Universidade de Lisboa
Faculdade de Ciências
Departamento de Engenharia Geográfica, Geofísica e Energia

# Criação de um módulo de base geográfica para a gestão comercial de clientes usando serviços Google Maps

Mestrado em Sistema de Informação Geográfica
Tecnologias e Aplicações

**Ana Filipa Neto Sintra Baptista**

Trabalho de Projeto orientado por:

Prof.ª Doutora Cristina Maria Sousa Catita

Eng.º Sandro Gonçalo da Fonseca Batista

**2015**

## RESUMO

No âmbito do projeto curricular do Mestrado em Sistemas de Informação Geográfica - Tecnologias e Aplicações, da Faculdade de Ciências da Universidade de Lisboa, realizado na consultora de negócios *Focus BC*, desenvolveu-se uma solução *web* de base geográfica para a gestão comercial de clientes, usando serviços disponibilizados pela *Google*. A aplicação permite a visualização no mapa e a edição das áreas geográficas afetas aos agentes comerciais de uma empresa, assim como o respetivo mapeamento dos seus clientes. A aplicação permite, ainda, criar e exportar cenários otimizados relativamente a um conjunto de métricas pré-definidas pela empresa, estando estas associadas à carga alocada de cada agente, sendo possível a visualização destas métricas. Para o desenvolvimento, recorreu-se a tecnologias como *PHP* e *JavaScript* e ao *MySQL* como Sistema de Gestão de Base de Dados. O caso de estudo concretizado revelou a otimização de um cenário inicial, usando a aplicação desenvolvida.

**Palavras–chave**: SIG , Solução Web, Inteligência Geográfica, Google Maps API

## ABSTRACT

This document is a report of a web solution developed in the context of an internship realized at *Focus BC*, a business consulting company, with the purpose of obtaining a Master's degree in Geographic Information Systems - Technologies and Applications at Science Faculty of Lisbon University. The objective is to develop a tool, based on *Google* services, to improve commercial management of customers and export optimized scenarios. The application enables visualization and edition of geographical areas allocated to commercial agents of a company as well as customers' mapping. The application also allows creating and exporting optimized scenarios in relation to a set of predefined metrics associated with each agent to compare current scenario with the new scenario and support decisions. It was used *MySQL* as Database Management System and technologies such as *PHP* and *JavaScript* for the development. In the case study was verified an optimization of the initial scenario, using the developed application.

**Keywords**: GIS, Web Solution, Location Intelligence, Google Maps API

## Agradecimentos

Agradeço a todas as pessoas que, de diferentes formas e em alturas distintas, tornaram possível a conclusão deste projeto.

À minha família, aos meus amigos e colegas pelo apoio desde o início desta etapa. Ao Pedro Vilar e à Isabel José também pela revisão.

À Prof.ª Doutora Cristina Catita e ao Eng.º Sandro Batista pela orientação.

A toda a equipa da *Focus BC* pela forma agradável como me receberam e pelos conhecimentos que me transmitiram.

Agradeço, de uma forma especial, à minha mãe, a quem dedico este projeto.

# Conteúdo

## Lista de Figuras



## Lista de Tabelas

# 1. Introdução

## 1.1. Enquadramento do Projeto e Definição do Problema

A gestão comercial de clientes é um exemplo de uma área de atuação em que a informação geográfica tem um papel fundamental. Um gestor que tenha a seu cargo a gestão comercial de clientes precisa de ter ao seu dispor formas rápidas e intuitivas de visualizar e interagir com informação de base geográfica, como por exemplo, as localizações dos clientes e as áreas de atuação dos agentes comerciais, o que constitui informação fundamental para a tomada de decisões. Entende-se por informação geográfica ou georreferenciada a informação que está associada a uma posição sobre a superfície terrestre.

Para além de informação de base geográfica, existem outros tipos de informação com que um gestor comercial tem de lidar, nomeadamente, métricas associadas aos agentes comerciais, pelo que é necessário encontrar soluções que se adequem às necessidades de visualização e manuseamento de informação característicos desta área de atuação.

De acordo com as necessidades e perspetivas da empresa de consultoria de negócios *Focus BC* (*Focus BC*, 2015), foi proposto desenvolver uma aplicação de gestão comercial de clientes. A *Focus BC* oferece serviços e soluções no âmbito de consultoria e de implementação de soluções de negócio, suporte à decisão e inteligência geográfica e é parceira *Google* for Work para o mercado EMEA (Europa, Médio Oriente e África) nas plataformas *Google* Maps for Work (Google, 2015 f) e *Google* Cloud Platform (Google, 2015 c).

Este documento visa descrever o projeto final do Mestrado em Sistemas de Informação Geográfica - Tecnologias e Aplicações, da Faculdade de Ciências da Universidade de Lisboa, desenvolvido durante um estágio curricular na Focus BC e cujos objetivos se descrevem de seguida.

## 1.2. Objetivos do Projeto

O objetivo deste projeto passa por criar uma solução que permita a um gestor visualizar os clientes e os territórios dos agentes comerciais no mapa e ter ao seu dispor métricas associadas à carga alocada de cada agente, podendo assim otimizar a sua força de vendas e servir melhor os seus clientes.

No contexto deste projeto, define-se como agente comercial a pessoa que tem a função de visitar clientes e território como a região criada pelo conjunto de clientes atribuídos a cada agente comercial. O território de cada agente comercial será representado por um polígono que abrange todos os clientes associados, para um determinado ciclo comercial.

As métricas associadas a cada agente são: o número total de clientes e o número total de clientes de cada segmento, a capacidade alocada, o tempo total para visitas a clientes num ciclo comercial, o ACR (*Allocated capacity ratio*) e a área (em $km^2$) do polígono desenhado que representa o território.

Para além de poder visualizar, o gestor deverá ter também a possibilidade de fazer alterações e criar, assim, um novo cenário, designado por "cenário TOBE", otimizado. Num cenário otimizado, o ACR, ou seja, a razão entre a carga alocada aos agentes e a disponibilidade destes para visitar clientes, é mais próxima da unidade do que no cenário atual e há uma maior homogeneidade quanto ao valor deste indicador para cada agente.

Um cenário poderá ser alterado de duas formas: por edição de territórios ou por definição de parâmetros. A edição de territórios consistirá em realizar, pelo menos uma vez, o processo de alteração do agente comercial associado a um cliente, modificando, consequentemente, a área geográfica afeta. A definição de parâmetros deverá passar pela possibilidade de modificar os seguintes parâmetros de entrada: duração do ciclo comercial, frequência de visitas, para o ciclo comercial definido, tempo de visita de cada segmento de clientes e o tempo destinado a visitas a clientes de cada agente. Qualquer alteração feita pelo gestor deverá ter implicações nas métricas, por forma a poder comparar o novo cenário criado com o cenário atual, ou "cenário ASIS".

## 1.3. Metodologia usada para o desenvolvimento do Projeto

Na w*eb* atual, soluções com base em mapas são comuns. (…) A maior parte da informação tem uma localização, permitindo assim que esta seja representada num mapa (Svennerberg, 2010).

A proposta apresentada pela *Focus BC*, a qual foi seguida, foi a de uma solução *web*, usando serviços *Google* Maps (*Google Maps API*) e tendo como *input* dois ficheiros de entrada, de acordo com *templates* pré-definidos, um com dados dos agentes comerciais e outro com dados dos clientes, incluindo a localização destes e o agente comercial associado.

A metodologia seguida começou com o levantamento dos requisitos e o desenho do sistema, tendo-se passado, então, à criação da base de dados de apoio e ao tratamento dos dados.

Após concluídos os métodos para *upload* dos ficheiros de entrada e importação dos dados para a base de dados, implementaram-se métodos para o cálculo das métricas dos agentes comerciais e geração dos polígonos representantes dos seus territórios e definição de parâmetros.

Seguiu-se o desenvolvimento da *interface*, o que implicou a construção da sua estrutura, o desenvolvimento de funcionalidades como a disponibilização de vários tipos de informação, o mapeamento dos clientes e o desenho dos territórios, entre outras funcionalidades secundárias tais como filtros da informação, e a definição dos estilos.

Durante o desenvolvimento do projeto, houve ainda uma fase de melhoramento da *interface*, tanto do *design* como de algumas funcionalidades, finalizando-se o projeto com testes à aplicação.

## 1.4. Contribuição do Projeto

Esta aplicação constitui uma mais valia para a empresa-cliente por ser uma ferramenta de auxílio a quem tem a função de gerir agentes comerciais. O gestor parte de duas folhas de cálculo com dados dos clientes e dos agentes comerciais e passa a ter à sua disposição, sob a forma de tabelas e gráficos, métricas calculadas pela aplicação, que são atualizadas a cada alteração feita, o mapeamento dos clientes e os territórios dos agentes comerciais, o que permite uma melhor consciência da distribuição geográfica destes.

Transformando dados em informação facilmente percetível, colunas com valores numéricos de Latitude e Longitude em mapeamento de clientes e permitindo simulações de novos cenários, esta aplicação torna o processo de tomada de decisões sobre o planeamento de visitas dos agentes comerciais a clientes mais simples e eficaz.

## 1.5. Estrutura do relatório

Após uma breve introdução do problema, com respetiva apresentação dos objetivos do projeto, segue-se uma descrição sucinta, no capítulo 2, sobre os fundamentos teóricos aplicados na implementação deste projeto, em particular sobre aplicações *web* e *webSIG*.

De seguida, no capitulo 3, descreve-se a metodologia na qual constam os vários tipos de requisitos da aplicação e as soluções encontradas para os cumprir.

No capítulo 4, é apresentado um exemplo simples de utilização, onde se mostra um cenário inicial que, após várias operações efetuadas através da aplicação, é otimizado.

Por fim, no capítulo 5, apresentam-se algumas considerações sobre a aplicação e sugerem-se melhorias futuras.

# 2. Fundamentos teóricos

## 2.1. Aplicações w*eb*

Aplicações *web* são aplicações cuja funcionalidade é processada num servidor *web*, e disponibilizada aos utilizadores através de uma rede como a *Internet* ou uma *intranet*.

Os utilizadores usam um *web browser* para correr a aplicação *web*, que sabe como apresentar os dados recebidos do servidor. Pelo contrário, as aplicações *desktop* são baseadas num *thick client* (também chamado de *fat client*) que realiza a maior parte do processamento (Brinzarea e Hendrix, 2009).

Com aplicações na *web*, tudo o que os utilizadores precisam é de um computador com um *web browser* a funcionar e uma ligação à *Internet* ou *intranet* (Brinzarea e Hendrix, 2009). Assim que a aplicação é atualizada no servidor, todos os utilizadores ficam com a versão atualizada (Brinzarea e Hendrix, 2009). Se uma aplicação for bem desenvolvida, corre em qualquer *web browser* moderno (Brinzarea e Hendrix, 2009).

As aplicações *web* facilitam a centralização do armazenamento dos dados. Quando se têm várias localizações a aceder aos mesmos dados, é mais fácil ter todos os dados armazenados num sítio só do que em bases de dados separadas para cada localização. Desta forma, também se evitam problemas relacionados com sincronização e reduzem-se riscos de segurança. (Brinzarea e Hendrix, 2009)

## 2.2. HTTP e HTML

O *HyperText Transfer Protocol* (HTTP) é suportado por todos os *web browsers* e é muito eficiente a realizar a tarefa para que foi concebido – transferir conteúdo *web* simples. Sempre que se requisita uma página *web* usando um *web browser*, o protocolo HTTP é assumido.

O tipo de documento *standard* da *Internet* é *HyperText Markup Language* (HTML), que os *web browsers* conseguem interpretar, analisar e disponibilizar. A linguagem HTML descreve o formato e o conteúdo dos documentos, que basicamente são compostos por texto e imagens estáticos. O HTML não foi concebido para desenvolver aplicações *web* complexas com conteúdo interativo ou *interfaces user-friendly* (Brinzarea e Hendrix, 2009).

A cada parte ou divisão de uma página *web* dá-se o nome de *div* e ao atribuir um identificador único a uma *div*, é possível disponibilizar informação no espaço (*pixeis*) reservado por ela, espaço este que é definido pelo seu estilo.

Um mecanismo simples para adicionar estilos (como por exemplo, fontes, cores ou espaçamento) a documentos da w*eb* é o Cascading Style Sheets (CSS) (Web Consortium, 2015).

As transferências HTTP ocorrem sempre entre um cliente *web* (o *software* que faz o pedido, como um *web browser*) e o servidor *web* (o *software* que responde ao pedido), como o Apache ou o IIS (Brinzarea e Hendrix, 2009).

O Apache (The Apache Software Foundation, 2015) é um servidor *web* livre que suporta PHP, que por sua vez disponibiliza métodos para interagir com o MySQL (Oracle Corporation, 2015), um Sistema de Gestão de Bases de Dados Relacionais (Relational Database Management Systems, RDBMS) com ferramentas para armazenar e gerir dados.

A combinação HTTP-HTML é muito limitada, no sentido em que apenas possibilita a transferência de conteúdo estático da *Internet* (Brinzarea e Hendrix, 2009).

## 2.3. Tecnologias *server-side* e *client-side*

As tecnologias do lado do servidor (*server-side technologies*) permitem ao servidor *web* fazer mais do que simplesmente devolver os ficheiros HTML pedidos, como realizar cálculos complexos, programação orientada a objetos, trabalhar com bases de dados e muito mais.

O PHP é uma das tecnologias usadas para implementar a lógica do lado do servidor e é também uma linguagem de *script*. No lado do servidor, normalmente é necessário também um servidor de base de dados para gerir os dados, como por exemplo o MySQL (Brinzarea e Hendrix, 2009). Mesmo usando PHP, o *browser* continua a apresentar conteúdo *web* estático, enfadonho e pouco "inteligente".

A necessidade de funcionalidades mais inteligentes e poderosas no cliente *web* gerou um outro conjunto de tecnologias, as chamadas tecnologias do lado do cliente (*client-side technologies*). Os *browsers* modernos sabem como analisar mais do que um simples HTML.

As várias tecnologias *client-side* diferem em muitos aspetos, começando pela forma como são carregadas e executadas pelo cliente *web*. *JavaScript* é uma linguagem de *script*, cujo código é escrito em texto simples e pode ser incorporado em páginas HTML para capacitá-las. Quando um cliente solicita uma página HTML, esta página pode conter *JavaScript*. O *JavaScript* é suportado por todos os *browsers* modernos, sem exigir a instalação de novos componentes no sistema.

O *JavaScript* é suportado pela maioria dos clientes *web* em qualquer plataforma e tem algumas capacidades orientadas a objetos (Brinzarea e Hendrix, 2009).

Se um utilizador escrever o seu email num formulário de forma incorreta, não é necessário carregar novamente uma página inteira, através de uma validação no lado do cliente, o que é possível usando *JavaScript*.

A combinação de HTML com uma tecnologia *server-side* e uma tecnologia *client-side* permite a criação de soluções *web* muito poderosas. Esta combinação de tecnologias exige, contudo, que

em cada requisição de novos dados ao servidor, seja necessário um novo pedido HTTP para recarregar a página, o que bloqueia a atividade do utilizador (Brinzarea e Hendrix, 2009).

## 2.4. AJAX

A tecnologia AJAX (Asynchronous *JavaScript* and XML) possibilita a criação de aplicações *web* mais versáteis e interativas, permitindo que páginas *web* façam chamadas assíncronas ao servidor de forma transparente, enquanto o utilizador continua a trabalhar (Brinzarea e Hendrix, 2009).

Através desta tecnologia, o *JavaScript* do lado do cliente pode atualizar certas partes de uma página sem causar o recarregamento de toda a página, ao fazer chamadas ao servidor e recuperar dados.

Não são necessárias instalações de módulos extra para correr um *website* com AJAX, uma vez que as tecnologias que estão na sua base, como *JavaScript* e Extensible Markup Language (XML) (World Wide Web Consortium, 2008), já estão implementadas em todos os *web browsers* modernos.

Na base do AJAX está:

- O *JavaScript*, para construir a funcionalidade do lado do cliente, recorrendo também ao Document Object Model (DOM) que permite manipular partes da página HTML (*divs*);
- O objeto XMLHttpRequest permite ao código *JavaScript* do lado do cliente pedir uma página ao servidor, de forma assíncrona, através de um pedido HTTP para um ficheiro ou *script* localizado no servidor;
- Uma tecnologia do lado do servidor para lidar com os pedidos do cliente *JavaScript*.

Na comunicação cliente-servidor, são enviados e interpretados dados. Ao aceder ao servidor, usando um objeto XMLHttpRequest, o *script* cliente pode enviar pares nome-valor através dos métodos GET e POST. O primeiro método requer dados de um recurso especificado e o segundo submete dados para serem processados por um recurso especificado.

A resposta do *script* do servidor, via HTTP, segue um formato que o código JavaScript no lado do cliente pode analisar, como XML ou JavaScript Object Notation (JSON) (Brinzarea e Hendrix, 2009).

A Figura 1 é uma representação do que acontece quando um utilizador faz um pedido a uma página *web* que tem AJAX.

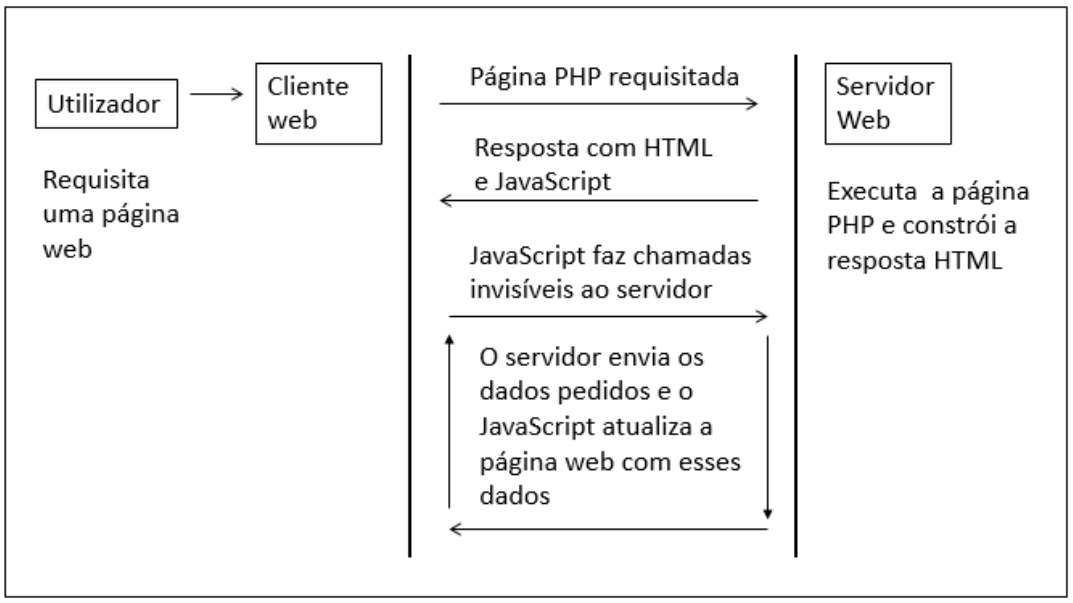



***Figura 1*** *- Ilustração de um pedido AJAX*

## 2.5. *Service Oriented Architecture*

O *Service Oriented Architecture* (SOA) é um estilo de arquitetura de *software* que modela as várias componentes do sistema pelas funcionalidades que implementam em forma de serviços (w*eb* s*ervices*). A implementação de aplicações complexas envolve a interligação entre as várias componentes através da invocação destes serviços. W*eb wervices* podem-se considerar como Interface de Programação de Aplicações (*Application Programming Interface*, API), que permite esta interligação entre aplicações de *software* heterogéneas através do protocolo HTTP usado na *web*. As mensagens trocadas pelos w*eb services* usam o formato XML, e podem seguir diferentes *standards*, tais como SOAP, REST, ou JSON (Couto, 2012).

Uma API é um conjunto de métodos e ferramentas que podem ser usadas para desenvolver aplicações de *software*.

## 2.6. Aplicação *webSIG*

As aplicações *webSIG* têm ganho popularidade devido à sua simplicidade e facilidade de utilização. Através de *browsers*, as aplicações *webSIG* disponibilizam mapas com camadas de informação como edificado, florestas, estradas, trânsito, entre outros. Além disso, os dados geográficos podem ser analisados e grandes quantidades de dados podem ser visualizadas em ecrãs pequenos por arrastamento (*panning*) do mapa e alterações de *zoom* (*zooming*) (Khan e Adnan, 2010). Pode haver, simultaneamente, interação de utilizadores e atualização de dados. Todos os utilizadores vêem ao mesmo tempo as atualizações e os dados espaciais são amplamente acessíveis a partir de qualquer local, usando tecnologia *web* (Zheng, Soomro e Pan, 2000).

O processo de conceção, implementação, geração e disponibilização de mapas, de dados geoespaciais e de serviços relacionados com mapas na *web é o webmaping*. O termo *webSIG*, muitas vezes usado como sinónimo de *web mapping*, traz funcionalidades adicionais para exploração e análise espacial suportadas por funções de geoprocessamento. Nas últimas duas décadas, *web mapping* e *webSIG* têm evoluído drasticamente desde uma simples apresentação estática de dados em mapas até à visualização e análise de dados de forma dinâmica e semi-dinâmica e, mais recentemente, para serviços de dados e serviços relacionados com mapas. Impulsionada pelas sempre disponíveis novas tecnologias da *web* e pelas necessidades das comunidades geoespaciais, tem sido desenvolvida uma ampla gama de tecnologias de *web mapping* e *webSIG* por agências governamentais, empresas, instituições de investigação e comunidades abertas de utilizadores. Estas tecnologias têm sido usadas para implementar aplicações para:

1) Acesso e divulgação de dados espaciais;
2) Exploração e visualização de dados espaciais;
3) Processamento, análise e modelação de dados espacial;
4) Apoio à decisão de base espacial de forma colaborativa, utilizando Sistema de Informação Geográfica (SIG) de participação pública;
5) Integração dos serviços geoespaciais baseados na *web* em processos empresariais tradicionais.

Quanto ao desenvolvimento de aplicações *webSIG*, há uma vasta coleção de APIs abertas e disponíveis, como a Google API, a *Bing Maps API*, a *Yahoo Maps API* e a *OS OpenSpace API*, que facilitam o acesso aos serviços e aos dados e permitem o desenvolvimento facilmente personalizável de aplicações que combinam conteúdos de vários recursos online (Li and Gonf 2008) (Li, Dragicevic e Bert, 2011 ).

# 3. Metodologia

## 3.1. Descrição das várias fases da metodologia

Antes de começar o desenvolvimento da aplicação, foi necessário elaborar o Caderno de Requisitos do Sistema, o qual visa enumerar os requisitos que, de algum modo, tenham influência na definição funcional do sistema a implementar e que constitui uma importante ferramenta de trabalho para a posterior elaboração dum segundo documento, o Caderno de Análise e Desenho, bem como para o respetivo desenvolvimento e testes da aplicação, que deverão estar em conformidade com os requisitos nele descritos.

O Caderno de Requisitos do Sistema engloba os Requisitos Funcionais e de *Interface*, onde se define o que o sistema terá que fazer e quais os requisitos ao nível da *interface* entre o sistema e o utilizador, e os Requisitos Técnicos, onde se especificam requisitos como infraestruturas e *software*, ou qualquer outro requisito técnico que tenha influência na definição funcional do sistema a implementar.

O Caderno de Análise e Desenho do Sistema constitui, em conjunto com o Caderno de Requisitos do Sistema, a fonte de todas as especificações funcionais e técnicas do sistema a implementar. Este documento mostra de que forma se respondeu aos requisitos técnicos e funcionais, encontrando-se dividido em duas partes: Análise Funcional e Desenho Técnico.

Quanto ao desenvolvimento da aplicação, optou-se por começar pelas funcionalidades mais simples, partindo depois para funcionalidades mais complexas, garantindo sempre que as funcionalidades mais importantes seriam concluídas, e só no final seriam desenvolvidas as funcionalidades menos relevantes, como por exemplo, a possibilidade de pesquisa de clientes.

Assim, primeiramente procedeu-se à criação da base de dados de apoio, seguindo-se o carregamento da mesma com dados extraídos dos ficheiros de entrada já tratados. Após concluída esta funcionalidade, acrescentou-se um método para *upload* de ficheiros a partir de uma pasta do computador do utilizador da aplicação

Tendo os métodos de *upload* e tratamento de ficheiros, de ligação à base de dados e respetivo carregamento concluídos, passou-se à fase de construção da *interface* e de uso de serviços Google Maps. Para tal, criou-se uma estrutura em HTML, que acabou por sofrer algumas alterações, onde inicialmente apenas se mostrava o painel de *upload* dos ficheiros de entrada e um mapa, a partir do primeiro pedido feito à *Google Maps JavaScript API*.

Passo a passo, foram sendo acrescentadas funcionalidades à aplicação, a par de pedidos mais complexos à *Google Maps API*, por forma a mostrar o mapeamento dos clientes, os polígonos representantes dos territórios dos agentes comerciais e métricas em forma de tabelas e gráficos.

Para mostrar os territórios dos agentes comerciais no mapa, é necessária a sua geração prévia.

Acrescentaram-se as definições de parâmetros e a possibilidade de mudar o agente comercial associado aos clientes, ou seja, a edição de territórios com criação de um novo cenário cuja exportação se tornou também possível, acompanhada da exportação das definições.

Foram ainda acrescentadas funcionalidades de pesquisa de clientes e filtros da informação mostrada por agente comercial.

Realizaram-se alguns testes para encontrar e corrigir possíveis erros e avaliar a usabilidade da aplicação e fizeram-se melhorias do funcionamento e do *design*.

## 3.2. Componentes da Aplicação

Na figura 2, apresenta-se um diagrama com o fluxo da aplicação.

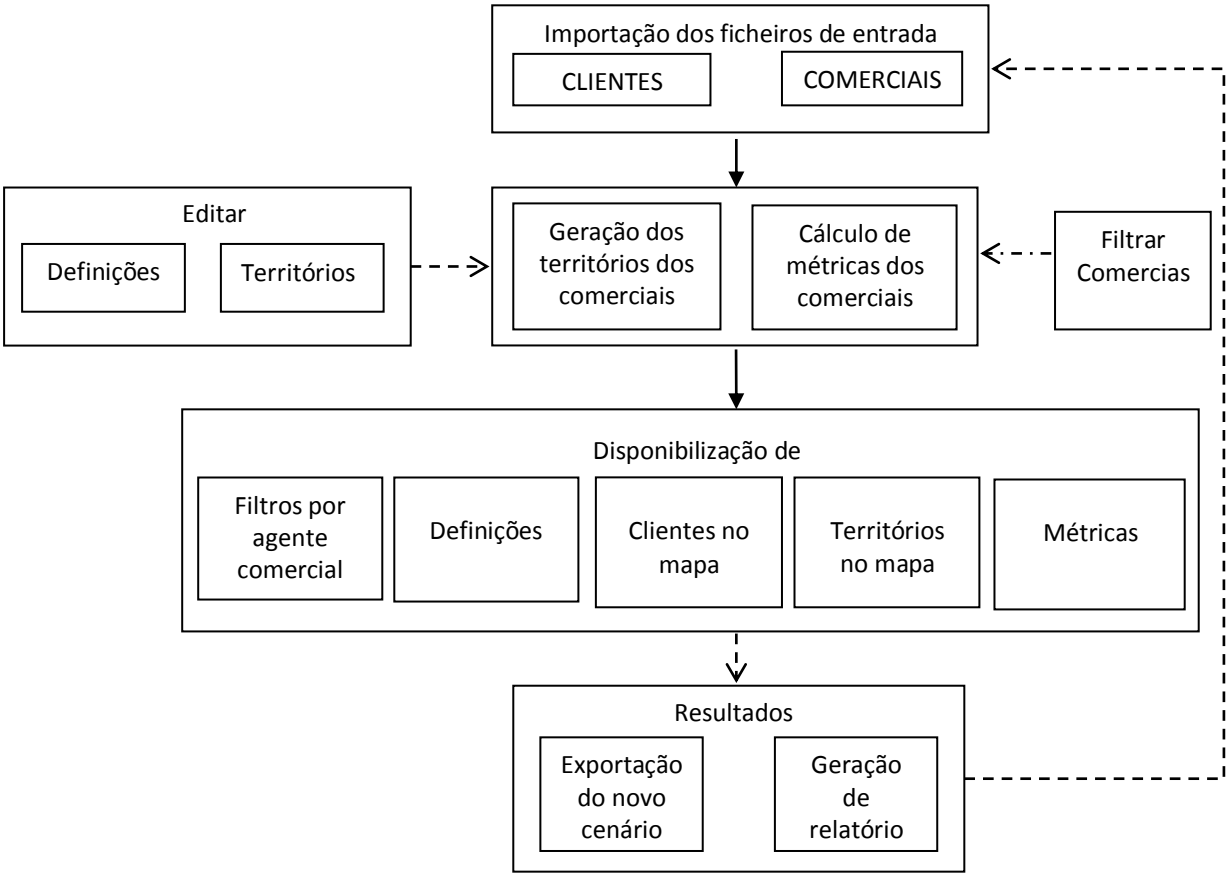



***Figura 2** - Diagrama do fluxo da aplicação*

A aplicação segue a seguinte ordem de processos:

1. Importação de ficheiros com os dados dos clientes e dos agentes comerciais;

2. a) Geração dos territórios dos agentes comerciais;

   b) Cálculo de métricas dos agentes comerciais;

3. a) Disponibilização dos clientes no mapa;

   b) Disponibilização dos territórios criados no mapa;

   c) Disponibilização de métricas relativas aos agentes comerciais;

   d) Disponibilização de parâmetros editáveis (definições);

   e) Disponibilização de um filtro por agentes comerciais.

Após a aplicação de um filtro por agentes comerciais, são executados novamente os passos 2. a) e b) e 3. a), b) e c).

A edição de definições e de territórios no mapa, origina nova execução dos passos 2. a) e b) e 3. a), b), c) e d).

Após exportação do novo cenário criado aquando da edição dos territórios, os ficheiros de clientes e de agentes comerciais exportados podem ser novamente importados, voltando a ser executados todos os passos do fluxo da aplicação.

## 3.3. Requisitos Funcionais e de *Interface*

Seguidamente, apresentam-se os Requisitos Funcionais e de *Interface* da aplicação.

**Requisitos gerais**

Pretende-se desenvolver um módulo para visualização e alteração de territórios de agentes comerciais com base no mapa e criação de um novo cenário.

Este módulo recebe como *input* dois ficheiros com a extensão .xls (*Microsoft Excel 1997 to 2003 workbook file*) ou .xlsx (*Microsoft Excel Open XML workbook file*), um com informação dos clientes e outro com informação dos agentes comerciais. O primeiro tem informação geográfica, ou seja, as localizações dos clientes, Latitude e Longitude, em WGS84 (em formato "gg,dddddd"). A aplicação disponibiliza uma caixa para escolha de ficheiros e um botão para os carregar.

Com base nos dados de entrada, é então possível criar o cenário atual a partir do qual são geradas as métricas (resultados) para os agentes comerciais. É criado ainda o novo cenário, inicialmente como cópia do cenário atual.

**Requisitos de edição de territórios**

A edição de territórios é feita no mapa, através de uma janela de informação para cada cliente, que aparece ao clicar no marcador respetivo, com uma caixa de seleção com os nomes dos agentes comerciais e um botão para guardar a alteração. As métricas são também atualizadas.

**Requisitos de visualização**

Como requisitos de visualização pretende-se que ambos os cenários sejam mostrados no mapa, assim como as tabelas e os gráficos com métricas. Os cenários englobam os clientes, representados por marcadores, e os territórios dos agentes, representados por polígonos, ambos com cores diferenciadas.

É possível abrir/fechar o painel das definições e dos filtros, contraindo/expandido a área do mapa e dos resultados. Em termos da visualização do mapa, estão disponíveis opções de *panning* (arrastar), *zoom-in* (aproximar) e *zoom-out* (afastar).

**Requisitos de filtros**

Pretende-se ainda que sejam disponibilizados filtros para seleção e visualização personalizada da informação, tanto a do mapa, como a dos resultados, por agente comercial ou múltiplos agentes comerciais, através de uma caixa de seleção e de um botão para aplicar o filtro. Para a desativação do filtro, há um botão apropriado.

**Requisitos das definições**

A aplicação deverá ainda permitir a alteração dos parâmetros relativos ao ciclo comercial, à segmentação dos clientes e aos agentes comerciais, através de caixas de preenchimento, caixas com opção de seleção e de um botão para guardar as alterações feitas e atualizar os resultados.

**Requisitos da informação temática**

O mapa deverá aparecer centrado nos territórios, com um *zoom* que permita a visualização destes. As camadas de informação visíveis são os territórios de ambos os cenários e os clientes no novo cenário. Deverá existir ainda a possibilidade de alternar entre visualizar no mapa ou remover os marcadores dos clientes e os polígonos dos territórios ASIS e TOBE, através de botões.

A aplicação deverá ainda contemplar a possibilidade de visualização dos mapas no contexto proveniente do Google Maps (Google, 2015 d), tais como Mapa de Estradas com Toponímia, Ortofotomapas e *Street View*, sendo que os dois primeiros serão obtidos com um botão apropriado, e o último com o ícone do *Street View*.

**Requisitos de pesquisa**

Pretende-se que a aplicação apresente a possibilidade de pesquisa de pontos de interesse, ruas, localidades, etc. A pesquisa é feita usando a caixa de pesquisa da Google.

A pesquisa de clientes, por nome, através de uma caixa de inserção de texto e de um botão para iniciar a pesquisa, com criação de marcadores no mapa com a localização dos clientes cujo nome satisfaça a *query* deverá também ser contemplada. A vista do mapa e o *zoom* serão ajustados aos clientes encontrados. Deverá ainda haver um botão para a remoção dos resultados da pesquisa.

**Requisitos de exportação de ficheiros .xlsx**

A aplicação a desenvolver deverá contemplar a geração de três ficheiros em formato .xlsx, a partir de um botão com um ícone apropriado, um com os dados dos clientes, outro com os dados dos agentes comerciais e um terceiro com as definições, todos de acordo com as alterações efetuadas.

Os ficheiros de clientes e de comerciais terão incluído no seu nome o momento em que foram importados. O nome do ficheiro das definições incluirá o momento em que foi exportado. Exemplo de nome de um ficheiro de clientes exportado no dia 22 de Junho de 2015, às 22h 35min 05s : CLIENTES_20150622_223505.

**Requisitos de geração de um relatório**

Será gerado um *layout*, a partir de um botão, com base em *template* pré-definido, com abertura de nova janela noutro separador. O *layout* deverá conter tabelas e gráficos com os parâmetros definidos e os resultados de ambos os cenários, bem como botões para imprimir o relatório e para voltar à aplicação e fechar o separador do relatório. O *layout* poderá ser impresso ou guardado em formato PDF.

**Requisitos de segurança e de monitorização do serviço**

A segurança e a monitorização do serviço serão realizados através da plataforma em que este módulo será integrado.

## 3.4. Requisitos Técnicos

A aplicação a desenvolver apresenta como requisitos técnicos, os seguintes:

- Estão previstas integrações do módulo Gestão de Territórios com sistemas externos;
- O sistema será desenvolvido tendo por base a *Google Maps API*;
- Os dados serão alojados em *MySQL*, num servidor disponibilizado pela *Focus BC*;
- As coordenadas dos clientes devem ser fornecidas em formato WGS84 - gg,dddddd.

A escolha da *Google Maps API* está relacionada com a parceria existente entre a *Focus BC* e a *Google*, não tendo havido portanto uma fase de pesquisa e comparação de APIs para selecionar a mais indicada de acordo com as necessiadades deste projeto.

## 3.5. Análise Funcional

A Análise Funcional é constituída pela identificação dos atores do sistema e pelos casos de uso. De seguida, descrevem-se alguns detalhes a contemplar para a análise funcional da aplicação.

### *3.5.1. Atores do Sistema*

Um ator, em relação a um sistema, representa alguém ou alguma coisa do ambiente envolvente que interage com o sistema. No âmbito deste módulo, foi identificado apenas um ator, o Gestor. O Gestor é o utilizador da empresa (cliente) com acesso ao módulo Gestão de Territórios e que irá editar, pesquisar, filtrar e obter resultados.

### *3.5.2. Casos de Uso*

Um caso de uso é uma descrição de um conjunto de sequências de ações, incluindo variantes, que um sistema realiza para produzir um resultado observável com valor para um ator.

Para cada caso de uso descrito, são referidos os seguintes dados:

- Nome: nome pelo qual é identificado ao longo do sistema o caso de uso em questão;
- Descrição: contém a sequência das operações e sua descrição;
- Ator: ator ao qual se aplica o caso de uso;
- Pré-requisitos: condições que têm de estar satisfeitas para que o caso de uso se aplique;
- *Input*: *inputs* no sistema pelo Ator ao longo do caso de uso;
- *Output*: *outputs* do sistema para o Ator ao longo do caso de uso;
- Casos de uso associados: casos de uso que se encadeiem com o que está a ser descrito;
- Requisitos abrangidos: requisitos do sistema que são abrangidos pelo caso de uso. É para dar resposta a estes requisitos que o sistema vai implementar o caso de uso em questão.

Para o desenvolvimento deste projeto, foram encontrados os seguintes casos de uso:

C.1 - Submissão de ficheiros a partir da página inicial da aplicação;

C.2 - Visualização dos clientes e dos territórios no mapa e das métricas;

C.3 - Alteração de definições por segmentação de clientes e do ciclo comercial;

C.5 - Pesquisa de clientes;

C.6 - Filtro de agentes comerciais;

C.7 - Pesquisa utilizando a caixa de pesquisa *Google*;

C.8 - Navegação no mapa;

C.9 - Edição de territórios;

C.10 - Expansão do mapa;

C.11 - Navegação em modo *Street View*;

C.12 - Alternar mapa de base;

C.13 - Remoção dos clientes;

C.14 - Remoção dos territórios;

C.15 - Visualização e impressão do relatório;

C.16 - Exportação de ficheiros.

Para não tornar este documento demasiado exaustivo, encontram-se descritos, nas Tabelas 1, 2 e 3, apenas alguns dos casos de uso identificados para o sistema.

***Tabela 1** - Caso de uso 1 - Submissão de ficheiros a partir da página inicial da aplicação*

| | |
|---|---|
| **Nome** | **C.1 – Submissão de ficheiros a partir da página inicial da aplicação** |
| **Descrição** |  <br>O utilizador deverá aceder à aplicação através do seu *browser*, escolher os ficheiros com informação dos clientes e dos agentes comerciais, através dos botões "Escolher ficheiro" e pressionar o botão "*Upload*", no canto superior esquerdo.<br>O utilizador pode navegar no mapa e utilizar a caixa de pesquisa da *Google*, mesmo sem submeter os ficheiros de entrada. |
| **Ator** | Gestor |
| **Pré-requisitos** | O utilizador deverá aceder à aplicação através do seu *browser*. |
| **Input** | 2 ficheiros .xls ou .xlsx: CLIENTES e COMERCIAIS<br>Clientes Escolher ficheiro CLIENTES.xlsx<br>Comerciais Escolher ficheiro COMERCIAIS.xlsx<br>Upload |
| **Output** | Verificação dos nomes dos ficheiros e dos formatos.<br>Se houver irregularidades nos nomes ou formatos dos ficheiros, são mostradas mensagens de erro com o problema encontrado. Exemplo: |

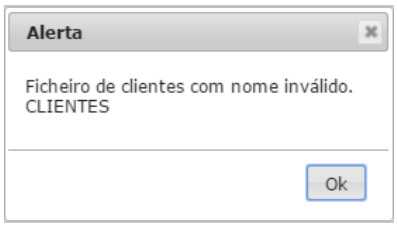
| | |
|---|---|
| |  <br>Se as verificações anteriores forem validadas e os ficheiros estiverem de acordo com os *templates* definidos:<br>- são disponibilizados no mapa os clientes e os territórios dos agentes comerciais;<br>- são disponibilizadas definições e métricas. |
| **Casos de uso associados** | Todos |
| **Requisitos abrangidos** | Requisitos gerais |

***Tabela 2** - Caso de uso 9 - Edição de territórios*

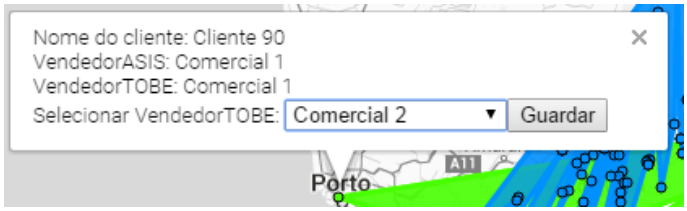
| | |
|---|---|
| **Nome** | **C.9 – Edição de territórios** |
| **Descrição** | Alteração do agente comercial associado a um cliente, através de janela de informação, originando alterações de territórios. |
| **Ator** | Gestor |
| **Pré-requisitos** | O utilizador deverá aceder à aplicação através do seu *browser* e submeter os ficheiros de entrada. |
| **Input** | Clicar com o botão esquerdo do rato sobre o marcador do cliente que se quer mudar de território, selecionar na janela de informação aberta o novo agente comercial e clicar no botão "Guardar".<br>  <br>Repetir o processo para todos os clientes cujo agente comercial associado se pretende alterar. |
| **Output** | Alterações nas formas do territórios de onde e para onde se mudou o cliente.<br>Alteração da cor do marcador do cliente passando a ter a cor do comercial do novo cenário.<br>Alterações nos resultados do novo cenário. |
| **Casos de uso associados** | C.2 |
| **Requisitos abrangidos** | Requisitos de edição de territórios |

***Tabela 3** - Caso de uso 15 - Visualização e impressão do relatório*

| **Nome** | **C.15 – Visualização e impressão do relatório** |
|---|---|
| **Descrição** | Geração de um relatório com base num *template* pré-definido, com resultados de ambos os cenários, que pode ser impresso ou guardado em formato PDF. |
| **Ator** | Gestor |
| **Pré-requisitos** | O gestor deverá abrir a aplicação no seu *browser* e submeter os ficheiros de entrada. |
| **Input** | Clicar sobre o botão  (no canto inferior direito) para imprimir relatório. |
| **Output** | É aberto um novo separador com o relatório gerado.<br>Ao carregar no botão "SAIR", o novo separador é fechado, voltando à aplicação.<br>Ao carregar "IMPRIMIR", é mostrada uma pré-visualização de impressão e opções de impressão. |
| **Casos de uso associados** | C.1 |
| **Requisitos abrangidos** | Requisitos da geração de um relatório. |

No total foram desenvolvidos dezasseis casos de uso que, resumidamente, cobrem funcionalidades de submissão e exportação de ficheiros, visualização e interação com informação de clientes e agentes comerciais, edição de cenários, geração de um relatório e navegação no mapa.

## 3.6. Desenho técnico

O desenho técnico é constituído pela especificação de ecrãs, modelo de dados e arquitetura da aplicação.

### *3.6.1. Especificação de ecrãs*

Neste subcapítulo, apresentam-se alguns dos ecrãs com que o utilizador se depara ao interagir com a aplicação.

**Ecrã inicial**

Ao aceder ao módulo, surge ao utilizador o ecrã mostrado na Figura 3, onde pode proceder ao *upload* dos ficheiros de entrada, no canto superior esquerdo, e dar início às funcionalidades da aplicação.

A partir deste ecrã, o utilizador pode navegar no mapa e utilizar a caixa de pesquisa da Google.

Dicidiu-se estilizar o mapa em tons de cinza de modo a dar mais realce à informação sobreposta.

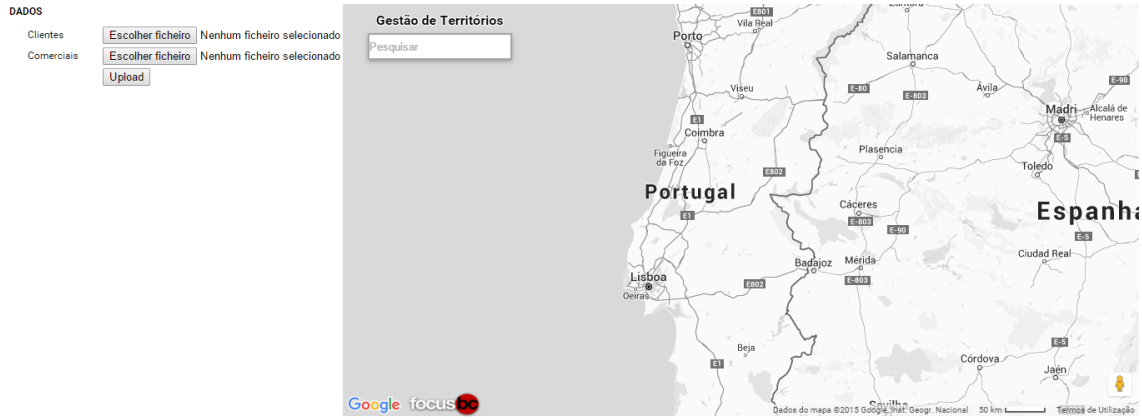



**Figura 3** - Ecrã inicial

### Ecrã após *upload* dos ficheiros com sucesso

Quando o *upload* dos ficheiros de entrada decorre sem problemas, o utilizador passa a poder usar todas as funcionalidades da aplicação a partir de um ecrã como o mostrado na Figura 4, que apresenta variações dependentes dos dados, tais como o centro e o *zoom* iniciais do mapa ou os polígonos desenhados.

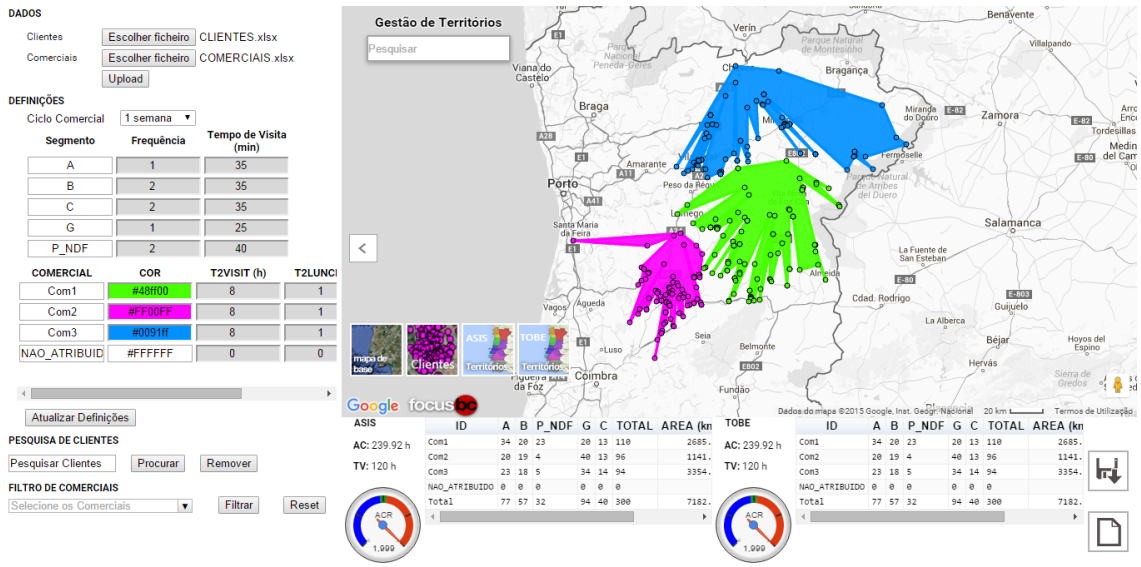



***Figura 4*** *- Ecrã após upload dos ficheiros com sucesso*

### Pesquisa por nome de clientes

Após realizada uma pesquisa por nome ou parte do nome de clientes, caso sejam encontrados clientes que correspondam à *query*, surge um ecrã como o que se apresenta na Figura 5, no qual se podem ver marcadores no mapa.

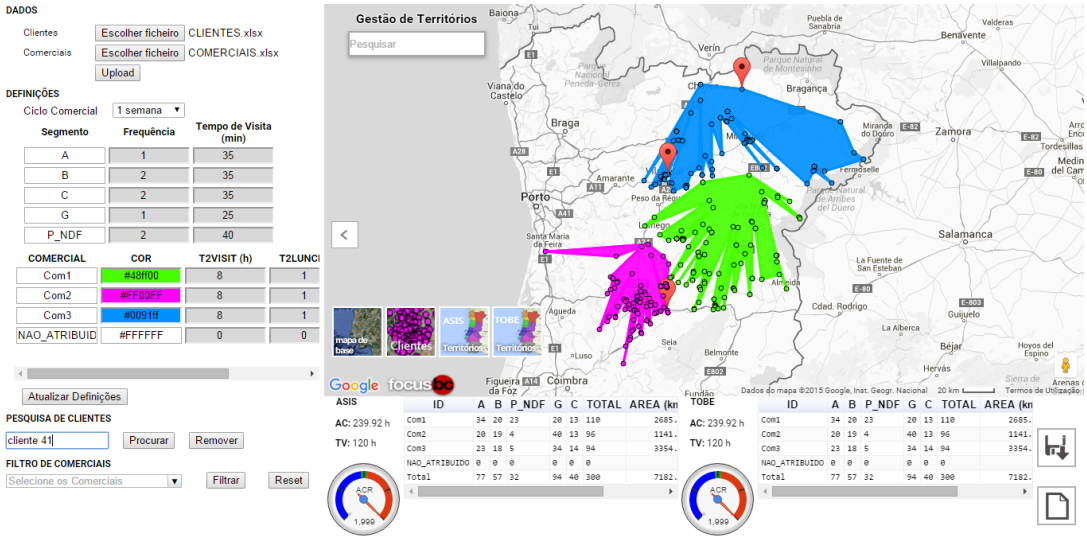
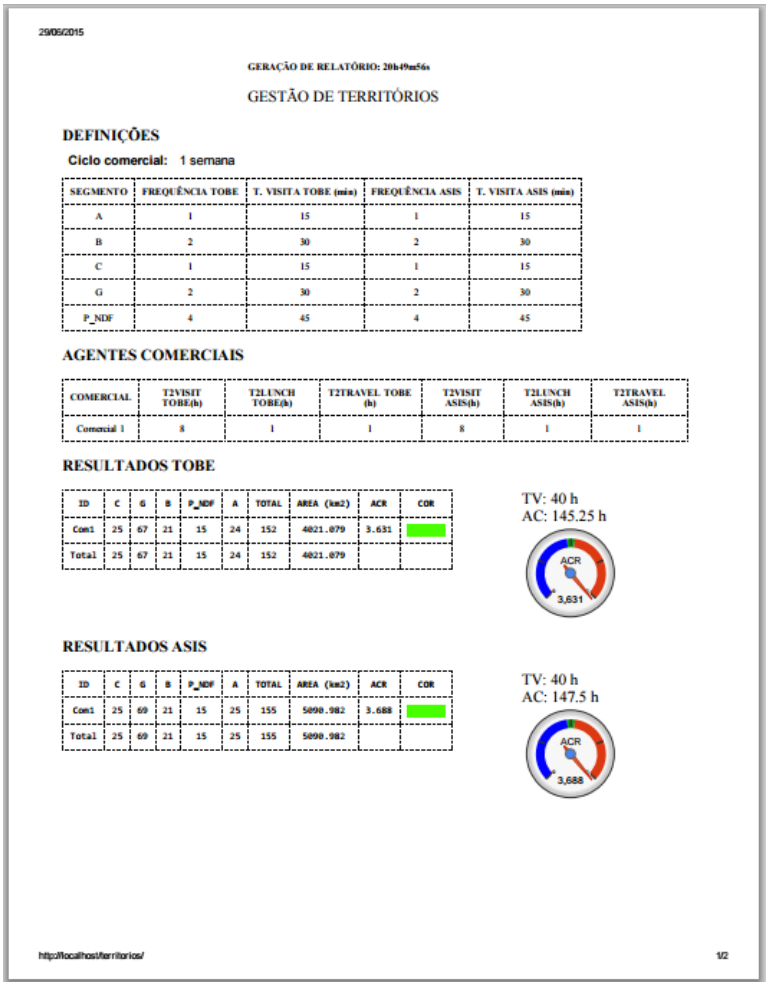



***Figura 5*** *- Pesquisa por nome de clientes*

**Exemplo de geração e impressão de um relatório**

Ao proceder à geração de um relatório e após escolher a opção de imprimir, é disponibilizado ao utilizador um relatório como o que se mostra na Figura 6.



29/05/2015

GERAÇÃO DE RELATÓRIO: 20h49m56s

GESTÃO DE TERRITÓRIOS

**DEFINIÇÕES**

**Ciclo comercial:** 1 semana

| SEGMENTO | FREQUÊNCIA TOBE | T. VISITA TOBE (min) | FREQUÊNCIA ASIS | T. VISITA ASIS (min) |
|---|---|---|---|---|
| A | 1 | 15 | 1 | 15 |
| B | 2 | 30 | 2 | 30 |
| C | 1 | 15 | 1 | 15 |
| G | 2 | 30 | 2 | 30 |
| P_NDF | 4 | 45 | 4 | 45 |

**AGENTES COMERCIAIS**

| COMERCIAL | T2VISIT TOBE(h) | T2LUNCH TOBE(h) | T2TRAVEL TOBE (h) | T2VISIT ASIS(h) | T2LUNCH ASIS(h) | T2TRAVEL ASIS(h) |
|---|---|---|---|---|---|---|
| Comercial 1 | 8 | 1 | 1 | 8 | 1 | 1 |

**RESULTADOS TOBE**

| ID | C | G | B | P_NDF | A | TOTAL | AREA (km2) | ACR | COR |
|---|---|---|---|---|---|---|---|---|---|
| Com1 | 25 | 67 | 21 | 15 | 24 | 152 | 4021.079 | 3.631 | |
| Total | 25 | 67 | 21 | 15 | 24 | 152 | 4021.079 | | |

TV: 40 h
AC: 145.25 h



**RESULTADOS ASIS**

| ID | C | G | B | P_NDF | A | TOTAL | AREA (km2) | ACR | COR |
|---|---|---|---|---|---|---|---|---|---|
| Com1 | 25 | 69 | 21 | 15 | 25 | 155 | 5090.982 | 3.688 | |
| Total | 25 | 69 | 21 | 15 | 25 | 155 | 5090.982 | | |

TV: 40 h
AC: 147.5 h



http://localhost/territorios/ 1/2

***Figura 6*** *- Exemplo de um relatório*

### Edição de territórios

Após clicar no marcador correspondente ao cliente cujo agente comercial se pretende alterar, surge uma janela de informação a partir da qual se altera o agente comercial, originando assim alterações de territórios. Na Figura 7, exemplifica-se o ecrã para edição de territórios.

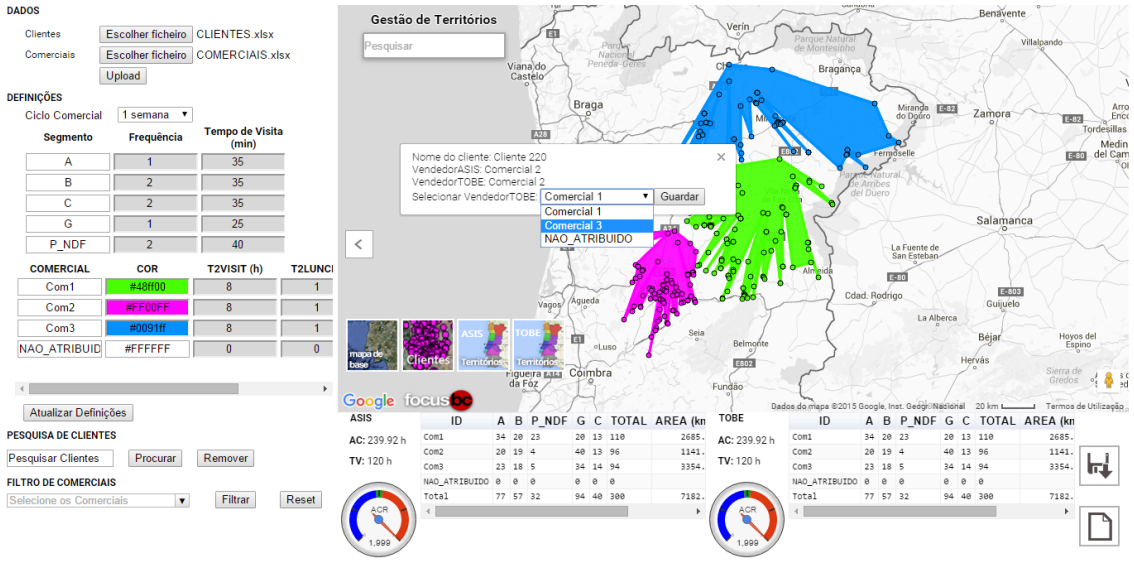



***Figura 7*** *- Edição de territórios*

### Navegação com o Google S*treet View*

Na Figura 8, mostra-se um serviço disponibilizado pela *Google Maps API* que se integrou nesta aplicação, o modo s*treet view* ou "vista de rua". Esta funcionalidade permite dar maior realismo à navegação no mapa.

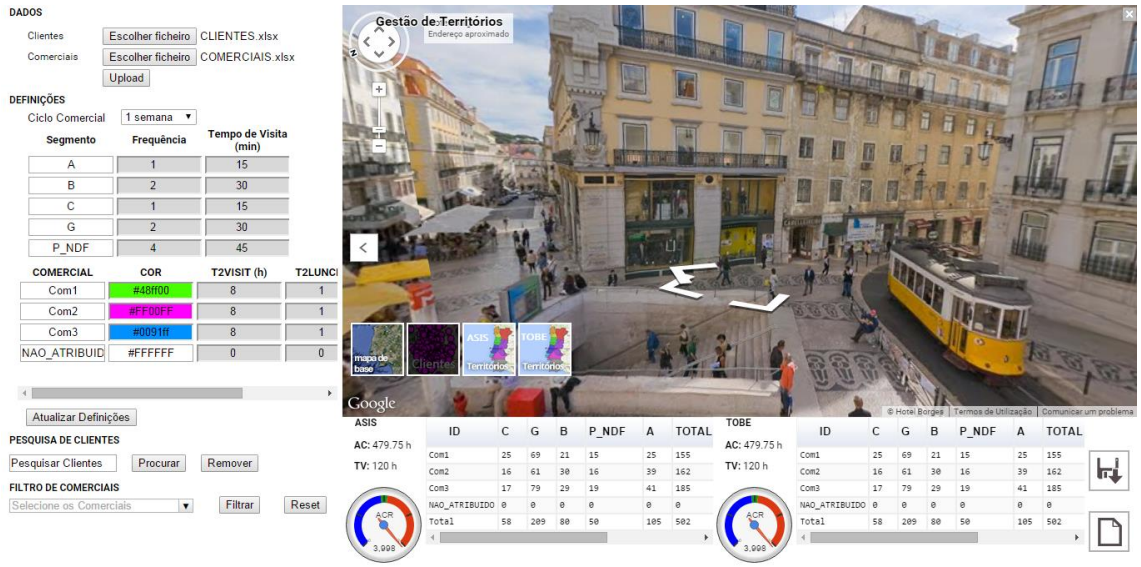



***Figura 8*** *- Navegação com o Google Street View*

### *3.6.2. Modelo da base de dados*

No âmbito deste projeto criou-se uma base de dados com o objectivo de ter a informação estruturada, organizada e acessível por vários utilizadores em simultâneo, para além de permitir um acesso rápido e simples à informação que se quer disponibilizar ao utilizador. A base de dados criada obedece a uma estrutura relacional e a Figura 9 descreve o modelo de dados definido para o efeito.

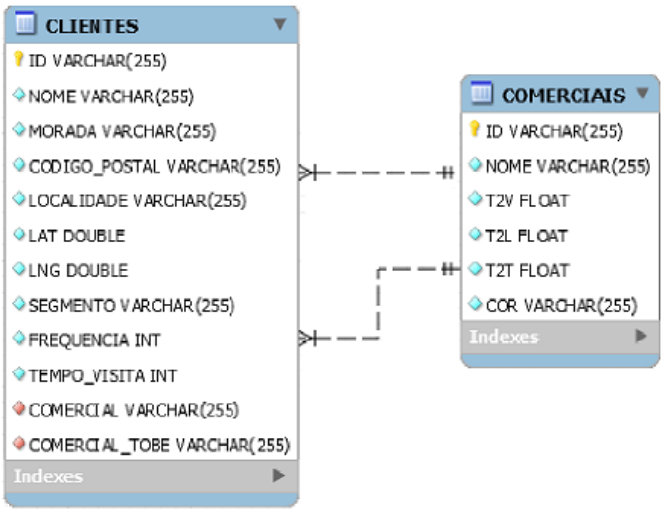



***Figura 9*** *- Modelo da Base de Dados relacional*

A base de dados é constituída por duas tabelas, CLIENTES e COMERCIAIS, sendo as relações existentes entre as tabelas as seguintes:

1. COMERCIAL - ID: permite saber a quantos clientes (no total e por segmento) está alocado cada agente comercial no cenário ASIS;
2. COMERCIAL_TOBE - ID**:** permite saber a quantos clientes (no total e por segmento) está alocado cada comercial no cenário TOBE.

As colunas das tabelas estão de acordo com os ficheiros de entrada, acrescentando-se ainda uma coluna à tabela de clientes, COMERCIAL_TOBE.

A tabela CLIENTES contém uma listagem de todos os clientes e tem as seguintes colunas:

– ID [VARCHAR]: identificador do cliente;
– NOME [VARCHAR]: nome do cliente;
– MORADA [VARCHAR]: morada do cliente;
– CODIGO_POSTAL [VARCHAR]: código postal do cliente;
– LOCALIDADE [VARCHAR]: localidade do cliente,
– LAT [DOUBLE]: Latitude do cliente, em WGS84;

– LNG [DOUBLE]: Longitude do cliente, em WGS84;
– SEGMENTO [DOUBLE]: segmento a que o cliente pertence;
– FREQUÊNCIA [INT]: número de visitas a efetuar ao cliente num ciclo comercial;
– TEMPO_VISITA [INT]: duração da visita ao cliente;
– COMERCIAL [VARCHAR]: agente comercial alocado ao cliente no cenário ASIS;
– COMERCIAL_TOBE [VARCHAR]: agente comercial alocado ao cliente no cenário TOBE.

A tabela COMERCIAIS contém uma listagem de todos os agentes comerciais e apresenta as seguintes colunas:

– ID [VARCHAR]: identificador do agente comercial;
– NOME [VARCHAR]: nome do agente comercial;
– T2V [FLOAT]: *time to visit* do agente comercial;
– T2L [FLOAT]: *time to lunch* do agente comercial;
– T2T [FLOAT]: *time to travel* do agente comercial;
– Cor [VARCHAR]: cor da representação no mapa do território do agente comercial.

### *3.6.3. Arquitetura do sistema*

Na implementação desta aplicação, seguiu-se uma arquitetura SOA e o *standard* JSON.

O diagrama de arquitetura encontra-se na Figura 10.

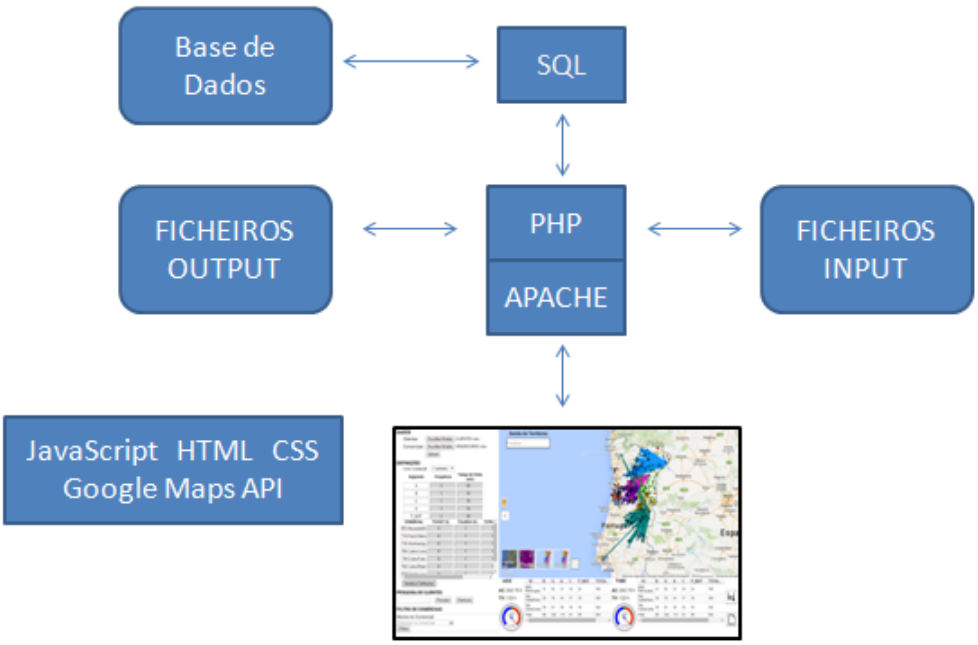



***Figura 10*** *- Diagrama da Arquitetura*

Componentes de arquitetura, *APIs* e tecnologias usadas:

- Sistema Operativo: *Windows* ou *Linux*;
- Servidor aplicacional: *Apache* 2;
- Servidor de Base de dados: *MySQL*;
- Linguagem de programação no servidor: *PHP* 5;
- Linguagem de programação no cliente (*browser*): HTML 5 (World Wide Web Consortium, 2014) + *Javascript*;
- *API* para acesso a conteúdos geográficos e geoprocessamento: *Google Maps API*;
- Outras *API* / Componentes utilizados: *jQuery*, *jQueryUI*, *PHPExcel* 2.0, *Google Visualization API*.

A escolha destas tecnologias resultou de serem as que a *Focus BC* utiliza neste tipo de projetos e pelo fato de esta aplicação poder vir a ser associada a outros e, possivelmente, alterada por outra pessoa.

## 3.7. Importação dos dados

Os dados dos clientes e dos agentes comerciais são fornecidos pelo cliente através do *upload* de ficheiros entrada CLIENTES.xls e COMERCIAIS.xls (ou .xlsx).

O ficheiro de clientes possui a estrutura definida na Tabela 4:

***Tabela 4*** *- Estrutura do ficheiro de clientes*

| **Campo** | **Tipo de Dados** | **Campo mandatório** | **Descrição do campo** |
|---|---|---|---|
| ID | Text | Not Null | Identificador do cliente |
| NOME | Text | Not Null | Nome do cliente |
| MORADA | Text | Null | Morada do cliente |
| CODIGO_POSTAL | Text | Null | Código postal do cliente |
| LOCALIDADE | Text | Null | Localidade do cliente |
| LAT | Number | Not Null | Latitude do cliente, em WGS84 |
| LNG | Number | Not Null | Longitude do cliente, em WGS84 |
| SEGMENTO | Text | Not Null | Segmento a que o cliente pertence |
| FREQUÊNCIA | Number | Not Null | Nº de visitas a efetuar ao cliente num ciclo comercial |
| TEMPO_VISITA | Number | Not Null | Duração da visita ao cliente |
| VENDEDOR | Text | Not Null | Identificador do agente comercial alocado ao cliente no cenário ASIS |

O ficheiro de agentes comerciais possui a estrutura definida na Tabela 5.

***Tabela 5** - Estrutura do ficheiros de agentes comerciais*

| **Campo** | **Tipo de Dados** | **Campo mandatório** | **Descrição do campo** |
|---|---|---|---|
| ID | Text | Not Null | Identificador do agente comercial |
| NOME | Text | Not Null | Nome do agente comercial |
| T2VISIT | Number | Not Null | *Time to visit* do agente comercial |
| T2LUNCH | Number | Null | *Time to lunch* do agente comercial |
| T2TRAVEL | Number | Null | *Time to travel* do agente comercial |
| COR | Text | Not Null | Cor da representação no mapa do território do agente comercial |

O carregamento da base de dados envolve um tratamento prévio dos dados, durante o qual:

- Os campos de texto, não obrigatórios, vazios são substituidos por "ND" (campo não definido);
- Os hífens são substituídos por *underscores*;
- As plicas são retiradas;
- Os espaços são suprimidos no caso de campos que correspondem a chaves primárias.

Para o *upload* dos ficheiros, o tratamento dos dados e a importação destes para a base de dados foram desenvolvidos ficheiros em PHP.

## 3.8. *Web services* implementados

Para aceder e enviar instruções *SQL* à base de dados, utilizou-se o *web service* database.php, disponibilizado pela *Focus BC*. Todos os *web services* implementados recorrem a este serviço.

Criaram-se serviços para a importação, o tratamento e a exportação dos dados. Todos eles utilizam a biblioteca PHPExcel (Microsoft, 2015), que serve para leitura e escrita de documentos em vários formatos, como .xls ou .xlxs. Para exportar o ficheiro das definições e os ficheiros dos clientes e dos agentes comerciais, implementaram-se, respetivamente, os serviços exportSettings.php e exportClients_Comerc.php. Para passar os ficheiros de entrada, carregados pelo utilizador, para uma pasta no servidor e atribuir-lhes nomes de acordo com a hora em que foram carregados, criou-se o serviço upload.php. Este serviço invoca o serviço load.php para fazer o tratamento dos dados, criar as tabelas de clientes e de agentes comerciais e carregá-las.

Os *web services* search.php e update.php servem, respetivamente, para encontrar na base de dados, clientes cujo nome ou parte do nome corresponda à *query* dada pelo utilizador ao usar a

caixa de pesquisa de clientes e atualizar o agente comercial associado ao cliente, após o utilizador realizar uma edição de territórios. Ambos devolvem objetos JSON, o primeiro com dados do cliente, nomeadamente a Latitude e a Longitude para ser possível adicionar um marcador ao mapa para cada cliente encontrado, o segundo devolve o resultado da operação na base de dados.

Relacionados com os agentes comerciais, implementaram-se dois *web services*, get.php e update.php. O primeiro devolve os dados da tabela de agentes comerciais. O segundo atualiza dados dos agentes comerciais, nomeadamente, quando o utilizador altera as definições associadas a um agente comercial.

Para ir buscar à base de dados e mostrar os dados estatísticos (os resultados) dos agentes comerciais, implementou-se o getStatistics.php.

## 3.9. Classes implementadas

As classes criadas foram: Cliente, Clientes, Vendedor, Vendedores, Territorio, Territorios, Metricas, Report e ainda a classe Main. A classe Main dá funcionalidade à aplicação ao recorrer a métodos das outras classes. É a partir desta que se define o fluxo da aplicação e as respostas às interações do utilizador. A criação do mapa e todos os eventos disponíveis na aplicação, associados ao mapa ou a qualquer botão, são definidos nesta classe.

Objetos da classe Cliente têm atributos de acordo com os dados dos clientes, como o nome, a localização, o agente comercial associado ou a janela de informação e métodos para mudar o agente associado, mostrar/retirar o cliente no mapa. A classe Clientes é um *array* de objetos do tipo Cliente e tem métodos para adicionar e remover clientes e mostrar/retirar todos os clientes.

A classe Vendedor só tem atributos e estes representam informações dos agentes comerciais, como o nome, o ACR, a cor, etc. A classe Vendedores representa um *array* de vendedores e tem métodos para adicionar, remover e apagar todos os vendedores do *array*.

A classe Territorio possui propriedades como o agente comercial associado, a cor, os clientes que o constituem ou uma janela de informação e métodos para geração e desenho do polígono representante de cada território e mostrar ou retirar o território do mapa. Objetos da classe Territorios possuem *arrays* com objetos da classe Territorio. O método de edição de territórios é definido nesta classe, que recorre a métodos da classe Territorio, como mostrar ou retirar do mapa.

A classe Metricas possui métodos para calcular e disponibilizar as métricas dos agentes comerciais.

Para os métodos relacionados com a geração de relatórios, criou-se a classe Report.

## 3.10. APIs da *Google* usadas no projeto

A *Google* tem disponíveis várias APIs que permitem a comunicação com os seus serviços, bem como a sua integração com outros serviços (Google, 2015 a). Os métodos de uso das APIs da *Google* seguem a mesma estrutura geral das de outros fornecedores de serviços *web*, em que a estrutura de *namespace* em *JavaScript* é a seguinte:

```
application.category.function(variables);
```

A *Google Maps JavaScript* API (Google, 2015 g) e a *Google Visualization* API (Google, 2015 k) foram os serviços *Google* usados para o desenvolvimento desta aplicação.

### *3.10.1. Google Maps JavaScript API*

A *Google Maps JavaScript* API foi desenvolvida para facilitar a introdução de serviços *Google Maps* em páginas HTML dinâmicas, sendo destinada à criação de aplicações na *web* que são executadas pelo *browser* do cliente e funciona com código HTML, CSS (*Cascading Style Sheets*) e *JavaScript* (ECMA International, 2011). A disponibilização e a personalização de mapas, *Geocoding*, *Directions e Street View* são exemplos das possibilidades de uso desta API.

Neste projeto utilizou-se a supracitada API para a disponibilização de informação de base geográfica, incluindo o mapa e o mapeamento dos clientes e outras funcionalidades, a geração de polígonos e o cálculo de áreas. *Leaflet* (*OpenStreetMap contributors*, 2015) ou *Bing Maps* (*Microsoft Corporation*, 2014) são exemplos de alternativas a este serviço.

Os dados geoespaciais fornecidos pelos serviços da *Google* apresentam-se em coordenadas geográficas, no sistema WGS 84. Para passar as coordenadas geográficas de uma localização no mundo para uma localização no mapa, a API do Google Maps transforma as coordenadas expressas em valores de Latitude e Longitude em coordenadas de tipo *world coordinates*, que referenciam um ponto no mapa, de forma unívoca. Para este propósito, é usada a projeção de Mercator, uma projeção cilíndrica.

As coordenadas de tipo *world coordinates* são independentes do nível de *zoom* e são medidas a partir da projeção de Mercator (o canto Noroeste do mapa, um valor de Longitude de 180 graus e de Latitude de aproximadamente 85 graus) e aumentam para Este e para Sul. Como a *tile* ou seção básica usada é de 256 × 256 *pixels*, o espaço de coordenadas do tipo *world coordinates* usável é {0-256}, {0-256} (Google, 2015 i).

As *tiles* do mapa são imagens que são carregadas em segundo plano com chamadas AJAX e, em seguida, inseridas em partes identificadas da página HTML. Quando um utilizador navega no mapa, a API do Google Maps envia informações sobre as novas coordenadas e níveis de *zoom* do mapa em chamadas AJAX, que retornam novas imagens (Svennerberg, 2010).

Quando se tem em conta o nível de *zoom* do mapa, trata-se já de coordenadas de tipo *pixel coordinates*, as quais são calculadas tendo em conta as coordenadas de tipo *world coordinates* e o nível de *zoom*.

Existem, ainda, as coordenadas de tipo *tile coordinates*. Cada *tile* é referenciada por um par "(x,y)", com origem no canto Noroeste do mapa e com valores de x e de y a crescerem da forma como se mostra na Figura 11.

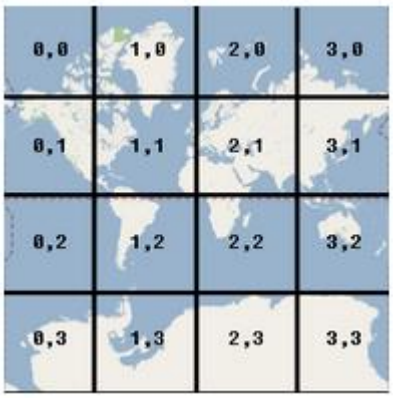



***Figura 11*** *- Esquema das tiles coordinates (imagem tirada de Google Maps JavaScript API - Map Types)*

### *3.10.2. Google Visualization API e Google Chart API*

A *Chart API* fornece uma forma simples de criar *charts* de vários tipos, como gráficos e tabelas, enviando um URL que inclui os dados e as opções de configuração do *chart* para um servidor da Google. Um *chart* é tudo o que seja passível de ser processado por um *browser*, como por exemplo, imagens.

A *Visualization API* constitui uma forma de ligar *charts* e fontes de dados através da *web* e de os publicar. Uma boa parte dos *charts* da *Chart API* é acessível através da anterior.

A *Google Visualization API* permite a comunicação entre uma fonte de dados estruturados e uma página HTML onde se mostram os dados de forma dinâmica. Esta API oferece uma API de *JavaScript* para aceder a *charts* e, independentemente da plataforma, segue a mesma arquitetura que outros produtos da Google API.

Neste projeto, recorreu-se ao serviço *Google Visualization API* para a disponibilização de informação alfanumérica, sob a forma de tabelas e gráficos e para aplicar filtros à informação disponibilizada, que constituem exemplos de possibilidades oferecidas por esta API.

## 3.11. Mapeamento dos clientes

A disponibilização do mapa é feita através da *Google Maps JavaScript API*, à qual se fornece o identificador da *div* em que vai ser inserido na página HTML e, para além de outras propriedades de estilo, como o tipo de mapa, que podem ser personalizadas, é obrigatório fornecer o nível de *zoom* e as coordenadas do centro do mapa. Apesar do *zoom* e das coordenadas definidas inicialmente para o mapa, como se recorreu ao método *fitBounds*(), os valores destes parâmetros vão ser ajustados aos dados, isto é, às localizações dos clientes, à medida que se vai chamando este método para cada novo cliente criado.

Sendo aux "document.getElementById('Div_Map')", a criação de um objeto da classe *Map* é feita da seguinte forma:

```
map = new google.maps.Map(aux, mapOptions);
```

Com este pedido é criado um objeto da classe *Map*, pelo que todos os métodos, eventos e propriedades desta classe são herdados. Cada cliente é representado no mapa por um marcador com a propriedade "cor", de acordo com o agente comercial.Aquando da criação de cada marcador, é também criada uma janela de informação com o nome e os agentes comerciais ASIS e TOBE do cliente representado por este. Cada marcador tem associado um evento do tipo "*click*" que, ao ocorrer, origina a visualização da janela de informação que lhe foi associada. Esta janela de informação, para além de dados sobre o cliente, tem ainda outra funcionalidade que será explicitada no capítulo 3.13. "Edição de territórios".

O pedido à *Google Maps JavaScript API* para a criar um objeto da classe *Marker* tem a seguinte estrutura:

```
marker = new google.maps.Marker(markerOptions);
```

Fornece-se como argumento a localização do marcador e o mapa a que está associado, entre outras propriedades opcionais como a cor, o ícone, os eventos que lhe são aplicados, o título, etc. A localização é o único parâmetro obrigatório. Se não se indicar qual é o mapa, é criado um marcador sem mapa associado.

Ao mapear os clientes, constatou-se o fato de existirem vários clientes com a mesma localização, o que se revelou um problema a resolver, dado que se dois ou mais clientes tiverem a mesma localização, para além de não ser possível a visualização de todos, torna-se também impossível a alteração do agente comercial associado a todos estes, porque fica disponível apenas uma janela de informação por localização. Encontraram-se várias soluções para este problema:

Solução 1: se, quando é criado um novo marcador, já existir um marcador nessa mesma localização, é atribuído aleatoriamente um pequeno desvio às coordenadas do novo marcador para que todos fiquem visíveis e clicáveis.

Em cada vez que se faz o carregamento dos clientes, a posição destes muda e perde-se demasiado tempo a encontrá-los, pelo que esta não é uma opção adequada.

Solução 2: uma *infowindow* com a informação de todos os clientes que estão nessa localização. Não é uma solução viável porque, para além de poder ficar demasiada informação numa *infowindow*, não permite a edição de territórios, dado que, para que tal aconteça, é usado o identificador do marcador clicado.

Solução 3: recorrendo ao *plug-in Overlapping Marker Spiderfier*.

Ao clicar no marcador visível, os vários marcadores que estão invisíveis na mesma posição, passam a ficar visíveis e clicáveis, através da sua expansão numa ou em várias circunferências em redor da localização e com segmentos que os unem à mesma.

Na Figura 12, apresenta-se um exemplo em que há quatro clientes com a mesma localização.

***Figura 12** - Solução para múltiplos clientes com a mesma localização - Overlapping Marker Spiderfier*

Cada marcador visível na expansão é passível de ser clicado, sendo assim possível visualizar a *infowindow* correspondente, resolvendo o problema apresentado. Outra vantagem desta solução é a de que a expansão dos marcadores não é aleatória, pelo que esta foi a solução escolhida.

Por defeito, o *plug-in* aplica esta solução a marcadores que se encontrem a uma distância *(pixel radius*) igual ou inferior a 20m do marcador clicado.

A utilização deste *plug-in* constitui um melhoramento do sistema.

## 3.12. Desenho dos territórios dos agentes comerciais

### *3.12.1. Algoritmo de desenho de polígonos*

A ordem pela qual o polígono é desenhado segue a ordem dos pontos no vetor fornecido, o que implica que um mesmo conjunto de pontos, ordenado de formas diferentes, possa originar polígonos diferentes, o que afeta a visualização dos dados. Na Figura 13, ilustra-se esta situação.

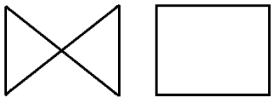


***Figura 13** - Exemplo de desenho de diferentes polígonos com o mesmo conjunto de pontos*

É, portanto, necessário escolher a forma de ordenação dos pontos para o desenho dos polígonos, o que varia de acordo com os objetivos da visualização dos dados.

O algoritmo implementado para a realizar a ordenação dos pontos teve por base um algoritmo apresentado no *Stack Overflow* (*Stack Exchange Inc*, 2015 b), um fórum de programação.

Os passos do algoritmo são os seguintes:

1. Seleção do ponto mais a Norte, ou seja, o que tiver o maior valor de Latitude de entre o conjunto de pontos. Em caso de empate, escolhe-se o que tiver o menor valor de Longitude, isto é, o que se encontra mais a Oeste. Para simplificar, passa-se a chamar este ponto de *UpperLeft*.
2. Ordenação dos pontos em relação ao *UpperLeft*. Sendo m1 o declive da reta que passa por um ponto P1 e pelo *UpperLeft*, e m2 o declive da reta que passa por um ponto P2 e pelo *UpperLeft:*

– Caso os declives sejam iguais, o ponto mais próximo do *UpperLeft* é desenhado primeiro, no sentido horário;
– Se m1 for negativo ou nulo e m2 positivo, desenha-se primeiro P1;
– Caso ambos os declives sejam positivos ou negativos, se m1 for maior do que m2, primeiro é desenhado o P1. (Stack Exchange Inc, 2015 a)

Apesar de se ter tido em conta um algoritmo de polígonos convexos (*convex hull algorithms*), verificavam-se mais sobreposições de territórios, o que tornava a visualização e a perceção da distribuição dos territórios mais difícil. De acordo com o algoritmo escolhido, todos os pontos fazem parte da fronteira do polígono, contribuindo para a sua forma. No caso de um *convex hull* isto não acontece, ficando pontos no interior do polígono, não pertencentes à fronteira.

### *3.12.2. Disponibilização dos territórios*

Os polígonos que representam os territórios dos agentes comerciais são desenhados no mapa recorrendo à classe *Polygon* da *Google Maps JavaScript API*.

Um objeto *Polygon* representa uma área delimitada por um caminho (*path*) fechado, que é definido por uma série de coordenadas, expressas em Latitude e Longitude, numa sequência ordenada. Podem ser definidas algumas propriedades como a cor do preenchimento, a opacidade ou a espessura e a cor da linha que o delimita.

É possível definir polígonos com formas complexas, fornecendo múltiplos *paths* (Google, 2015 j), o que não foi aplicado neste projeto.

O pedido à *Google Maps JavaScript API* para criar um objeto da classe *Polygon* tem a seguinte estrutura: *new google.maps.Polygon (polygonOptions);*

Fornece-se um vetor com o conjunto de pontos que o constituem, o seu *path*, e propriedades de estilo.

Depois de criar um objeto *Polygon*, é necessário associá-lo ao mapa e para tal, recorre-se ao método *setMap(map)*, disponível para objetos desta classe, o qual recebe como argumento o identificador do mapa.

Para calcular a área dos polígonos, invoca-se o método *computeArea(path)* da *Geometry Library* da *Google Maps API* (Google, 2015 h). No pedido à *API*, é passado o *path* do polígono. O resultado é devolvido em $m^2$.

## 3.13. Edição de territórios

A edição de territórios é realizada cliente a cliente. Um território de um agente comercial é alterado quando um dos clientes que o constituem é mudado para outro território ou quando um cliente pertencente ao território de outro agente comercial passa a ser alocado a este.

É ainda possível definir o agente responsável por um cliente como "Não atribuído", o que pode ser usado para remover clientes ou testar o impato que teria no novo cenário adicionar-se um novo agente comercial, uma vez que se cria assim o território "Não atribuído" cujas métricas são igualmente disponibilizadas.

Tal como detalhado e exemplificado anteriormente, a edição de territórios é realizada através das janelas de informação dos marcadores que representam os clientes.

Cada edição é realizada através de um pedido AJAX de tipo "POST", através do *web service* update.php. Este *web service* usa métodos de outro *web service*, o database.php, para a ligação à base de dados e atualização de dados, fornecendo como parâmetro uma *query SQL* (*Structured Query Language*) com uma instrução de *update* construída com o identificador do cliente, que é

conhecido por se guardar o identificador do marcador clicado, e o identificador do novo agente comercial que também é guardado ao ser selecionado.

Para além de a base de dados ser atualizada, são também gerados novos territórios, redesenhados os polígonos e recalculadas e atualizadas as métricas.

## 3.14. Cálculo de disponibilização das métricas

### *3.14.1. Cálculo das métricas*

As métricas definidas para os agentes comerciais são: o número total de clientes, o número total de clientes por segmento, a Capacidade Alocada (*Allocated Capacity*, AC), o Tempo para Visitas (*Time to Visit*, TV) e o Rácio da Capacidade Alocada (*Allocated Capacity Ratio*, ACR). Cada uma destas métricas é calculada e apresentada por agente comercial e para todos os agentes comerciais.

Através do *web service* getStatistics.php, fizeram-se *queries* à base de dados, com uma chamada AJAX, que devolve um *array* com objetos JSON, tantos quantos os agentes comerciais, com o número de clientes que cada agente tem de cada segmento. Iterou-se este *array* para se obter os totais pretendidos.

A métrica AC corresponde total de horas necessárias para visitar todos os clientes, de acordo com o número de visitas e o tempo de cada visita definidos para cada segmento de clientes, para um ciclo comercial.

A métrica TV representa o total de horas que os agentes comerciais têm disponível para visitas a clientes, por ciclo comercial.

A métrica ACR é um rácio e, como tal, não tem unidades. Representa a razão entre a carga alocada aos agentes e a disponibilidade destes para visitar clientes e deve ter um valor próximo da unidade.

Para calcular a AC, o TV e o ACR, usaram-se fórmulas, aplicadas em *JavaScript*.

Primeiro, calcula-se a AC para cada agente comercial, $j$:

$$AC_j = \sum (NC_1 * F_1 * VT_1 + \ldots + NC_i * F_i * VT_i) \quad (1)$$

em que $NC_i$ é o número de clientes do tipo $i$ alocados ao agente comercial, $F_i$ é o número de visitas a fazer a clientes do segmento $i$, por ciclo comercial, e $VT_i$ é o tempo definido, em horas, para a duração das visitas a clientes do segmento $i$, por ciclo comercial.

Depois, calcula-se o TV de cada agente comercial:

$$TV_j = T_j * CC * 5 \quad (2)$$

em que $T_j$ é o tempo destinado a visitas, em horas, por dia, de cada agente comercial $j$, e $CC$ é a duração, em semanas, do ciclo comercial.

Tendo a AC e o TV de cada agente comercial, calcula-se o ACR para cada um deles:

$$ACR\_individual = AC / TV \quad (3)$$

Calcula-se, também, o ACR para todos os agentes comerciais, inicialmente, e para todos os agentes comerciais selecionados, após aplicação de filtros pelo utilizador.

$$ACR = \sum AC / \sum TV \quad (4)$$

### *3.14.2. Disponibilização das métricas*

A disponibilização das métricas dos agentes comerciais é feita através de tabelas e gráficos e a visualização é interativa, isto é, o utilizador pode selecionar o/os agentes comerciais dos quais quer ver a informação disponível. Esta seleção ou filtragem afeta também os territórios que são mostrados no mapa.

A *Google Visualization API* em conjunto com a *Google Chart API* (Google, 2015 b) fornece classes e métodos que permitem este tipo de disponibilização de dados de forma interativa.

Todos os tipos de *charts*, como tabelas ou manómetros, recebem objetos da classe *DataTable*.

Para a criação das tabelas, usa-se um objeto da classe *DataTable* que representa uma tabela de valores bidimensionais e mutáveis.

Adicionam-se colunas através de chamadas ao método *addColumn()*, que recebe como argumentos, o tipo de dados, campo obrigatório, e ainda o nome da coluna e um identificador da coluna. É devolvido o índice da nova coluna.

Para preencher a tabela, usa-se o método *addRow()* que recebe um vetor com dados. Cada vetor corresponde a uma linha da tabela, pelo que os seus elementos seguem a ordem das colunas. Se não se enviar nenhum argumento ao invocar este método, é adicionada uma linha vazia à tabela. Este método devolve um número que corresponde ao índice da linha que foi adicionada.

Para a visualização das tabelas, criaram-se objetos da classe *Table* ao qual se aplica o método *draw()* que recebe como argumentos as opções de configuração da tabela e uma instância da classe DataView. Objetos desta classe são vistas dinâmicas de uma *DataTable*, de tipo *read-only* e possibilitam a seleção de algumas colunas ou linhas, e outras funcionalidades como a reordenação destas.

A criação dos gráficos, que são manómetros neste caso, resultou do uso da classe *Gauge*, que recebe a *div* correspondente. A visualização dos gráficos é possível através do método *draw()*, o qual recebe os dados que se quer mostrar e as opções de configuração pretendidas.

Para filtrar a informação disponibilizada por agente comercial, criou-se um objeto da classe *CategoryFilter*, que é um selecionador para escolher um ou mais valores de entre um conjunto de valores definidos.

## 3.15. Desenvolvimento da *interface*

É através da *interface* que são realizadas todas as interações entre o utilizador e a aplicação, sendo o acesso feito através de um *web browser*.

O maior desafio do desenho da *interface* passou pela necessidade de mostrar muita informação ao utilizador apenas no espaço de um ecrã, informação essa em vários formatos: tabelas, gráficos e mapa.

Para além de toda a informação que o utilizador deve ter ao seu dispor ao interagir com a aplicação, era ainda necessário acrescentar ferramentas de pesquisa e filtragem e também botões de exportação de resultados e de submissão de ficheiros.

Dividiu-se o ecrã em várias seções:

1. Painel lateral, à esquerda, constituído pelas áreas de submissão de dados, de definições, de pesquisa de clientes e de filtragem de agentes comerciais;
2. Mapa, em cima à direita, que ocupa a maior parte do espaço disponível, por ser a área de visualização dos clientes e dos territórios e de edição dos últimos;
3. Resultados/ métricas, por baixo do mapa, que engloba a apresentação dos resultados de ambos os cenários em gráficos e tabelas e os botões de exportação de ficheiros e impressão de relatório.

Uma solução encontrada para minimizar o problema do espaço foi a de colapsar/expandir elementos, pelo que, quando o utilizador não precisa do painel lateral, pode colapsá-lo, expandindo o mapa, onde está, por exemplo, a editar territórios, e as métricas que quer ter ao seu dispor e que vão sendo atualizadas a par das alterações que está a realizar no mapa. Outra solução encontrada foi o uso de deslizadores verticais e horizontais.

Retiraram-se, ainda, os botões de *panning* e de *zoom* do mapa, que são disponibilizados por defeito pela *Google Maps API*, dado que o utilizador pode realizar estas opções com o rato ou no ecrã, no caso de um dispositivo móvel.

Quanto às tecnologias usadas para o desenvolvimento da *interface*, a estrutura foi construída com HTML, definiram-se os estilos com CSS e as funcionalidades foram implementadas através de *JavaScript*.

Na Figura 14, apresenta-se um exemplo da *interface*.

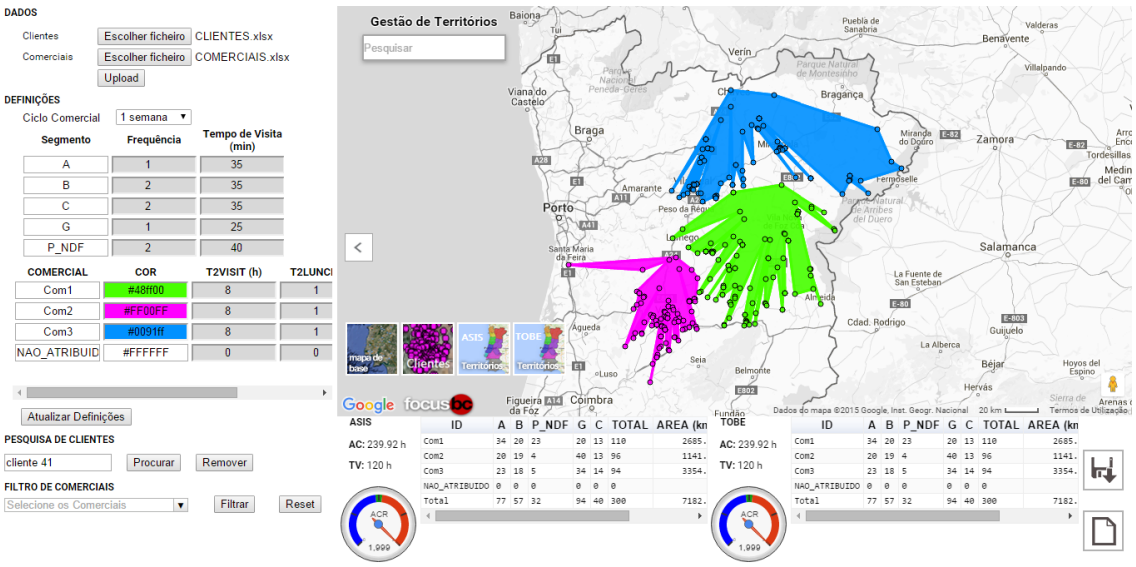



***Figura 14*** *- Exemplo da interface*

## 4. Exemplo de aplicação

Para exemplificar um caso de estudo desta aplicação, usou-se uma amostra de 300 clientes, atribuídos a três agentes comerciais, para um ciclo comercial de duas semanas.

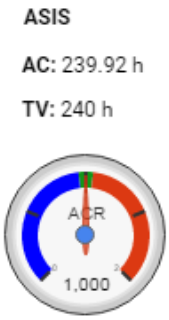
Todos os agentes comerciais têm atribuídas 8 horas para visitas a clientes.

O tempo de visita e a frequência das visitas para o ciclo comercial definido, de acordo com a segmentação dos clientes, consta da Tabela 6.

***Tabela 6** - Definições do cenário ASIS*

| SEGMENTO | FRENQUÊNCIA | TEMPO DE VISITA (min) |
|---|---|---|
| A | 1 | 35 |
| B | 2 | 35 |
| C | 2 | 35 |
| G | 1 | 25 |
| P_NDF | 2 | 40 |

No cenário ASIS, verifica-se um ACR total de uma unidade, o que é ideal. No entanto, há um desequilíbrio nos valores de ACR de cada agente comercial, individualmente, como se pode verificar na Figura 15. O agente comercial "Com1" apresenta um valor de ACR de 1.217, o que indica que este se encontra sobrealocado, enquanto que os agentes comerciais "Com2" e "Com3" se encontram subalocados, apresentando, respetivamente, valores de ACR de 0.888 e 0.895.




| ID | A | B | P_NDF | G | C | TOTAL | AREA (km2) | ACR | COR |
|---|---|---|---|---|---|---|---|---|---|
| Com1 | 34 | 20 | 23 | 20 | 13 | 110 | 2685.985 | 1.217 | |
| Com2 | 20 | 19 | 4 | 40 | 13 | 96 | 1141.981 | 0.888 | |
| Com3 | 23 | 18 | 5 | 34 | 14 | 94 | 3354.682 | 0.895 | |
| Total | 77 | 57 | 32 | 94 | 40 | 300 | 7182.648 | | |

***Figura 15** - Resultados do cenário ASIS do caso de estudo*

Na Figura 16, apresentam-se os territórios do cenário inicial.

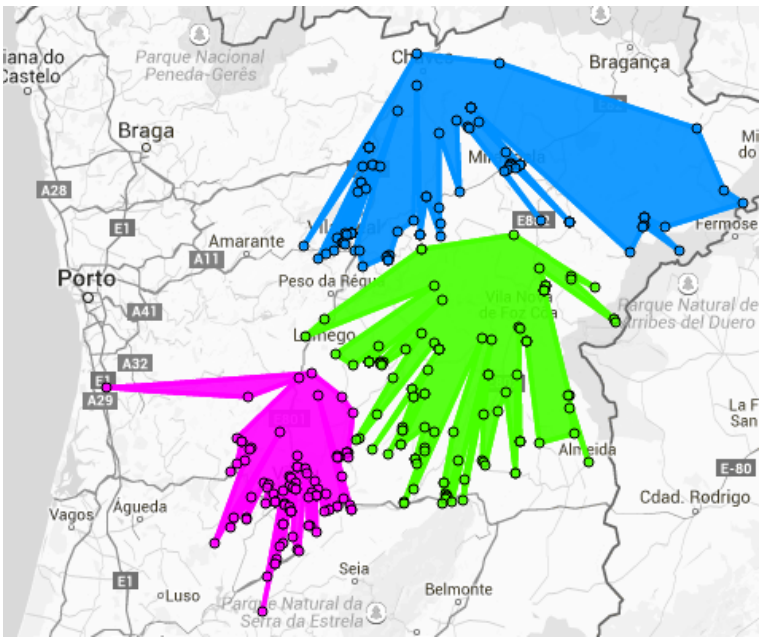



***Figura 16*** *- Territórios do cenário ASIS do caso de estudo*

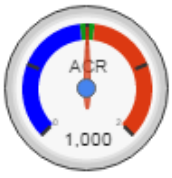
O agente identificado por "Com1" tem uma carga alocada superior à sua disponibilidade para visitas a clientes, enquanto os outros dois agentes comerciais se passa o oposto.

Depois de se editar o cenário ASIS, através da passagem de clientes do agente sobrealocado para os outros agentes, obtiveram-se os resultados apresentados na Figura 17 para o novo cenário criado.

**TOBE**

**AC:** 239.92 h

**TV:** 240 h




| ID | A | B | P_NDF | G | C | TOTAL | AREA (km2) | ACR | COR |
|---|---|---|---|---|---|---|---|---|---|
| Com1 | 29 | 18 | 18 | 16 | 10 | 91 | 1951.904 | 1.003 | |
| Com2 | 22 | 19 | 8 | 41 | 15 | 105 | 1470.534 | 1.003 | |
| Com3 | 26 | 20 | 6 | 37 | 15 | 104 | 3650.483 | 0.993 | |
| Total | 77 | 57 | 32 | 94 | 40 | 300 | 7072.921 | | |

***Figura 17*** *- Resultados do cenário TOBE do caso de estudo*

Durante a edição, alteraram-se os agentes comerciais de dezanove clientes.

No novo cenário, observa-se uma maior homogeneidade nos valores de ACR dos agentes, estando todos próximos da unidade. Os valores de ACR no novo cenário variam entre 0.993 e 1.003.

Comparando os resultados dos cenários ASIS e TOBE, verifica-se que o segundo é o mais favorável, consistindo numa otimização do primeiro.

Após a criação do novo cenário, obtiveram-se os territórios apresentados na Figura 18.

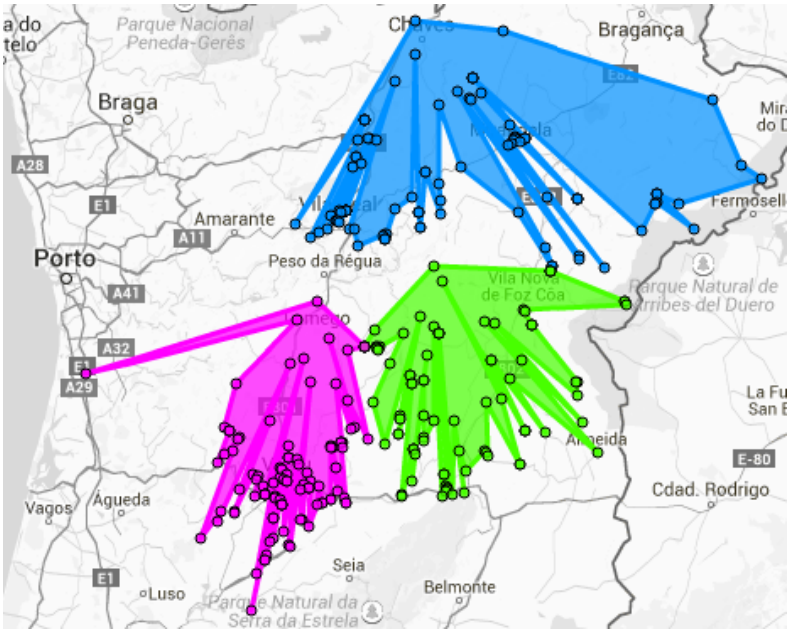



***Figura 18*** *- Territórios do cenário ASIS do caso de estudo*

## 5. Considerações finais

A aplicação desenvolvida possibilita uma forma de gestão de territórios de agentes comerciais, como inicialmente proposto. O utilizador, um gestor de uma empresa, pode assim usar esta aplicação para obter uma boa consciência da distribuição geográfica dos clientes que os agentes comerciais têm de visitar e dos resultados obtidos e alterar territórios diretamente no mapa, em vez de o fazer a partir de folhas de cálculo, tendo sempre ao dispor métricas para apoio à decisão.

A *Google Maps API* revelou-se bem documentada e com uma variedade de métodos capaz de responder às necessidades de base geográfica deste projeto, pelo que não foi necessário recorrer a outras API deste género.

Quanto a melhorias futuras, uma possibilidade é a de migração para uma base de dados espacial, o que poderia facilitar a inclusão de outras funcionalidades de SIG, nomeadamente, que exijam informação topológica e de geometria.

Sugere-se a inclusão de alguns serviços que poderiam trazer mais valor à aplicação e facilitar a tarefa dos gestores, entre eles, um serviço de transformação de coordenadas e a possibilidade de *geocoding*, porque nem sempre os gestores têm ao dispor as localizações dos clientes em Latitude e Longitude, e quando têm, nem sempre é no sistema de coordenadas pretendido.

Caso o cenário atual não tenha já partido da aplicação de um algoritmo de otimização de rotas, recomenda-se a inclusão deste serviço, tendo em conta os tempos das deslocações, os sentidos da rede viária, o trânsito, entre outros.

Outra sugestão de uma melhoria a fazer à aplicação é a de definições *default*, definidas por cada utilizador, guardadas e associadas à sua conta, pelo que, após *login* na plataforma onde este módulo será englobado, estas são disponibilizadas.

Sugere-se, ainda, a inclusão de vários algoritmos de desenho de polígonos e a possibilidade de escolha por parte do utilizador, de acordo com o que prefere para a visualização dos territórios dos agentes comerciais.

## Referências


Brinzarea, B. and Hendrix, A. (2009). *AJAX and PHP: Building Modern Web Applications*. 2ª Edição. Packt Publishing.

Couto, Francisco M. (2012) *Desenvolvimento de Aplicações Web baseadas em SOA e AJAX*. [Online] Disponível em: http://webpages.fc.ul.pt/~fjcouto/files/manual_soa_ajax_20120221.pdf. [Acesso em Abril de 2015].

ECMA International (2011). *ECMAScript Language Specification.* [Online] Disponível em: http://www.ecma-international.org/ecma-262/5.1/Ecma-262.pdf. [Acesso em Setembro de 2015].

ECMA International (2013). *The JSON Data Interchange Format*. [Online] Disponível em: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf. [Acesso em Setembro de 2015].

Focus BC (2015) [Online] Disponível em: http://www.focus-bc.com/pt/. [Acesso em Junho de 2015].

Google (2015 a). *Acerca da Google*. [Online] Disponível em: https://www.google.pt/about/. [Acesso em Setembro de 2015].

Google (2015 b). *Charts.* [Online] Disponível em: https://developers.google.com/chart/. [Acesso em Setembro de 2015].

Google (2015 c). *Google Cloud Platform*. [Online] Disponível em: https://cloud.google.com/. [Acesso em Setembro de 2015].

Google (2015 d). *Google Maps.* [Online] Disponível em: https://www.google.com/maps/about/. [Acesso em Setembro de 2015].

Google (2015 e). *Google Maps APIs*. [Online] Disponível em: https://developers.google.com/maps/. [Acesso em Setembro de 2015].

Google (2015 f). *Google Maps for Work*. [Online] Disponível em: https://www.google.com/work/mapsearth/. [Acesso em Setembro de 2015].

Google (2015 g). *Google Maps Javascript API.* [Online] Disponível em: https://developers.google.com/maps/documentation/javascript/. [Acesso em Setembro de 2015].

Google (2015 h). *Google Maps Javascript API Geometry Library*. [Online] Disponível em: https://developers.google.com/maps/documentation/javascript/geometry. [Acesso em Setembro de 2015].

Google (2015 i). *Google Maps Javascript API Map Types.* [Online] Disponível em: https://developers.google.com/maps/documentation/javascript/maptypes. [Acesso em Setembro de 2015].

Google (2015 j). *Google Maps Javascript API Shapes.* [Online] Disponível em: https://developers.google.com/maps/documentation/javascript/shapes. [Acesso em Setembro de 2015].

Google (2015 k). *Google Visualization API Reference.* [Online] Disponível em: https://developers.google.com/chart/interactive/docs/reference. [Acesso em Setembro de 2015].

Khan, Z. and Adnan, M. (2010) *A Comparative Study of Google Maps and MapQuest - Usability Evaluation of Web-based GIS Applications.* [Online] Disponível em: http://www.diva-portal.org/smash/get/diva2:832106/FULLTEXT01.pdf. [Acesso em Abril de 2015].

Li, S., Dragicevic, S., and Bert, V. (2011). *Advances in Web-based GIS, Mapping Services and Applications.* CRC Press.

MacKerron, G. (2013). *Overlapping Marker Spiderfier for Google Maps API v3*. [Online] Disponível em: https://github.com/jawj/OverlappingMarkerSpiderfier. [Acesso em Abril de 2015].

Microsoft (2015) *PHPExcel*. [Online] Disponível em: https://phpexcel.codeplex.com/. [Acesso em Setembro de 2015].

Microsoft Corporation (2014). *Bing Maps API*. [Online] Disponível em: http://www.microsoft.com/maps/choose-your-bing-maps-API.aspx. [Acesso em Setembro de 2015]

OpenStreetMap contributors (2015). *Leaflet* [Online] Disponível em: http://leafletjs.com/. [Acesso em Setembro de 2015]

Oracle Corporation (2015). *MySQL* [Online] Disponível em: https://www.mysql.com/. [Acesso em Setembro de 2015].

Stack Exchange Inc. (2015 a). *Sort latitude and longitude coordinates into clockwise ordered quadrilateral* [Online] Disponível em: http://stackoverflow.com/questions/2855189/sort-latitude-and-longitude-coordinates-into-clockwise-ordered-quadrilateral. [Acesso em Setembro de 2015].

Stack Exchange Inc. (2015 b). *Stack overflow.* [Online] Disponível em: http://stackoverflow.com/. [Acesso em Setembro de 2015].

Svennerberg, G. (2010). *Beginning Google Maps API 3*. 2ª Edição. Apress.

The Apache Software Foundation (2015) [Online] Disponível em: http://www.apache.org/. [Acesso em Abril de 2015].

The jQuery Foundation (2015) *jQuery API.* [Online] Disponível em: https://api.jquery.com/. [Acesso em Setembro de 2015].

The jQuery Foundation (2015) *jQuery.ajax().* [Online] Disponível em: http://api.jquery.com/jquery.ajax/. [Acesso em Junho de 2015].

The jQuery Foundation (2015) *Category: CSS.* [Online] Disponível em: http://api.jquery.com/category/css/. [Acesso em Junho de 2015].

The PHP Group (2015) *PHP.* [Online] Disponível em: https://secure.php.net/. [Acesso em Junho de 2015].

World Wide Web Consortium (2015) *Cascading Style Sheets.* [Online] Disponível em: http://www.w3.org/Style/CSS/. [Acesso em Junho de 2015].

World Wide Web Consortium. (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition).* Disponível em: http://www.w3.org/TR/xml/. [Acesso em Outubro de 2015].

World Wide Web Consortium (2014). *Recommendation 28 October 2014* [Online] Disponível em: http://www.w3.org/TR/2014/REC-html5-20141028/. [Acesso em Junho de 2015].

Zheng, K., Soomro, T., Pan Y. (2000) *Web GIS: Implementation issues*. [Online] Disponível em: https://www.researchgate.net/.[Acesso em Setembro de 2015].