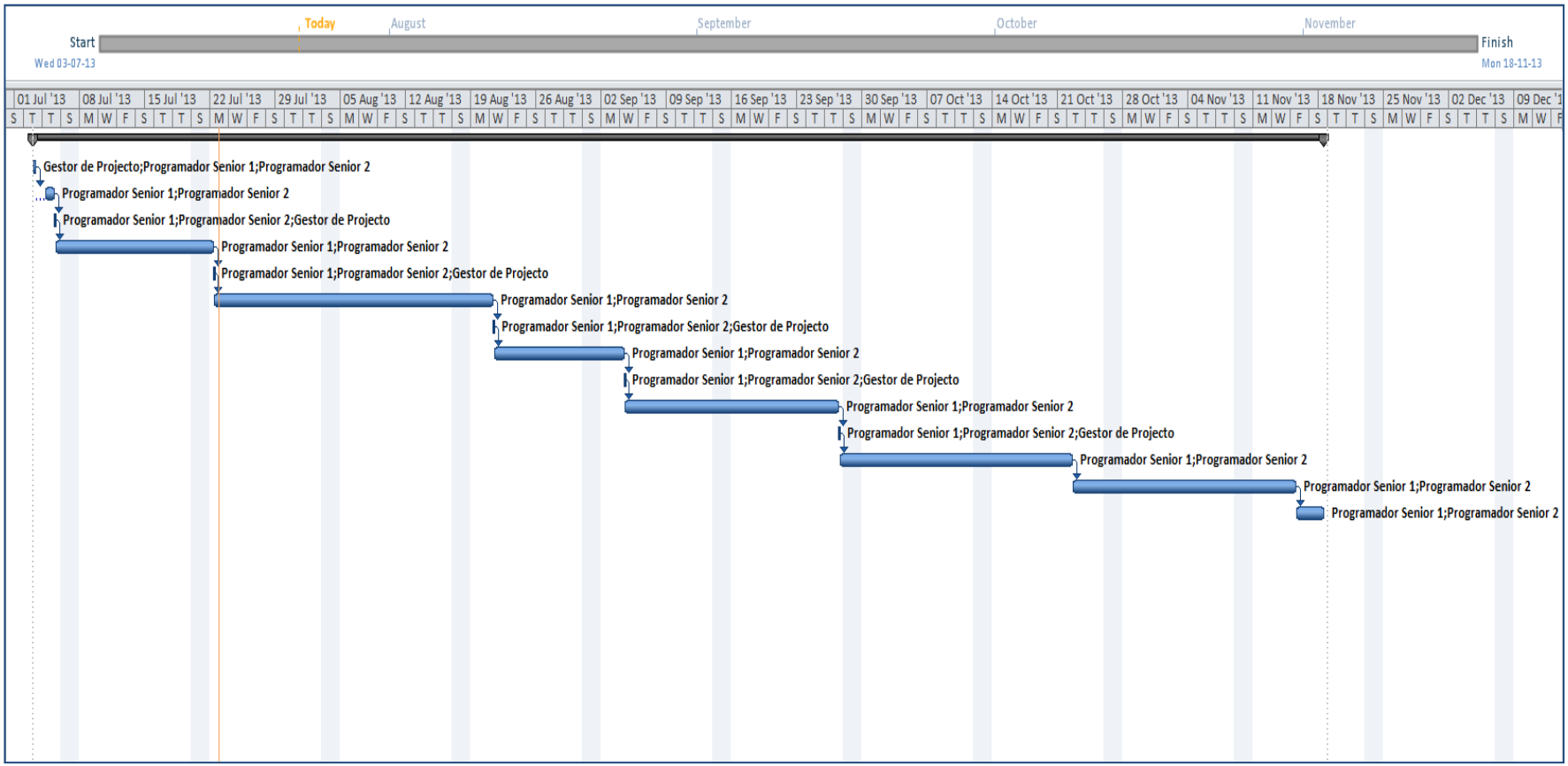
- *Minificação* do código para diminuir o número de *requests* ao servidor. Trata-se de comprimir e unir os diversos módulos num só ficheiro, preservando a sua definição. Esta tarefa não foi executada porque iria retirar a legibilidade ao código, e seria contraproducente no contexto académico.
- Internacionalização (i8n). Implica a extracção de todas as *strings*, e a sua substituição por referências para chaves de um *hashmap*, em que os valores serão alterados em *runtime* através de um novo módulo de selecção da língua. Este módulo terá que invocar todos os módulos de desenho e forçar que os mesmos voltem a desenhar com as novas *strings*.
- Melhor controlo de erros, nomeadamente na eventualidade da falha na resposta do *webservice*. Esta tarefa terá que ser detalhada quando houver um *webservice* de produção, para acautelar quais os melhores mecanismos de *fallback*.
- Criação de objectos que mapeiam respostas de serviços diferentes para as representações gráficas que existem. Esta funcionalidade permitiria separar a apresentação da recolha e tratamento dos dados.
- Criação de um editor de mapas. Esta tarefa não é menor em complexidade do que o actual projecto. O editor teria que utilizar as mesmas definições de objecto existentes na aplicação, e permitir ao utilizador desenhar um contorno de edifício e posteriormente caracterizar o número de pisos, a ordem de desenho, o *layer*, bem como outros detalhes (bolha informativa, cores, etc...). Esta tarefa daria um trabalho de fim de curso interessante e que complementaria o actual projecto.

Bibliografia

- jQuery API : http://api.jquery.com/
- Raphael.js Reference: http://raphaeljs.com/reference.html
- Wrangle Async Tasks With jQuery Promises: http://net.tutsplus.com/tutorials/javascript-ajax/wrangle-async-tasks-with-jquery-promises/
- JSON specification: http://www.json.org/
- JSON versus JSONP Tutorial – Live: http://json-jsonp-tutorial.craic.com/index.html

Anexos

1. Diagrama de Gantt



2. Tabela de utilização de recursos

Task Name	Duration	Start	Finish	Predecessors	Resource Names	Cost
<TFC - WIFI Viewer>	**98,75 days**	**Wed 03-07-13**	**Mon 18-11-13**			**23.580,00 €**
Reunião KickOff Projecto	1 hr	Wed 03-07-13	Wed 03-07-13		Gestor de Projecto;Programador Senior 1;Programador Senior 2	50,00 €
Levantamento e Análise da solução actual	1 day	Wed 03-07-13	Fri 05-07-13	2	Programador Senior 1;Programador Senior 2	240,00 €
Reunião Ponto de Situação	1 hr	Fri 05-07-13	Fri 05-07-13	3	Programador Senior 1;Programador Senior 2;Gestor de Projecto	50,00 €
Levantamento e Análise da Tecnologia a utilizar	11 days	Fri 05-07-13	Mon 22-07-13	4	Programador Senior 1;Programador Senior 2	2.640,00 €
Reunião Ponto de Situação	1 hr	Mon 22-07-13	Mon 22-07-13	5	Programador Senior 1;Programador Senior 2;Gestor de Projecto	50,00 €
Redesenho da aplicação Actual	22 days	Mon 22-07-13	Wed 21-08-13	6	Programador Senior 1;Programador Senior 2	5.280,00 €
Reunião Ponto de Situação	1 hr	Wed 21-08-13	Wed 21-08-13	7	Programador Senior 1;Programador Senior 2;Gestor de Projecto	50,00 €
Desenvolvimento e Testes (Fase 1)	10 days	Wed 21-08-13	Wed 04-09-13	8	Programador Senior 1;Programador Senior 2	2.400,00 €
Reunião Ponto de Situação	1 hr	Wed 04-09-13	Wed 04-09-13	9	Programador Senior 1;Programador Senior 2;Gestor de Projecto	50,00 €
Desenvolvimento e Testes (Fase 2)	17 days	Wed 04-09-13	Fri 27-09-13	10	Programador Senior 1;Programador Senior 2	4.080,00 €

Reunião Ponto de Situação	1 hr	Fri 27-09-13	Fri 27-09-13	11	Programador Senior 1;Programador Senior 2;Gestor de Projecto	50,00 €
Testes com utilizadores	17 days	Fri 27-09-13	Tue 22-10-13	12	Programador Senior 1;Programador Senior 2	4.080,00 €
Elaboração Relatório Projectos	18 days	Tue 22-10-13	Fri 15-11-13	13	Programador Senior 1;Programador Senior 2	4.320,00 €
Entrega do Projecto	1 day	Fri 15-11-13	Mon 18-11-13	14	Programador Senior 1;Programador Senior 2	240,00 €

3. Source code

a. Estrutura de pastas

```
/
|   index.html
|
+---css
|       app.css
|
+---data
|       buildings.json
|       buildingStats.json
|       globalStatsHolder.json
|       ground.json
|       toplayer.json
|
\---scripts
    |   main.js
    |
    +---config
    |       config.js
    |
    +---domain
    |       app.js
    |       appObjects.js
    |       BuildingLevel.js
    |       BuildingStatsPresenter.js
    |       Credits.js
    |       DispersionToggler.js
    |       MultiLevelBuilding.js
    |       Preloader.js
    |       ScenaryItem.js
    |       Spinner.js
    |       TotalPresenter.js
    |       WifiHotspotAccessData.js
    |
    +---fonts
    |       Sansation-light.font.js
    |       Sansation.font.js
    |
    \---vendor
            jquery.js
            raphael-min.js
            require.js
            scale.raphael.js
```

Nota: A pasta **data** contém os dados vectoriais dos objectos da aplicação; a pasta **fonts** contém duas fontes utilizadas na aplicação; e a pasta **vendor** contém as bibliotecas externas referenciadas. O conteúdo destes ficheiros não será listado por questões de relevância e extensão dos mesmos.

b. Root (/)

Index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>TFC - WIFI VIEWER</title>
    <link rel='stylesheet' href='css/app.css' type='text/css' />
    <script                                          data-main="scripts/main"
src="scripts/vendor/require.js"></script>
  </head>
  <body>
    <div id="container">
      <div id="buildings"></div>
      <div id="toplayer"></div>
    </div>
  </body>
</html>
```

c. CSS

app.css

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p,
blockquote,
pre, a, abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd,
q, s,
samp, small, strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd,
ol, ul,
li, fieldset, form, label, legend, table, caption, tbody, tfoot, thead, tr,
th, td,
article, aside, canvas, details, embed, figure, figcaption, footer, header,
hgroup,
menu, nav, output, ruby, section, summary, time, mark, audio, video {
  margin: 0;
  padding: 0;
  border: 0;
  font-size: 100%;
  font: inherit;
  vertical-align: baseline;
  outline: none;
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
html {
  height: 101%;
}
body {
  font-size: 62.5%;
  line-height: 1;
  font-family: Arial, Tahoma, sans-serif;
  overflow: hidden;
}

article, aside, details, figcaption, figure, footer, header, hgroup, menu,
nav, section {
  display: block;
```

```
}
ol, ul {
  list-style: none;
}

blockquote, q {
  quotes: none;
}
blockquote:before, blockquote:after, q:before, q:after {
  content: '';
  content: none;
}
strong {
  font-weight: bold;
}

table {
  border-collapse: collapse;
  border-spacing: 0;
}
img {
  border: 0;
  max-width: 100%;
}

p {
  font-size: 1.2em;
  line-height: 1.0em;
  color: #333;
}

svg {
  width: 100%;
  height: 100%;
}

body {
  margin: 0;
}

#container {
  position: absolute;
  top: 0px;
  left: 0px;
  width: 740px;
  height: 496px :
  margin: 0 auto;
}

#buildings {
  position: absolute;
  top: 0px;
  left: 0px;
  z-index: 200;
  overflow: visible;
  min-height: 496px;
  background-color: #1f221f;
  opacity: 1;
  visibility: hidden;
```

```
}

#toplayer {
  position: absolute;
  top: 0px;
  left: 0px;
  z-index: 250;
  overflow: visible;
  min-height: 496px;
  background-color: #1f221f;
  opacity: 1;
}
```

d. scripts

main.js

```
requirejs.config({
  paths: {
    jquery: 'vendor/jquery',
    jquerysvg: 'vendor/jquery.svg',
    raphael: 'vendor/raphael-min',
    scaleraphael : 'vendor/scale.raphael',
    raphaelutils: 'vendor/raphael-utils'
  }
});

require(['domain/app'], function(App){
  App.initialize();
});
```

e. scripts/config

config.js

```
define({
    "appname": "Wifi Viewer 2",
    "layers": [
        {
            "id": "ground",
            "type": "static",
            "pos": 0,
            "url": "data/ground.json"
        },
        {
            "id": "buildings",
            "type": "buildings",
            "pos": 1,
            "url": "data/buildings.json"
        },
        {
            "id": "props",
            "type": "static",
            "pos": 2,
            "url": "data/toplayer.json"
```

```
        },
        {
            "id": "stats",
            "type": "stats",
            "pos": 3,
            "url": "data/buildingStats.json"
        },
        {
            "id": "global",
            "type": "globalStats",
            "pos": 4,
            "url": "data/globalStatsHolder.json"
        }
    ],
    "order": [
        "ground",
        "buildings",
        "props"
    ],
    "dataService" : {
        "url":                              "http://ec2-54-213-34-158.us-west-
2.compute.amazonaws.com/api/getbuildingstats.json"
    },
    "statUpdateService" : {
        "url":                              "http://ec2-54-213-34-158.us-west-
2.compute.amazonaws.com/api/getlastminute.json"
    }
});
```

f. scripts/domain

app.js

```
define([
        'jquery',
        'raphael',
        'scaleraphael',
        'config/config',
        'domain/MultiLevelBuilding',
        'domain/BuildingLevel',
        'domain/ScenaryItem',
        'domain/Preloader',
        'domain/BuildingStatsPresenter',
        'domain/TotalPresenter',
        'domain/appObjects',
        'domain/Spinner',
        'domain/Credits',
        'domain/DispersionToggler',
        'fonts/Sansation.font',
        'fonts/Sansation-light.font'],
  function ($,
            Raphael,
            ScaleRapha,
            config,
            MultiLevelBuilding,
            BuildingLevel,
            ScenaryItem,
```

```
          Preloader,
          BuildingStatsPresenter,
          TotalPresenter,
          appObjects,
          Spinner,
          Credits,
          DispersionToggler) {

var initialize = function() {

  function updateStats(campus) {
    var spin = new Spinner({
      r : campus,
      x : 110,
      y : 30,
      color : '#fff',
      width : 10,
      r1 : 20,
      r2 : 20,
      sectorsCount : 12
    });
    spin.animate();
    $.ajax({
      type : "GET",
      url : config.statUpdateService.url,
      async : false,
      beforeSend : function(x) {
        if (x && x.overrideMimeType) {
          x.overrideMimeType("application/j-son;charset=UTF-8");
        }
      },
      dataType : "json",
      success : function(dataGot) {
        appObjects.global[0].update({
          paper : campus,
          total : dataGot.active,
          newConns : dataGot['new'],
          drops : dataGot.drops
        });

        var dt = dataGot['buildings'];
        for (var all = 0; all < appObjects.stats.length; all++) {
          for (var o = 0; o < dt.length; o++) {
            if (appObjects.stats[all].id == dt[o].building) {
              appObjects.stats[all].update({
                paper : campus,
                percent : dt[o].percent,
                total : dt[o].conns
              });
              if (!appObjects.showDispersion) {
                appObjects.stats[all].hide();
              } else {
                appObjects.stats[all].show();
              }
              dt.splice(o, 1);
            }
          }
        }
```

```
          spin.remove();
        }
      });
    }


    jQuery(document).ready(function() {
      (function extendRaphael() {
        Raphael.el.zIndex = function(z) {
          this.node.style.zIndex = z;
        }
      })();

      function resizePaper() {
        var win = $(this);
        campus.changeSize(win.width(), win.height(), true, false);
      }

      var campus = ScaleRaphael("buildings", 740, 496);
      campus.customAttributes.arc = function(value, total, R) {
        var alpha = 360 / total * value,
            a = (90 - alpha) * Math.PI / 180,
            x = 300 + R * Math.cos(a),
            y = 300 - R * Math.sin(a),
            color = "hsb(".concat(Math.round(R) / 200, ",",
            value / total, ", .75)"),
            path;
        if (total == value) {
          path = [["M", 300, 300 - R], ["A", R, R, 0, 1, 1, 299.99, 300 -
R]];
        } else {
          path = [["M", 300, 300 - R], ["A", R, R, 0, +(alpha > 180), 1, x,
y]];
        }
        return {
          path : path,
          stroke : color
        };
      };
      campus.canvas.style.zIndex = "100";
      campus.canvas.style.position = "absolute";

      resizePaper();
      $(window).resize(resizePaper);

      $.createCache = function(requestFunction) {
        var cache = {};
        return function(key, callback) {
          if (!cache[key]) {
            cache[key] = $.Deferred(function(defer) {
              requestFunction(defer, key);
            }).promise();
          }
          return cache[key].done(callback);
        };
      }

      $.loadObjects = $.createCache(function(defer, url) {
        $.ajax({
```

```
          type : "GET",
          url : url,
          async : true,
          beforeSend : function(x) {
            if (x && x.overrideMimeType) {
              x.overrideMimeType("application/j-son;charset=UTF-8");
            }
          },
          dataType : "json",
          success : defer.resolve,
          error : defer.reject
        });
      });

      function putLoadedObjectsInArray(pos, data) {
        if (config['layers'][pos].type == 'static') {
          console.log("Got   %i   %s   paths   from   service",   data.length,
config['layers'][pos].id);
          var dataElements = new Array();
          for (var i = 0; i < data.length; i++) {
            var path = new ScenaryItem(data[i]);
            dataElements.push(path);
          }
          appObjects[config['layers'][pos].id] = dataElements;
        } else if (config['layers'][pos].type == 'buildings') {
          console.log("Got      %i      %s      paths      from      service",
data['buildings'].length, config['layers'][pos].id);
          var dataElements = new Array();
          for (var it = 0; it < data['buildings'].length; it++) {
            var bdng = new MultiLevelBuilding();
            bdng.fromJSON(data['buildings'][it]);
            dataElements.push(bdng);
          }
          appObjects[config['layers'][pos].id] = dataElements;
        } else if (config['layers'][pos].type == 'stats') {
          console.log("Got      %i      %s      paths      from      service",
data['buildingStatsPresenters'].length, config['layers'][pos].id);
          var dataElements = new Array();
          for (var it = 0; it < data['buildingStatsPresenters'].length; it++)
{
            var tooltip = new BuildingStatsPresenter();
            tooltip.fromJSON(data['buildingStatsPresenters'][it]);
            dataElements.push(tooltip);
          }
          appObjects[config['layers'][pos].id] = dataElements;
        } else if (config['layers'][pos].type == 'globalStats') {
          console.log("Got      %i      %s      paths      from      service",
data['globalStats'].length, config['layers'][pos].id);
          var dataElements = new Array();
          for (var it = 0; it < data['globalStats'].length; it++) {
            var global = new TotalPresenter();
            global.fromJSON(data['globalStats'][it]);
            dataElements.push(global);
          }
          appObjects[config['layers'][pos].id] = dataElements;
        }
      }

      function drawSet(paper) {
```

```
        p.disappear();
        for (var layer = 0; layer < config.order.length; layer++) {
          //console.log("%d : %s -> %d objects", layer, config.order[layer],
objects[config.order[layer]].length);
          for (var item = 0; item < appObjects[config.order[layer]].length;
item++) {
            //console.log("layer: " + config.order[layer]);
            appObjects[config.order[layer]][item].draw(paper);
          }
        }
      }

      /** STARTUP **/

      var calls = new Array();

      var p = new Preloader(campus);
      //p.updateVal(900);

      for (var l = 0; l < config.layers.length; l++) {
        var localid = l;
        //console.log(localid);
        (function(index) {
          calls.push($.loadObjects(config.layers[l].url, function(algo) {
            putLoadedObjectsInArray(index, algo);
            //p.increment(101/(config.layers.length));
          }));
        })(l);
      }

      $.when.apply($, calls).done(function() {
        drawSet(campus);
        var wi = campus.print(385, 60, 'Wi', "Sansation", 60).attr({
          fill : '#dbdbdb'
        }).glow({
          width : 5,
          opacity : 0.6,
          color : "#005A9C"
        });
        var fi = campus.print(455, 60, 'Fi', "Sansation", 60).attr({
          fill : '#dbdbdb'
        }).glow({
          width : 5,
          opacity : 0.6,
          color : "#005A9C"
        });
        campus.print(520, 60, 'Viewer', "Sansation", 60).attr({
          fill : '#dbdbdb'
        }).glow({
          width : 5,
          opacity : 0.6,
          color : "#005A9C"
        });
        campus.print(705, 80, '2', "Sansation", 40).attr({
          fill : '#00A9A2'
        }).glow({
          width : 3,
          opacity : 0.2,
          color : "#00A9A2"
```

```
        });

        cred = new Credits({
          paper : campus,
          x : 690,
          y : 470,
          attrs : {
            fill : "#dbdbdb"
          }
        });
        cred.draw();

        toggler = new DispersionToggler({
          paper : campus,
          x : 10,
          y : 130,
          attrs : {
            fill : "#fff",
            stroke : "#ff3d41",
            "stroke-width" : 2
          }
        });
        toggler.draw();

        updateStats(campus);
        var timerId = setInterval(function() {
          updateStats(campus);
        }, 60000);

      });
    });
    // document ready
  }//initialize

  return {
    initialize : initialize
  };
  //return
});
//define
```

appObjects.js

```
define({
  objects : []
});
```

BuildingLevel.js

```
define(function() {

  // constructor
  var buildingLevel = function() {
    // vars aqui são privadas
    var id = '';
    var floor = '';
    var hasName = false;
    var hasHotspotInfo = false;
```

```
    var topX = 0;
    var topY = 0;
    var drawAngle = 45;
    var height = 1;
    var width = 0;
    var length = 0;
    var topPathString = '';
    var lengthPathString = '';
    var widthPathString = '';
    var visibleTop = true;
    for (var n in arguments[0]) {
      this[n] = arguments[0][n];
    }
  };

  // public (todas as instâncias)
  buildingLevel.prototype = {
    getInternalId : function() {
      return internalId;
    },
    setId : function(v) {
      this.id = v;
    },
    getId : function() {
      return this.id;
    },
    setTag : function(v) {
      this.tag = v;
    },
    getTag : function() {
      return this.tag;
    },
    setTopX : function(v) {
      this.topX = v;
    },
    getTopX : function() {
      return this.topX;
    },
    setTopY : function(v) {
      this.topY = v;
    },
    getTopY : function() {
      return this.topY;
    },
    setHeight : function(v) {
      this.height = v;
    },
    getHeight : function() {
      return this.height;
    },
    setWidth : function(v) {
      this.width = v;
    },
    getWidth : function() {
      return this.width;
    },
    setLength : function(v) {
      this.length = v;
    },
```

```
    getLength : function() {
      return this.length;
    },
    getTopPathString : function() {
      return this.topPathString;
    },
    getLengthPathString : function() {
      return this.lengthPathString;
    },
    getWidthPathString : function() {
      return this.widthPathString;
    },
    setDrawAngle : function(v) {
      this.drawAngle = v;
    },
    getDrawAngle : function() {
      return this.drawAngle;
    },
    generatePaths : function() {
      /* offset altura = altura + (piso * altura) + piso // p0 10+(0*10)+0 =
10 //p1 10+(1*10)+1 = 21 // p5 10+(5*10)+5 = 65 **/
      var calc_w = Math.abs(Math.round(this.width *
Math.cos(this.drawAngle)));
      var calc_h = Math.abs(Math.round(this.length *
Math.cos(this.drawAngle)));
      this.topPathString = 'M ' + this.topX + ' ' + (this.topY - this.height)
+ ' l -' + this.width + ' -' + calc_w + ' l ' + this.length + ' -' + calc_h +
' l ' + this.width + ' ' + calc_w + ' z';
      this.lengthPathString = 'M ' + this.topX + ' ' + this.topY + ' l ' +
this.length + ' -' + calc_h + ' l 0 -' + this.height + ' l -' + this.length +
' ' + calc_h + ' z';
      this.widthPathString = 'M ' + this.topX + ' ' + this.topY + ' l -' +
this.width + ' -' + calc_w + ' l 0 -' + this.height + ' l ' + this.width + '
' + calc_w + ' z';
    },
    setFloor : function(floorNum) {
      this.floor = floorNum;
    },
    getFloor : function() {
      return this.floor;
    },
    setHasHotspotMarker : function(v) {
      this.hasHotspotInfo = v;
    },
    hasHotspotMarker : function() {
      return this.hasHotspotInfo;
    },
    setHasName : function(v) {
      this.hasName = v;
    },
    hasName : function() {
      return this.hasName;
    },
    setVisibleTop : function(v) {
      this.visibleTop = v;
    },
    isVisibleTop : function() {
      return this.visibleTop;
    }
```

```
  };

  return buildingLevel;
});
```

BuildingStatsPresenter.js

```
define(['jquery',      'raphael',      'scaleraphael',      'config/config',
'domain/appObjects', 'domain/WifiHotspotAccessData', 'fonts/Sansation.font'],
function($, r, scaleRaph, config, appObjects, WifiHotspotAccessData) {

  // constructor
  var buildingStatsPresenter = function() {
    // vars aqui são privadas
    var id = '';
    var textTotal = '';
    var textPercent = '';
    var tipX = 0;
    var tipY = 0;
    var width = 0;
    var height = 0;
    var tipPos = '';
    var tAttrs = {};
    var pAttrs = {};
    var object;
    for (var n in arguments[0]) {
      this[n] = arguments[0][n];
    }
  };

  // public (todas as instâncias)
  buildingStatsPresenter.prototype = {
    setId : function(v) {
      this.id = v;
    },
    getId : function() {
      return this.id;
    },
    setTextTotal : function(v) {
      this.textTotal = v;
    },
    getTextTotal : function() {
      return this.textTotal;
    },
    setTextPercent : function(v) {
      this.textPercent = v;
    },
    getTextPercent : function() {
      return this.textPercent;
    },
    setTextTextNewConns : function(v) {
      this.textNewConns = v;
    },
    getTextNewConns : function() {
      return this.textNewConns;
    },
    setTextDrops : function(v) {
      this.textDrops = v;
```

```
    },
    getTextDrops : function() {
      return this.textDrops;
    },
    setTopX : function(v) {
      this.topX = v;
    },
    getTopX : function() {
      return this.topX;
    },
    setTopY : function(v) {
      this.topY = v;
    },
    getTopY : function() {
      return this.topY;
    },
    setWidth : function(v) {
      this.width = v;
    },
    getWidth : function() {
      return this.width;
    },
    setLength : function(v) {
      this.length = v;
    },
    getLength : function() {
      return this.length;
    },
    getPathString : function() {
      return this.pathString;
    },
    setPathString : function(v) {
      this.pathString = v;
    },
    setDrawAngle : function(v) {
      this.drawAngle = v;
    },
    getDrawAngle : function() {
      return this.drawAngle;
    },
    setObject : function(v) {
      this.object = v;
    },
    getObject : function() {
      return this.object;
    },
    hexColorFromPercent : function(percent) {
      var percent = parseInt(percent);
      var r = 0, g = 255;
      if (percent > 50) {
        r = 255;
        g = 255 - Math.round(255 * (percent / 100));
      } else {
        r = Math.round(255 * (percent / 50));
        g = 255;
      }
      var ret = Raphael.rgb(r, g, 0);
      return ret;
    },
```

```
    show : function() {

    },
    draw : function(params) {
      this.textPercent = params['percent'].toString();
      this.textTotal = params['total'].toString();

      if (this.tipPos == 'left') {
        params['paper'].setStart();
        params['paper'].rect(this.tipX + 10, this.tipY -
Math.round(this.height / 2) - 1, this.width, this.height, 10).attr({
          fill : '#141414',
          stroke : 'none',
          opacity : 0
        });
        params['paper'].rect(this.tipX + 12, this.tipY -
Math.round(this.height / 2) + 2, this.width - 4, this.height - 6, 8).attr({
          fill : this.hexColorFromPercent(params['percent']),
          stroke : 'none',
          opacity : 0
        });
        params['paper'].rect(this.tipX + 16, this.tipY -
Math.round(this.height / 2) + 6, this.width - 12, this.height - 14, 6).attr({
          fill : '#fff',
          stroke : '#141414',
          'stroke-size' : 1,
          opacity : 0
        });
        params['paper'].print(this.tipX + 20 + (18 - (6 *
this.textPercent.length)), this.tipY - Math.round(this.height / 2) + 16,
this.textPercent + "%", "Sansation", 16).attr(this.pAttrs).attr({
          opacity : 0
        });
        params['paper'].print(this.tipX + 18, this.tipY -
Math.round(this.height / 2) + 29, "Active: " + this.textTotal, "Sansation",
10).attr(this.tAttrs).attr({
          opacity : 0
        });
        params['paper'].path("M " + this.tipX + ", " + this.tipY + " l 11, -
11 l 0, 20 z").attr({
          fill : '#141414',
          stroke : 'none',
          opacity : 0
        });
        this.object = params['paper'].setFinish();
        for (var i = 0; i < this.object.length; i++) {
          this.object[i].animate({
            opacity : 1
          }, 1000);
        }
      } else if (this.tipPos == 'right') {
        params['paper'].setStart();
        params['paper'].rect(this.tipX - this.width - 10, this.tipY -
Math.round(this.height / 2) - 1, this.width, this.height, 10).attr({
          fill : '#141414',
          stroke : 'none',
          opacity : 0
        });
```

```
        params['paper'].rect(this.tipX - this.width - 8, this.tipY -
Math.round(this.height / 2) + 2, this.width - 4, this.height - 6, 8).attr({
          fill : this.hexColorFromPercent(params['percent']),
          stroke : 'none',
          opacity : 0
        });
        params['paper'].rect(this.tipX - this.width - 4, this.tipY -
Math.round(this.height / 2) + 6, this.width - 12, this.height - 14, 6).attr({
          fill : '#fff',
          stroke : '#141414',
          'stroke-size' : 1,
          opacity : 0
        });
        params['paper'].print(this.tipX - this.width + (18 - (6 *
this.textPercent.length)), this.tipY - Math.round(this.height / 2) + 16,
this.textPercent + "%", "Sansation", 16).attr(this.pAttrs).attr({
          opacity : 0
        });
        params['paper'].print(this.tipX - this.width, this.tipY -
Math.round(this.height / 2) + 29, "Active: " + this.textTotal, "Sansation",
10).attr(this.tAttrs).attr({
          opacity : 0
        });
        params['paper'].path("M " + this.tipX + ", " + this.tipY + " l -11, -
11 l 0, 20 z").attr({
          fill : '#141414',
          stroke : 'none',
          opacity : 0
        });
        this.object = params['paper'].setFinish();
        for (var i = 0; i < this.object.length; i++) {
          this.object[i].animate({
            opacity : 1
          }, 1000);
        }
      } else {
        params['paper'].setStart();
        params['paper'].rect(this.tipX - Math.round(this.width / 2),
this.tipY - (this.height + 10), this.width, this.height, 10).attr({
          fill : '#141414',
          stroke : 'none',
          opacity : 0
        });
        params['paper'].rect(this.tipX - Math.round(this.width / 2) + 2,
this.tipY - (this.height + 10) + 2, this.width - 4, this.height - 4,
8).attr({
          fill : this.hexColorFromPercent(params['percent']),
          stroke : 'none',
          opacity : 0
        });
        params['paper'].rect(this.tipX - Math.round(this.width / 2) + 6,
this.tipY - (this.height + 10) + 6, this.width - 12, this.height - 12,
6).attr({
          fill : '#fff',
          stroke : '#141414',
          'stroke-size' : 1,
          opacity : 0
        });
```

```
        params['paper'].print(this.tipX - (6 * this.textPercent.length),
this.tipY - this.height + 6, this.textPercent + "%", "Sansation",
16).attr(this.pAttrs).attr({
          opacity : 0
        });
        params['paper'].print(this.tipX - Math.round(this.width / 2) + 8,
this.tipY - this.height + 20, "Active: " + this.textTotal, "Sansation",
10).attr(this.tAttrs).attr({
          opacity : 0
        });
        params['paper'].path("M " + this.tipX + ", " + this.tipY + " l -10, -
11 l 20, 0 z").attr({
          fill : '#141414',
          stroke : 'none',
          opacity : 0
        });
        this.object = params['paper'].setFinish();
        for (var i = 0; i < this.object.length; i++) {
          this.object[i].animate({
            opacity : 1
          }, 1000);
        }
      }
      this.object.click( function(attrs) {
        return function() {
          var dg = new WifiHotspotAccessData();
          dg.fetchRemoteData(attrs);
        }
      }({
        paper : params['paper'],
        building : this.id
      }));
    },
    update : function(params) {
      if (!appObjects.suspend) {
        if (this.object) {
          for (var i = 0; i < this.object.length; i++) {
            this.object[i].animate({
              opacity : 0
            }, 500);
          }
          this.object = null;
          this.draw(params)
        } else {
          this.draw(params);
        }
      }
    },
    hide : function() {
      this.object.stop().animate({
        opacity : 0
      }, 300, function(obj) {
        obj.hide();
      }(this.object));
      //.hide();
    },
    show : function() {
      this.object.show();
      this.object.stop().animate({
```

```
        opacity : 1
      }, 300);
    },
    toJSON : function() {
      var str = '{';
      str += '"id":"' + this.id;
      str += '", "tipX":' + this.tipX;
      str += ', "tipY":' + this.tipY;
      str += ', "width":' + this.width;
      str += ', "height":' + this.height;
      str += ', "tipPos":"' + this.tipPos;
      str += '", "tAttrs":' + JSON.stringify(this.tAttrs || {}, null, 2);
      str += ', "pAttrs":' + JSON.stringify(this.tAttrs || {}, null, 2);
      str += '}';
      return str;
    },
    fromJSON : function(b) {
      this.id = b.id;
      this.tipX = b.tipX;
      this.tipY = b.tipY;
      this.width = b.width;
      this.height = b.height;
      this.tipPos = b.tipPos;
      this.tAttrs = b.tAttrs;
      this.pAttrs = b.pAttrs;
    }
  };

  return buildingStatsPresenter;
});
```

Credits.js

```
define(['jquery',      'raphael',      'scaleraphael',      'config/config',
'domain/appObjects', 'fonts/Sansation.font'], function($, Raphael, scaleRaph,
config, appObjects) {
  var credits = function() {
    var paper = '';
    var x = 0;
    var y = 0;
    var attrs = [];
    var object = null;
    var creditsHolder = null;

    for (var n in arguments[0]) {
      this[n] = arguments[0][n];
    }
  };

  credits.prototype = {
    draw : function() {
      this.paper.setStart();
      //this.paper.rect(270,   198,   200,   100,   10).attr({fill:   '#141414',
stroke: 'none', opacity: 0});
      this.paper.print(0,  377,  "(C)  Copyrigth  2013\nTrabalho  realizado  no
contexto  da  cadeira\nde  Trabalho  de  Fim  de  Curso  da  Licenciatura\nem
Informática de Gestão da ULHT por\n\n21000490 José Miguel Verissimo\n21002427
```

```
Luís Filipe Ferreira\n\nOrientado por Prof.ª Dr.ª Inês Oliveira",
"Sansation", 12).attr({
        fill : '#ffe40a',
        stroke : 'none'
      });
      var creditsHolder = this.paper.setFinish();
      creditsHolder.transform('s0.1');
      creditsHolder.hide();

      var cred = this.paper.print(this.x, this.y, "credits", "Sansation",
12).attr(this.attrs);
      var credbb = cred.getBBox();
      this.object = this.paper.rect(credbb.x, credbb.y, credbb.width,
credbb.height).attr({
        fill : '#141414',
        stroke : 'none',
        'fill-opacity' : 0
      });
      this.object.hover(function() {
        cred.stop().animate({
          transform : 's2'
        }, 400, "bounce");
        creditsHolder.show();
        creditsHolder.stop().animate({
          opacity : 1,
          transform : 's1'
        }, 400, "linear");
      }, function() {
        cred.stop().animate({
          transform : 's1'
        }, 200, "bounce");
        creditsHolder.stop().animate({
          opacity : 0,
          transform : 's0.1'
        }, 400, "linear");
      });
      this.object.click(this.showCredits());
    },
    showCredits : function() {
      //this.paper.rect(50,350,200,400).attr({fill: "#05001e", fill:
"none"});
    }
  };

  return credits;
});
```

DispersionToggler.js

```
define(['jquery', 'raphael', 'scaleraphael', 'config/config',
'domain/appObjects', 'fonts/Sansation.font'], function($, Raphael, scaleRaph,
config, appObjects) {
  var despersionToggler = function() {
    var paper = '';
    var x = 0;
    var y = 0;
    var attrs = [];
    var object = null;
```

```
    var creditsHolder = null;

    for (var n in arguments[0]) {
      this[n] = arguments[0][n];
    }
  };

  despersionToggler.prototype = {
    draw : function() {
      //this.paper.setStart();
      console.log('aqui');
      this.paper.rect(this.x, this.y, 100, 15, 2).attr({
        fill : "#fff",
        stroke : "#ff3d41",
        "stroke-width" : 1
      });
      this.paper.text(this.x + 43, this.y + 7, "Show distribution",
"Sansation", 14).attr({
        fill : "#000",
        stroke : "none"
      });
      var toggler = this.paper.circle(this.x + 97, this.y + 7,
10).attr(this.attrs);
      //var dispersionTogglerButton = this.paper.setFinish();

      toggler.hover(function() {
        toggler.stop().animate({
          fill : "#ff5b5f"
        }, 100, "linear");

      }, function() {
        var color;
        if (appObjects.showDispersion) {
          color = "#ff3d41";
        } else {
          color = "#fff";
        }
        toggler.stop().animate({
          fill : color
        }, 200, "linear");
      });

      toggler.click(function() {
        appObjects.showDispersion = !appObjects.showDispersion;
        for (var all = 0; all < appObjects.stats.length; all++) {
          if (!appObjects.showDispersion) {
            appObjects.stats[all].hide();
          } else {
            appObjects.stats[all].show();
          }
        }
      });
    },
    showCredits : function() {
      //this.paper.rect(50,350,200,400).attr({fill:      "#05001e",      fill:
"none"});
    }
  };
```

```
  return despersionToggler;
});
```

MultiLevelBuilding.js

```
define(['domain/BuildingLevel',              'domain/WifiHotspotAccessData'],
function(BuildingLevel, WifiHotspotAccessData) {

  // constructor
  var building = function() {
    // vars aqui são privadas
    var id = '';
    var levels = new Array();
    var levelObjects = new Array();
    var numLevels = 1;
    var color = '#ffffff';
    var altColor = "#555555";
    var drawAngle = 45;
    var labelData = new Array({
      x : 0,
      y : 0,
      size : 0,
      color : '#FFFFFF',
      text : '',
      'text-rotation' : 37
    });

    for (var n in arguments[0]) {
      this[n] = arguments[0][n];
    }
  };

  // public (todas as instâncias)
  building.prototype = {
    getInternalId : function() {
      return internalId;
    },
    setId : function(v) {
      this.id = v;
    },
    getId : function() {
      return this.id;
    },
    setTag : function(v) {
      this.tag = v;
    },
    getTag : function() {
      return this.tag;
    },
    colorLuminance : function(hex, lum) {
      // validate hex string
      hex = String(hex).replace(/[^0-9a-f]/gi, '');
      if (hex.length < 6) {
        hex = hex[0] + hex[0] + hex[1] + hex[1] + hex[2] + hex[2];
      }
      lum = lum || 0;
      // convert to decimal and change luminosity
      var rgb = "#", c, i;
```

```
      for ( i = 0; i < 3; i++) {
        c = parseInt(hex.substr(i * 2, 2), 16);
        c    =    Math.round(Math.min(Math.max(0,    c    +    (c    *    lum)),
255)).toString(16);
        rgb += ("00" + c).substr(c.length);
      }
      return rgb;
    },
    setAltColor : function(v) {
      this.altColor = v;
    },
    getAltColor : function() {
      return this.altColor;
    },
    setColor : function(v) {
      this.color = (v.toString().charAt(0) != '#') ? '#' + v : v;
      this.setAltColor(this.colorLuminance(this.color, -0.3));
    },
    getColor : function() {
      return this.color;
    },
    setDrawAngle : function(v) {
      this.drawAngle = v;
    },
    getDrawAngle : function() {
      return this.drawAngle;
    },
    getTopStyle : function() {
      return "{'fill': '" + this.color + "', 'stroke': '#141414', 'stroke-
width': 1}";
    },
    getLengthStyle : function() {
      return "{'fill': '" + this.altColor + "', 'stroke': '#141414', 'stroke-
width': 1}";
    },
    getWidthStyle : function() {
      return "{'fill': '" + this.color + "', 'stroke': '#141414', 'stroke-
width': 1}";
    },
    setLevels : function(levels) {
      this.levels = levels;
    },
    getLevels : function() {
      return this.levels;
    },
    addLevel : function(level) {
      this.levels = this.levels || new Array();
      this.levels.push(level);
    },
    getLevelObjects : function() {
      return this.levelObjects;
    },
    setLevelObjects : function(v) {
      this.levelObjects = v;
    },
    getLabelData : function() {
      return this.labelData;
    },
    setLabelData : function(v) {
```

```
      this.labelData = v;
    },
    toJSON : function() {
      var str = '{';
      str += '"id":"' + this.id;
      str += '", "levels":' + JSON.stringify(this.levels || {}, null, 2);
      str += ', "numLevels":"' + this.numLevels;
      str += '", "color":"' + this.color;
      str += '", "altColor":"' + this.altColor;
      str += '", "drawAngle":"' + this.drawAngle;
      str += '", "labelData":' + JSON.stringify(this.labelData || {}, null,
2);
      str += '}';
      return str;
    },
    fromJSON : function(b) {
      this.id = b.id;
      this.levels = new Array();
      for (var i = 0; i < b.levels.length; i++) {
        var bl = new BuildingLevel(b.levels[i]);
        this.levels.push(bl);
      }
      this.numLevels = b.numLevels;
      this.color = b.color;
      this.altColor = b.color;
      this.drawAngle = b.drawAngle;
      this.labelData = b.labelData;
    },
    buildMultiStory : function(_levels, _topX, _topY, _drawAngle, _height,
_width, _length, _hotSpotLevel, _infoLevel) {
      this.levels = new Array();
      this.numLevels = _levels;
      var levels = _levels;
      var topX = _topX;
      var topY = _topY;
      var drawAngle = _drawAngle;
      var height = _height;
      var width = _width;
      var length = _length;
      var hotSpotLevel = _hotSpotLevel;
      var infoLevel = _infoLevel;

      /* offset altura = altura + (piso * altura) */

      for (var i = 0; i < levels; i++) {
        var yCorrected = topY - ((i * height) + i);
        this.levels[i] = new BuildingLevel({
          'topX' : topX,
          'topY' : yCorrected,
          'drawAngle' : drawAngle,
          'height' : height,
          'width' : width,
          'length' : length,
          'visibleTop' : false
        });
        if (i == hotSpotLevel) {
          this.levels[i].setHasHotspotMarker(true);
        }
        if (i == infoLevel) {
```

```
          this.levels[i].setHasName(true);
        }
        if (i == levels - 1) {
          this.levels[i].setVisibleTop(true);
        }

        this.levels[i].generatePaths();
      }
    },
    draw : function(paper) {
      this.levelObjects = this.levelObjects || new Array();
      paper.setStart();
      for (var i = 0; i < this.levels.length; i++) {
        this.levels[i].generatePaths();
        if (this.levels[i].isVisibleTop()) {
          paper.path(this.levels[i].getTopPathString()).attr({
            fill : this.color
          });
        }
        paper.path(this.levels[i].getLengthPathString()).attr({
          fill : this.colorLuminance(this.color, -0.2)
        });
        paper.path(this.levels[i].getWidthPathString()).attr({
          fill : this.colorLuminance(this.color, -0.5)
        });
      }

      if (this.labelData) {
        fillcolor       =       (this.labelData.hasOwnProperty('fill')       ?
this.labelData['fill'] : this.color);
        var  lbl  =  paper.ellipse(this.labelData['x'],  this.labelData['y'],
this.labelData['size'], this.labelData['size'] - 2).attr({
          stroke : this.labelData['color'],
          'stroke-width' : 2,
          fill : fillcolor
        });
        var tagLetter = paper.text(this.labelData['x'], this.labelData['y'],
this.labelData['text']).attr({
          fill : this.labelData['color']
        }).rotate(this.labelData['text-rotation'],       this.labelData['x'],
this.labelData['y']);
        this.id = (this.id == null ? this.labelData['text'] : this.id);
      }
      var bdng = paper.setFinish();

      bdng.click( function(attrs) {
        return function() {
          var dg = new WifiHotspotAccessData();
          dg.fetchRemoteData(attrs);
        }
      }({
        paper : paper,
        building : this.id
      }));
      this.levelObjects.push(bdng);
      return (bdng);
    }
  };
```

```
  return building;
});
```

Preloader.js

```
define(['jquery', 'raphael'], function($, Raphael) {

  function preload(canvas) {
    this.R = 100;
    this.init = true;
    this.total = 100;
    this.currentVal = 0;
    this.param = {
      stroke : "#fff",
      "stroke-width" : 30
    };
    this.hash = document.location.hash;
    this.marksAttr = {
      fill : this.hash || "#444",
      stroke : "none"
    };

    this.paper = ScaleRaphael("toplayer", 740, 496);
    this.paper.changeSize(window.innerWidth,     window.innerHeight,     true,
false);

    this.paper.setStart();
    this.bg = this.paper.rect(0, 0, 740, 496).attr({
      fill : "#1f221f",
      stroke : "none"
    });
    this.handle = this.paper.path().attr(this.param).attr({
      arc : [0, 100, 100]
    });
    this.pl = this.paper.setFinish();
  }


  preload.prototype = {
    updateVal : function(value) {
      var  color  =  "hsb(".concat(Math.round(this.R)  /  100,  ",",  value  /
this.total, ", .75)");
      if (this.init) {
        this.handle.animate({
          arc : [Math.round(value), this.total, this.R]
        }, 900, ">");
      } else {
        if (!value || value > this.total) {
          value = this.total;
          this.handle.animate({
            arc : [Math.round(value), this.total, this.R]
          }, 750, "bounce", function() {
            this.handle.attr({
              arc : [0, this.total, this.R]
            });
          });
        } else {
          this.handle.animate({
```

```
            arc : [Math.round(value), this.total, this.R]
          }, 750, "elastic");
        }
      }
      this.bg.toFront();
      this.handle.toFront();
    },
    increment : function(value) {
      this.currentVal += Math.round(value);
      if (this.currentVal > this.total) {
        this.currentVal = this.total;
      }
      var color = "hsb(".concat(Math.round(this.R) / 100, ",", value /
this.total, ", .75)");
      if (!value || Math.round(this.currentVal) > this.total) {
        value = this.total;
        this.handle.animate({
          arc : [Math.round(this.currentVal), this.total, this.R]
        }, 750, "bounce", function() {
          this.handle.attr({
            arc : [0, this.total, this.R]
          });
        });
      } else {
        this.handle.animate({
          arc : [Math.round(this.currentVal), this.total, this.R]
        }, 750, "elastic");
      }
      this.bg.toFront();
      this.handle.toFront();
    },
    disappear : function() {
      this.bg.animate({
        y : -500
      }, 2000, "backOut");
      this.handle.hide();
      this.paper.remove();
      $("#toplayer").css("visibility", "hidden");
      $("#buildings").css("visibility", "visible");
    }
  };

  return preload;
});
```

ScenaryItem.js

```
define(['jquery', 'raphael', 'scaleraphael'], function($, r, scaleRaph) {

  // constructor
  var scenaryItem = function() {
    // vars aqui são privadas
    var path = "";
    var attrs = new Array();
    var transform = "";
    var id = "";

    for (var n in arguments[0]) {
      this[n] = arguments[0][n];
    }
```

```
  };

  // public (todas as instâncias)
  scenaryItem.prototype = {
    getPath : function() {
      return this.path;
    },
    setPath : function(v) {
      this.path = v;
    },
    getAttrs : function() {
      return this.attrs;
    },
    setAttrs : function(v) {
      this.attrs = v;
    },
    getId : function() {
      return this.id;
    },
    setId : function(v) {
      this.id = v;
    },
    draw : function(paper) {
      var p = paper.path(this.path);
      if (this.attrs) {
        p.attr(this.attrs);
      }
      if (this.transform) {
        p.transform(this.transform);
      }
      if (this.data) {
        p.data('id', this.id);
      }

      return p;
    }
  };

  return scenaryItem;
});
```

Spinner.js

```
define(['jquery',      'raphael',      'scaleraphael',      'config/config',
'domain/appObjects', 'fonts/Sansation.font'], function($, Raphael, scaleRaph,
config, appObjects) {
  var spinner = function() {
    var sectorsCount = 12;
    var color = "#fff", width = 10, r1 = 20, r2 = 20, x, y, r, tick, sectors
= [], opacity = [];

    for (var n in arguments[0]) {
      this[n] = arguments[0][n];
    }
  };

  spinner.prototype = {
    ticker : function() {
```

```
      this.opacity.unshift(this.opacity.pop());
      for (var i = 0; i < this.sectorsCount; i++) {
        this.sectors[i].attr("opacity", this.opacity[i]);
      }
      this.r.safari();
    },
    animate : function() {
      this.sectors = [];
      this.opacity = [];
      var cx = this.r2 + this.width;
      var cy = this.r2 + this.width;
      var beta = 2 * Math.PI / this.sectorsCount;
      var pathParams = {
        stroke : this.color,
        "stroke-width" : this.width,
        "stroke-linecap" : "round"
      };

      for (var i = 0; i < this.sectorsCount; i++) {
        var alpha = beta * i - Math.PI / 2, cos = Math.cos(alpha), sin =
Math.sin(alpha);
        this.opacity[i] = 1 / this.sectorsCount * i;
        this.sectors[i] = this.r.path([["M", cx + this.x + this.r1 * cos, cy
+ this.y + this.r1 * sin], ["L", cx + this.x + this.r2 * cos, cy + this.y +
this.r2 * sin]]).attr(pathParams);
      }
      this.tick = setTimeout(this.ticker, (1000 / this.sectorsCount));
    },
    remove : function() {
      clearTimeout(this.tick);
      for (var x = 0; x < this.sectors.length; x++) {
        this.sectors[x].remove();
      }
    }
  }

  return spinner;
});
```

TotalPresenter.js

```
define(['jquery',      'raphael',      'scaleraphael',      'config/config',
'domain/appObjects', 'domain/WifiHotspotAccessData', 'fonts/Sansation.font'],
function($, r, scaleRaph, config, appObjects, WifiHotspotAccessData) {

  // constructor
  var totalPresenter = function() {
    // vars aqui são privadas
    var id = '';
    var active = '';
    var newConns = '';
    var drops = '';
    var topX = 0;
    var topY = 0;
    var width = 0;
    var height = 0;
    var attrs = {};
    var object;
```

```
  for (var n in arguments[0]) {
    this[n] = arguments[0][n];
  }
};

// public (todas as instâncias)
totalPresenter.prototype = {
  setId : function(v) {
    this.id = v;
  },
  getId : function() {
    return this.id;
  },
  setActive : function(v) {
    this.textTotal = v;
  },
  getActive : function() {
    return this.textTotal;
  },
  setNewConns : function(v) {
    this.textPercent = v;
  },
  getNewConns : function() {
    return this.textPercent;
  },
  setDrops : function(v) {
    this.textNewConns = v;
  },
  getDrops : function() {
    return this.textNewConns;
  },
  setTopX : function(v) {
    this.topX = v;
  },
  getTopX : function() {
    return this.topX;
  },
  setTopY : function(v) {
    this.topY = v;
  },
  getTopY : function() {
    return this.topY;
  },
  setWidth : function(v) {
    this.width = v;
  },
  getWidth : function() {
    return this.width;
  },
  setHeight : function(v) {
    this.height = v;
  },
  getHeight : function() {
    return this.height;
  },
  setObject : function(v) {
    this.object = v;
  },
  getObject : function() {
```

```
      return this.object;
    },
    hexColorFromPercent : function(percent) {
      var percent = parseInt(percent);
      var r = 0, g = 255;
      if (percent > 50) {
        r = 255;
        g = 255 - Math.round(255 * (percent / 100));
      } else {
        r = Math.round(255 * (percent / 50));
        g = 255;
      }
      var ret = Raphael.rgb(r, g, 0);
      return ret;
    },
    show : function() {

    },
    draw : function(params) {
      this.active = params['total'].toString();
      this.newConns = params['newConns'].toString();
      this.drops = params['drops'].toString();

      params['paper'].setStart();
      params['paper'].rect(this.topX,  this.topY,  this.width,  this.height,
10).attr({
        fill : '#141414',
        stroke : '#fff',
        'stroke-width' : 2,
        opacity : 0
      });
      params['paper'].rect(this.topX + 1, this.topY + 1, this.width - 2, 15,
9).attr({
        fill : '#fff',
        stroke : 'none',
        'stroke-width' : 2,
        opacity : 0
      });
      params['paper'].rect(this.topX + 1, this.topY + 5, this.width - 2,
20).attr({
        fill : '#fff',
        stroke : 'none',
        'stroke-width' : 2,
        opacity : 0
      });

      params['paper'].print(this.topX + 7, this.topY + 12, "Last Minute",
"Sansation", 16).attr({
        fill : '#005A9C',
        stroke : 'none'
      });
      params['paper'].print(this.topX + 7, this.topY + 34, "Total    :    " +
this.active, "Sansation", 12).attr(this.attrs);
      params['paper'].print(this.topX + 7, this.topY + 49, "New     :    " +
this.newConns, "Sansation", 12).attr(this.attrs);
      params['paper'].print(this.topX + 7, this.topY + 64, "Drops :    " +
this.drops, "Sansation", 12).attr(this.attrs);
      this.object = params['paper'].setFinish();
      for (var i = 0; i < this.object.length; i++) {
```

```
        this.object[i].animate({
          opacity : 1
        }, 1000);
      }
      this.object.click( function(attrs) {
        return function() {
          console.log('click');
          var dg = new WifiHotspotAccessData();
          dg.fetchRemoteData(attrs);
        }
      }({
        paper : params['paper'],
        building : this.id
      }));
    },
    update : function(params) {
      if (!appObjects.suspend) {
        if (this.object) {
          for (var i = 0; i < this.object.length; i++) {
            this.object[i].animate({
              opacity : 0
            }, 500);
          }
          this.object = null;
          this.draw(params)
        } else {
          this.draw(params);
        }
      }
    },
    toJSON : function() {
      var str = '{';
      str += '"id":"' + this.id;
      str += '", "topX":' + this.topX;
      str += ', "topY":' + this.topY;
      str += ', "width":' + this.width;
      str += ', "height":' + this.height;
      str += ', "attrs":' + JSON.stringify(this.attrs || {}, null, 2);
      str += '}';
      return str;
    },
    fromJSON : function(b) {
      this.id = b.id;
      this.topX = b.topX;
      this.topY = b.topY;
      this.width = b.width;
      this.height = b.height;
      this.attrs = b.attrs;
    }
  };

  return totalPresenter;
});
```

WifiHotspotAccessData.js

```
define(['jquery',      'raphael',      'scaleraphael',      'config/config',
'domain/appObjects', 'fonts/Sansation-light.font'], function($, r, scaleRaph,
config, appObjects) {

  // constructor
  var build = function() {
    // vars aqui são privadas
    var visibleTop = true;
    var dataObj = new Array();
    var visual;

    for (var n in arguments[0]) {
      this[n] = arguments[0][n];
    }
  };

  // public (todas as instâncias)
  build.prototype = {
    getInternalId : function() {
      return internalId;
    },
    getData : function() {
      return this.dataObj;
    },
    setData : function(v) {
      this.dataObj = v;
    },
    parse : function(v) {
      this.dataObj = JSON.parse(v);
    },
    fetchRemoteData : function(params) {
      $.ajax({
        type : "GET",
        url : config.dataService.url,
        async : false,
        data : {
          building : params['building']
        },
        beforeSend : function(x) {
          if (x && x.overrideMimeType) {
            x.overrideMimeType("application/j-son;charset=UTF-8");
          }
        },
        dataType : "json",
        success : function(dataGot) {
          appObjects.suspend = true;
          this.dataObj = data;
          var r = params['paper'];
          var   axisx   =   dataGot.xAxis,   axisy   =   dataGot.yAxis,   data   =
dataGot.data, adjustedX = 70;
          adjustedY = 148;
          width = 600, height = 200, leftgutter = 5, bottomgutter = 20, txt =
{
            stroke : "none",
            fill : "#fff"
```

```
          }, X = (width - leftgutter) / axisx.length, Y = (height -
bottomgutter) / axisy.length, color = $("#chart").css("color");
          max = Math.round(X / 2) - 1;
          r.setStart();
          r.rect(0, 0, 740, 496).attr({
            fill : "#000",
            "fill-opacity" : 0,
            stroke : "none"
          });
          r.rect(55, 108, width + 50, height + 50, 3).attr({
            fill : "#000",
            stroke : "none"
          });

          var titulo = '';

          if (params['building'] == 'global') {
            titulo = r.print(Math.round(width / 2) - 160, adjustedY - 25,
"Acessos aos hotspots do campus na última semana", "Sansation-light",
20).attr({
              stroke : "none",
              fill : "#fff"
            });
          } else {
            titulo = r.print(Math.round(width / 2) - 160, adjustedY - 25,
"Acessos aos hotspots no edificio " + params['building'] + " na última
semana", "Sansation-light", 20).attr({
              stroke : "none",
              fill : "#fff"
            });
          }

          for (var i = 0, ii = axisx.length; i < ii; i++) {
            r.print(adjustedX + leftgutter + 14 + X * (i + .5), adjustedY +
height - 7, axisx[i], "Sansation-light", 10).attr(txt);
          }

          for (var i = 0, ii = axisy.length; i < ii; i++) {
            r.print(adjustedX + leftgutter - 10, adjustedY + Y * (i + .5),
axisy[i], "Sansation-light", 12).attr(txt);
          }

          var o = 0;
          for (var i = 0, ii = axisy.length; i < ii; i++) {
            for (var j = 0, jj = axisx.length; j < jj; j++) {
              //var R = data[o] && Math.min(Math.round(Math.sqrt(data[o] /
Math.PI)*2), max);
              var R = Math.round((data[o] * 11) / dataGot.max);
              //console.log("R: " + R + " data[" + o + "]: " + data[o]);
              if (R == 0 && data[o] > 0) {
                R = 1;
              }
              if (R) {
                (function(dx, dy, R, value) {
                  var color = "hsb(" + [(1 - R / max) * .5, 1, .75] + ")";
                  var dt = r.circle(dx + 80 + adjustedX + R, dy + adjustedY +
10, R).attr({
                    stroke : "none",
                    fill : color
```

```
                  });
                  if (R < 6) {
                    var bg = r.circle(dx + 80 + adjustedX + R, dy + adjustedY
+ 10, 6).attr({
                      stroke : "none",
                      fill : "#000",
                      opacity : .4
                    }).hide();
                  }
                  var adjustTxtX = data[o] > 19 ? 0 : data[o] > 9 ? 2 : 4;
                  var lbl = r.print(dx + 72 + adjustedX + R + adjustTxtX, dy
+ adjustedY + 10, data[o], "Sansation", 12).attr({
                    stroke : "none",
                    fill : "#fff"
                  }).hide();
                  var dot = r.circle(dx + 80 + adjustedX + R, dy + adjustedY
+ 10, max).attr({
                    stroke : "none",
                    fill : "#000",
                    opacity : 0
                  });
                  dot[0].onmouseover = function() {
                    if (bg) {
                      bg.show();
                    } else {
                      var clr = Raphael.rgb2hsb(color);
                      clr.b = .5;
                      dt.attr("fill", Raphael.hsb2rgb(clr).hex);
                    }
                    lbl.show();
                  };
                  dot[0].onmouseout = function() {
                    if (bg) {
                      bg.hide();
                    } else {
                      dt.attr("fill", color);
                    }
                    lbl.hide();
                  };
                })(leftgutter + X * (j + .5) - 60 - R, Y * (i + .5) - 10, R,
data[o]);
              }
              o++;
            }
          }
          this.visual = r.setFinish();
          this.visual.click( function(set) {
            return function() {
              appObjects.suspend = false;
              for ( index = 0; index < set.length; ++index) {
                set[index].hide();
              }
            }
          }(this.visual));
        }
      });
    }
  };
```

```
  return build;
});
```

4. Descrição do WebService

O webservice dummy criado para alimentar a aplicação com dados de utilização de *hotspots WiFi*, contém um método que a cada minuto (cron job) cria eventos na base de dados, para cada edificio, representando novas ligações e ligações perdidas.

Em contexto de produção esse método é desnecessário, uma vez que os dados viriam directamente da aplicação que gere o sistema, no entanto para testes, a criação de eventos pseudo-aleatórios tornou-se essencial para desenvolver a aplicação.

Àparte disso apenas dois métodos de API foram criados dois *endpoints*: getbuildingstats.json e getLastMinute.json. A descrição das chamadas é a seguinte:

getBuildingStats.json

Call

<webservice url>/getbuildingstats.json? Method=GET

Request

Nome	Descrição
building	Identificador do edificio ou “global” para as estatísticas globais do campus

Response

Tipo	Descrição	Formato
Objecto	{“max”: int, “xAxis”: String [24], “yAxis”: String [7], “data”: int [168] }	 Ultimo indice é a hora actual, e no primeiro é (hora actual) – 23 horas, ex: ["01","02","03","04","05","06","07","08","09","10","11","12","13","14","15","16","17","18","19","20","21","22","23","24"] Último indice é o dia actual, primeiro índice é (dia actual) – 6, ex: ["Qui","Sex","Sab","Dom","Seg","Ter","Qua"] Último índice são as ligações actuais, primeiro índice são as ligações que correspondem ao cruzamento da hora do xAxis[0] com o dia do yAxis[0]. O array é linear em

		ordem a X, o que significa que incrementa primeiro os valores em X (horas) e a cada 24 indíces incrementa o valor de y. (data[24] = dados de utilização na hora de xAxis[0] no dia de yAxis[1])

getLastMinute.json

Call

<webservice url>/getLastMinute.json Method=GET

Request

Sem parâmetros

Response

Tipo	Descrição	Formato

Objecto	{“active”: int,
	“new”: int,
	
	“drops”: int,
	“buildings”: Object [{
	“building”: string,
	“conns”: int,
	“percent”: int
	}]
	}
		Total de ligações no campus no último minuto
		Novas ligações no campus no último minuto
		
		Ligações terminadas no campus no último minuto
		array de objectos com dados para o último minuto, para cada edificio
		Identificador do edificio
		
		Total de ligações no edifício no último minuto
		Percentagem das ligações no edifício em relação ao conjunto global