Rui Pedro Silva Cruz

Impacto das tecnologias de Informação Geoespacial no Contexto Humanitário: Disponibilização de dados alfanuméricos e geoespaciais abertos através de uma aplicação web

Dissertação de Mestrado em Tecnologias de Informação Geográfica, área de especialização em Ciências e Tecnologias de Informação Geográfica, orientada pelo Professor Doutor José Paulo Elvas Duarte de Almeida e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

2022

· U C ·

Universidade de Coimbra

Rui Pedro Silva Cruz

Impacto das tecnologias de Informação Geoespacial no Contexto Humanitário: Disponibilização de dados alfanuméricos e geoespaciais abertos através de uma aplicação web

Dissertação de Mestrado em Tecnologias de Informação Geográfica, área de especialização em Ciências e Tecnologias de Informação Geográfica, orientada pelo Professor Doutor José Paulo Elvas Duarte de Almeida e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

2022

· U C ·

UNIVERSIDADE DE COIMBRA

Agradecimentos

Quero agradecer ao meu orientador, o Professor José Paulo, por toda a sua ajuda, disponibilidade e paciência comigo ao longo de todo este processo. Foi sem dúvida inspirador tudo o que aprendi consigo durante as nossas reuniões, e espero um dia conseguir implementa-lo na minha vida profissional e pessoal.

Quero também deixar um grande agradecimento aos meus pais por todo o apoio que me deram ao longo destes 10 anos de trabalho que agora findam.

Por fim, quero agradecer a todos os meus amigos, e em particular ao Olavo, pelo seu apoio e sensatez durante o confinamento.

Nº do aluno: 2018227388

Nome: Rui Pedro Silva Cruz

Título da dissertação:
Impacto das tecnologias de Informação Geoespacial no Contexto Humanitário: Disponibilização de dados alfanuméricos e geoespaciais abertos através de uma aplicação web

Palavras-Chave:

- webSIG
- Ação Humanitária
- Sistemas de Informação Geográfica
- Python
- GDAL

Resumo

O Gabinete para a Coordenação de Assuntos Humanitários das Nações Unidas estima que em 2022, 274 milhões de pessoas venham a necessitar de ajuda humanitária, um número consideravelmente superior aos 235 milhões registados no ano anterior. No entanto, as Nações Unidas e as respetivas organizações parceiras fixaram o seu objetivo anual nas 183 milhões mais carenciadas, o que por si só exigirá um montante a rondar os 41 mil milhões de dólares em donativos. Apesar do número exorbitante, esta quantia deixará por assistir cerca de 91 milhões de indivíduos que, tanto por conflitos armados ou desastres naturais, são arrastados para uma condição de vida onde nem as mais básicas necessidades inerentes à sua condição humana verão satisfeitas. Torna-se assim imperativo maximizar os efeitos que todos os atores humanitários possam vir a ter no terreno, de modo a extrair o máximo valor possível de cada donativo. Para tal, só uma tomada de decisão apoiada em dados muito concretos sobre as populações afetadas, bem como das infraestruturas disponíveis e do seu presente estado de conservação, poderá conduzir a um auxílio o mais abrangente possível no mais curto espaço de tempo. Dado ser através da visão que consumimos a maioria da informação acerca do mundo que nos rodeia, esta torna-se mais inteligível quando apresentada em imagens que representem fielmente a informação numérica da qual originaram, ou no caso de informação geoespacial, de mapas onde através de diferentes noções geométricas se representem esses elementos e as suas características de interesse para o leitor. Com o advento da internet e a possibilidade de partilha de todo o tipo de informação, a geoespacial viu assim uma forma de ser não só partilhada mas também usada para a produção de mapas interativos que sirvam um propósito bem definido. Assistiu-se assim no início do século XXI aos primeiros sistemas de informação geográfica em contexto web, ou webSIGs, e ao potencial que tinham para deles se extrair informação através de grandes volumes de dados disponibilizados de uma forma apelativa para o utilizador.

Foi neste paradigma que o presente trabalho assentou, tendo-se dado os primeiros passos na conceção de um webSIG capaz de informar a tomada de decisão em contexto humanitário.

Abstract

The United Nations Office for the Coordination of Humanitarian Affairs estimates that in 2022, 274 million people will be in need of humanitarian aid, a figure considerably greater than the 235 million registered the previous year. However, the United Nations and its respective partner organizations have set their yearly target for the 183 million people most in need, which alone will require around $41 billion in donations. Despite this outrageous number, this amount will leave around 91 million individuals unassisted who, either due to armed conflict or natural disasters, are dragged into a state where not even the most basic needs inherent to their human condition will be met. It is therefore imperative to maximize the effects that all humanitarian actors may have on the ground, in order to extract the maximum possible value from each donation. To this end, only decision-making supported by very concrete data on the affected populations, as well as the available infrastructure and its present state of conservation, can lead to the widest possible assistance in the shortest possible amount of time. Since it is through our eyes that we take in most of the information about the world around us, it thus becomes more intelligible when presented in images that faithfully represent the numerical information from which they've originated, or in the case of geospatial information, maps that by means of different geometric concepts can represent these elements and their characteristics of interest to the reader. With the advent of the internet and the possibility of sharing all kinds of information, geospatial information thus saw a way to be not only shared but also used for the production of interactive maps that serve a well-defined purpose. Thus, in the beginning of the 21st century, we witnessed the first web based geographic information system, or webGIS, and the potential they had to extract information from large volumes of data and make it available in an appealing way to the user.

It was off of this paradigm that the present work was based, having taken the first steps in the design of a webGIS capable of informing the decision-making process in a humanitarian context.

Índice

1. Introdução

1.1. Enquadramento

A ação humanitária tem sido primordial na mitigação dos efeitos de calamidades desde Florence Nightingale, a famosa enfermeira inglesa que juntamente com a sua equipa conseguiu reduzir drasticamente a mortalidade dos soldados feridos em combate durante a guerra da Crimeia, na década de cinquenta do século XIX (Fee & Garofalo, 2010). Outro marco importante foram as convenções de Genebra, assinadas entre 1864 e 1949, que viriam a estabelecer os princípios para um tratamento condigno de presos de guerra e de civis durante conflitos armados, apesar de não terem conseguido impedir as barbáries cometidas durante as duas guerras mundiais, em particular a segunda (Cameron et al., 2015). Desde então surgiram várias organizações no sentido de não só impedir que as atrocidades cometidas no passado se voltassem a repetir no futuro, como também de mitigar os efeitos nefastos provocados por décadas de era colonial. A primeira e mais importante seria a Organização das Nações Unidas, estabelecida logo após o fim da segunda grande guerra, na tentativa de travar possíveis retaliações por parte dos membros do Eixo. Desde então que a ação das Nações Unidas se tem focado não só na resolução de conflitos entre estados beligerantes como na manutenção da paz após esses conflitos, tendo sido a guerra entre Irão e Iraque em 1980, o genocídio da minoria Tutsi por parte do grupo Hutu no Rwanda em 1994, ou a ocupação forçada de Timor-Leste por parte da Indonésia até 1999, alguns exemplos dos conflitos que esta organização ajudou a mediar e o pós-guerra a monitorizar. Outra área de atuação das Nações Unidas é a ação humanitária. Seja devido a conflito armado ou desastre natural, várias organizações dentro e fora das Nações Unidas surgiram no sentido de dar resposta às necessidades mais imediatas de populações fragilizadas por estes acontecimentos. Alguns exemplos notórios seriam o Alto Comissariado da ONU para os Refugiados (1950), O Programa Alimentar Mundial (1961), Médicos Sem Fronteiras (1971) ou O Gabinete para a Coordenação de Assuntos Humanitários (1999). Todas estas organizações contribuem desde a sua conceção para o alívio do sofrimento de populações dispersas maioritariamente pelos continentes africano, asiático e sul americano, largamente financiadas por donativos de membros das Nações Unidas.

O modus operandi destas organizações assenta numa ótica de otimização dos recursos disponíveis. A título de exemplo, por cada donativo efetuado ao Programa Alimentar Mundial através da aplicação móvel Share The Meal, 62% reverterá de forma direta para ajudar as populações com bens alimentares, 28% será investido em marketing e campanhas de angariação de fundos, 6% para suportar custos da própria organização e 4% para comissões de pagamento associadas aos donativos. (ShareTheMeal) Como se pode ver, organizações de relevo como esta acabam por ter outros custos associados que as impedem de canalizar a totalidade dos seus fundos para o apoio às populações, daí a importância da sua cooperação com outras organizações locais que porventura tenham acesso a fundos de entidades nacionais e um melhor entendimento das faixas demográficas com maiores necessidades. Essas organizações terão também um bom conhecimento da rede viária disponível, tantas vezes deficitária e parcamente mapeada, mas de extrema importância para uma ajuda eficaz às populações.

Dado que tanto deste processo de ajuda humanitária se coaduna com um bom volume de informação, tanta dela geolocalizada, até que ponto se poderia considerar útil um sistema que não só a agregasse como também a disponibilizasse, possivelmente até de uma forma interativa, mas acima de tudo visualmente apelativa, ao ponto de fazer passar a qualquer ator humanitário uma resposta clara a qualquer pergunta sobre a missão que lhe compete? Voltando um pouco atrás nesta reflexão, é interessante considerar que para além de uma grande humanista, Nightingale era também uma forte adepta de Estatística e do poder da visualização de dados, tendo conseguido provar desta forma um nexo de causalidade entre as pobres condições sanitárias da Índia e a alta taxa de mortalidade dos soldados britânicos aquando da ocupação deste território pelo império vitoriano. Para ilustrar a suas conclusões, recorreu a gráficos circulares, divididos em 12 porções, onde a área de cada uma estaria relacionada com o número e a causa das mortes militares de cada mês. Graças aos seus esforços, Nightingale conseguiu o apoio do império para as reformas que queria implementar ao nível das instalações de saúde por toda a Índia, tendo conseguido baixar significativamente a mortalidade entre soldados britânicos num espaço de dez anos. (Hess, 1998)

Considerando a realidade humanitária não menos complexa dos dias atuais, particularmente no contexto africano, considera-se que um sistema de informação terá não só de revelar informação de variabilidade temporal mas também espacial, permitindo assim tirar o máximo de conclusões possíveis sobre o teatro de operações em que se opera. Nesse sentido, com este trabalho propõe-se um webSIG, ou um Sistema de Informação Geográfica que permita tudo o referido acima mas totalmente acedível via internet.

1.2. Utilidade do WebSIG no contexto humanitário

A título de exemplo considere-se a guerra civil que deflagrou em novembro de 2020 entre forças reacionárias e militares etíopes pelo controlo da região de Tigray, no norte da Etiópia, a qual já provocou a morte de pelo menos 10.000 civis. (Gesesew et al., 2021) De forma a evitar-se uma mortandade de maiores dimensões, esforços terão de ser feitos no sentido de uma evacuação desses civis para regiões vizinhas sobre o controlo das forças nacionais. Para se definirem prioridades nessa operação será necessário um conhecimento exaustivo:

- Da composição demográfica da região de Tigray no sentido de se perceber os números de:
    - Crianças, e quantas estarão subnutridas;
    - Mulheres, e quantas grávidas ou vítimas de abuso sexual;
    - Homens, e quantos com capacidade para trabalhar e assim continuar a apoiar as suas famílias.
    - Agregados familiares, e quantos deles professam a fé muçulmana, um detalhe importante na medida em que esta crença religiosa é favorável a relações poligâmicas de um homem para várias mulheres, dando aso a agregados familiares monoparentais, onde a mulher fica por vezes a cargo de vários filhos.
- Das infraestruturas afetadas pelo conflito, nomeadamente o grau de inoperacionalidade:
    - Das vias de comunicação terrestres que fazem a ligação entre Tigray e as regiões vizinhas;
    - Das instalações de saúde ou outras de grande dimensões que permitam albergar grandes quantidades de pessoas.
- Da localização de infraestruturas como depósitos de água ou armazéns usados por ONGs para armazenamento de mantimentos a distribuir pelas populações, e de que forma o seu acesso ficou comprometido devido ao conflito.

A resposta a todas estas perguntas passará por um sistema capaz de agregar toda esta informação, tanto a vetorial como a alfanumérica, numa base de dados que, a partir de um interface gráfico, seja capaz de disponibilizar toda esta informação de forma interativa, permitindo ao utilizador um cruzamento dos dados de forma a permitir a resposta a questões que estejam dependentes de duas ou mais variáveis.

2. Estado da arte

2.1. Breve contextualização da história dos WebSIG

De modo a entender-se ainda melhor a utilidade que este WebSIG possa vir a ter no contexto humanitário, é importante refletir sobre o caminho já percorrido neste domínio, o das tecnologias de informação geográfica, nomeadamente na forma como estas foram sendo disponibilizadas através da Internet e a gama de utilizadores que abrangem.

De facto, as capacidades de visualização e até de geoprocessamento de dados de um SIG disponibilizados num sítio da Internet evoluíram drasticamente em comparação com aquilo que eram nos seus primórdios, ou seja, pouco depois da própria Web ter sido lançada. Numa fase inicial, a disponibilização destes dados resumia-se a mapas estáticos, dando a ilusão de algum dinamismo, como no caso do PARC Map Viewer da Xerox, lançado apenas quatro anos após a Web 1.0. ter começado em 1989 (Choudhury 2014). Neste último caso, os cartogramas seriam apresentados no formato GIF (Graphics Interchange Format) após terem sido produzidos a partir de imagens Raster armazenadas nos servidores da Xerox. Estes mapas poderiam ser posteriormente partilhados via Hyperlink, sendo possível ao destinatário do mesmo a visualização da informação com os parâmetros de visualização definidos pelo remetente, tais como a escala do mapa ou a ordem das camadas de informação, nunca sendo possível alterar nenhum destes componentes (Neumann, 2017). Só mais tarde, em 1999, é que todo este processo viria a ser uniformizado através da implementação de um protocolo denominado WMS (Web Map Service) da autoria do OGC - Open Geospatial Consortium (Open Geospatial Consortium Inc., 2006). Esta era de facto não só a primeira era da informação geográfica na Web mas também a primeira era da Web propriamente dita, ou a Web 1.0, orientada para um acesso simples e pouco exaustivo da pouca informação relevante que havia disponível, um tempo em que apenas os mais curiosos se aventuravam a navega-la, em nada se assemelhando à quase dependência de que dela se sofre no momento presente, pelo menos nos países onde a penetração da Internet é mais acentuada.

Numa segunda fase, com o aumento na disponibilização de dados, sobretudo matriciais como no caso das imagens de satélite, e com o crescente interesse por este tipo de produtos face aos efeitos das alterações climáticas bem patentes em desastres como o furacão Katrina em 2005 ou o

tsunami que atingiu a Indonésia em 2004, surge a necessidade de um maior dinamismo na apresentação destes dados geográficos (Laituri & Kodrich 2008). Assim, começaram a emergir os primeiros WebSIGs, sendo o exemplo mais notório o Google Maps, lançado em 2005, e que já permitia não só a visualização de imagens de satélite como também algum geoprocessamento, como o cálculo de rotas entre pontos, uma das ferramentas mais utilizadas do Google Maps. Com o aumento da informação disponibilizada por esta plataforma e atendendo à dimensão da área mapeada, tornar-se-ia impossível para os servidores da Google responder de forma consistente aos pedidos de todos os utilizadores que estivessem a utilizar os serviços do Maps num dado momento utilizando apenas o protocolo WMS em que a responsabilidade de processamento de imagem cai exclusivamente do lado dos servidores da Google. Assim, em 2010 o OGC implementa um novo protocolo denominado WMTS (Web Map Tile Service) cuja lógica assenta na divisão das imagens a servir ao cliente em sub-imagens ou mosaicos, os quais seriam apenas parcialmente *renderizados* do lado do servidor, ficando a cabo do utilizador, mais concretamente ao *browser*, a conclusão desta tarefa (Open Geospatial Consortium Inc., 2010).

Numa terceira fase, e no seguimento da adoção de tecnologias como o AJAX (Asynchronous JavaScript and XML) e a vulgarização das APIs (Application Programming Interfaces), WebSIGs como o Google Maps permitiam a visualização contínua de imagens raster, ou seja, contrariamente ao paradigma anterior em que os pedidos era feitos de forma síncrona ao servidor, sendo portanto necessário esperar que este processasse os dados do pedido e devolvesse o resultado final ao utilizador, a tecnologia AJAX permitiu o mesmo resultado mas de forma assíncrona, podendo o utilizador interagir com o resto do SIG enquanto o resultado pedido era processado aos poucos (Tan et al., 2008). As APIs vieram dar resposta à necessidade de acesso por parte do cidadão comum às Bases de Dados de inúmeras organizações, contribuindo assim para uma maior democratização no acesso à informação, elemento chave da era da Big Data em que atualmente se vive. Foi este último desenvolvimento que permitiu a criação de *mashups*, ou aplicações Web que permitiam o cruzamento de informações provenientes de várias fontes, o que no caso dos SIG se traduziu em mapeamento Web com um maior grau de liberdade de escolha para o utilizador no que concerne às camadas de informação disponibilizadas no ecrã, sendo um bom exemplo disto o Google Map Maker, extinto em 2017 mas na sua essência ainda ativo através do Google My Maps (Li & Gong, 2008). Nesta plataforma é não só possível a criação de camadas interagindo diretamente com o mapa base do Google Maps, como também a importação de informação vetorial nos formatos KML ou GPX, alfanumérica em CSV ou XLSX, e ainda fotografias geo-referenciadas do álbum de fotos do utilizador Google.

Este foi também o início da era participativa da Web, não só marcada pela vulgarização dos Web Blogs e das redes sociais como também, ao nível dos SIG, pelas iniciativas de mapeamento voluntário como o Open Street Map, fundado em 2004, e cuja API já foi integrada não só em aplicações de navegação como o Maps.Me ou o OSandM, como também em plataformas de mapeamento cujo foco é a edição da Base de dados do OSM com vista ao mapeamento de zonas ou até de países inteiros afetados por catástrofes naturais, onde muitas vezes a cartografia oficial é escassa. Um bom exemplo de uma utilização pro bono da API anterior é a plataforma Humanitarian Open Street Map, onde as tarefas de mapeamento são por sua vez divididas em subtarefas ou áreas mais pequenas com um propósito de mapeamento muito específico, tal como a digitalização de edifícios ou de estradas, tendo por base as imagens de satélite do Microsoft Bing ou por vezes até da MAXAR, imagens de alta resolução espacial. Em boa verdade toda esta edição é passível de ser executada por qualquer pessoa com uma conta no OSM, mas estas iniciativas de mapeamento estão estruturadas de forma a priorizar as zonas onde as necessidades cartográficas são maiores ou mais recentes, dividindo as tarefas por níveis de experiência, – Iniciado, Intermédio e Avançado -, e dando apoio a qualquer utilizador através de fóruns para o esclarecimento de dúvidas sobre a edição do mapa. No final, todas essas modificações terão de ser aprovadas pelos membros mais experientes, de forma a garantir a fiabilidade destes dados geográficos.

Com a massificação dos Smartphones, a importância dos SIG aumenta com o uso de aplicações móveis que tiram proveito da capacidade de localização do dispositivo com recurso a SNGS (Serviços de Navegação Global por Satélite) como o GPS ou mais recentemente o Galileu. Olhando em específico para as aplicações de mapeamento como o Google Maps, esta consegue não só apresentar a localização do utilizador no mapa base, como permite também algumas *queries* baseadas nessa mesma localização, como a listagem por ordem de proximidade de edifícios orientados para um determinado propósito, ou o cálculo de rotas até um determinado destino, parametrizado por aspetos como estradas a evitar ou paragens intermédias obrigatórias. Outra vantagem mais controversa mas não menos interessante desta aplicação Google é a chamada Linha Cronológica que, uma vez ativada, dá permissão à Google para o armazenamento da informação vetorial gerada pelo utilizador, ou seja dos locais visitados, caminhos percorridos, tempos gastos em cada uma dessas atividades e até dos modos de transporte utilizados para essas deslocações, sendo o utilizador livre de corrigir qualquer parte desta informação na eventualidade da aplicação ter cometido um erro, como por exemplo, na atribuição do local onde o utilizador esteve num determinado período de tempo, algo passível de acontecer sobretudo em zonas onde

a reflexão do sinal do SNGS possa ser maior ou onde a ligação à Internet seja mais fraca. Todo este armazenamento serve não só para consultas simples pelo utilizador acerca do tempo que passou em determinado sítio numa data específica, mas providencia também formas mais interessantes de consumir estes dados, como através de relatórios genéricos de atividade mensal produzidos pela própria aplicação. De notar também que o utilizador é livre de a qualquer momento fazer o download de toda a sua informação recolhida, podendo esta informação em formato vetorial ser posteriormente trabalhada em softwares SIG como o QGIS ou até disponibilizada em WebSIGs. Há no entanto aplicações móveis desenhadas para utilizadores com um maior conhecimento na área da informação geográfica, e portanto mais focadas na recolha de dados ou até na edição de informação já existente, onde por vezes são necessárias correções que só podem ser feitas no terreno, acabando o smartphone por servir de substituto, ainda que limitado, de recetores RTK (Real Time Kinematic). Um bom exemplo destas aplicações é o QField, baseada no QGIS. Com esta App móvel é possível visualizar e editar dados geográficos já presentes no armazenamento interno do Smartphone, desde acrescentar pontos a camadas com essa geometria, a inserir trajetos gravados pelo próprio utilizador em camadas com geometria de Linha.

Por vezes, devido ao elevado volume de dados a ser apresentado em WebSIGs, os servidores convencionais para alojar as aplicações podem revelar-se incapazes na apresentação dos resultados de geoprocessamento numa janela temporal razoável para o utilizador. Assim, os gigantes tecnológicos como a Amazon, Google e Microsoft começaram a disponibilizar serviços de utilização dos seus servidores neste sentido. Um exemplo disto é o Harvard World Map, uma plataforma de disponibilização de WebSIGs alojada nos servidores da ESRI, nomeadamente na plataforma ArcGIS online, a qual partilha o nome com o software SIG principal desta empresa.

Por fim, os geoportais merecem também uma menção nesta revisão histórica dos SIG na web pois representam uma nova forma de disseminação de informação geográfica, em muitos casos proveniente de fontes oficiais, e cujas funcionalidades evoluíram não só para permitir a visualização desses dados, como também para facilitar a extração de grandes volumes destes através de APIs próprias, por sua vez manipuláveis através de programas externos sob o controlo do utilizador como scripts Python. Neste aspeto pode destacar-se o GeoPortal INSPIRE (Infrastructure for Spatial Information in the European Community) da União Europeia, criado ao abrigo da diretiva de 2007 do mesmo nome que se propõe a facilitar e a homogeneizar os formatos dos dados geográficos produzidos pelos estados membros, não só no sentido de maior interoperabilidade, como também de facilitação na tomada de decisões a nível climático. Ao nível

Humanitário destaca-se o geoportal Humanitarian Data Exchange da OCHA (Office for the Coordination of Humanitarian Affairs), um agregador de dados geográficos e alfanuméricos à escala nacional e sub-nacional para uma variedade considerável de países. Entre as fontes de dados incluem-se o Open Street Map, a FAO (Food and Agriculture Organization), a Cruz Vermelha, a UNHCR (United Nations High Commissioner for Refugees), a UNICEF (United Nations Children's Fund), o Banco Mundial, e a própria OCHA sob a forma de departamentos regionais para os vários países onde opera.

Feito o enquadramento acima, nas secções seguintes apresentam-se três WebSIGs que se consideraram suficientemente exemplificativos – em termos de tecnologias envolvidas, funcionalidades e entidades responsáveis – daquilo que se pretende com esta dissertação.

2.2. WebSIGs semelhantes ao proposto

De forma a melhor se entender o objetivo desta dissertação, procedeu-se à análise de alguns WebSIG que se julgam semelhantes ao proposto sob a perspetiva da visualização dos dados geográficos, ficando estes aquém na componente do geoprocessamento.

2.2.1. Armed Conflict Location & Event Data Project (ACLED)

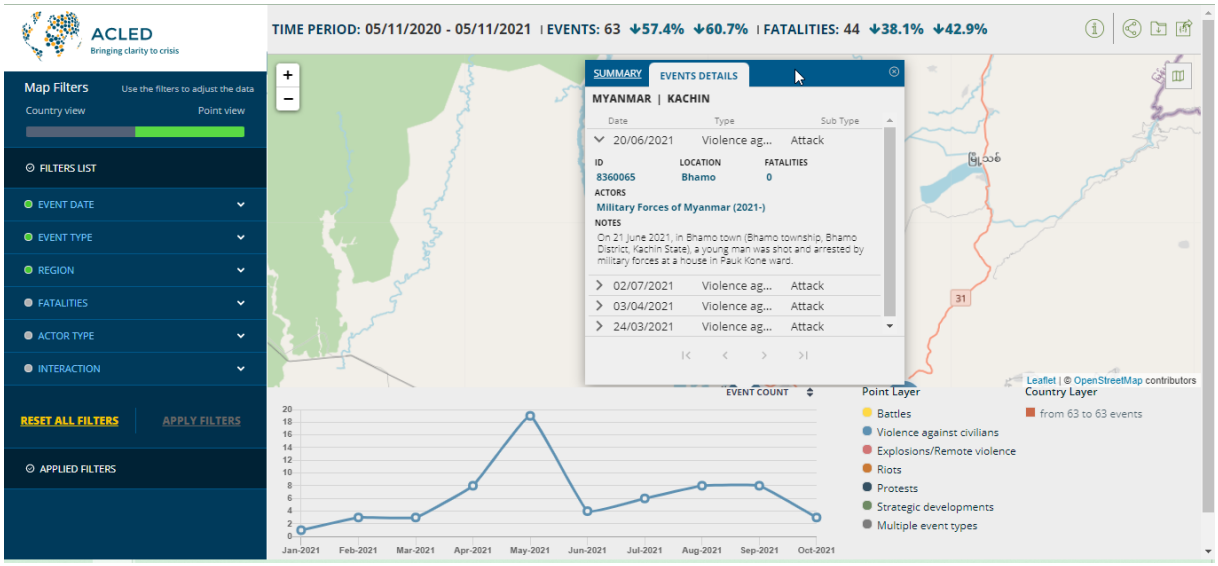

Fig. 1 – Incidentes de violência contra civis no estado de Kachin, Myanmar.

A ACLED é uma organização sem fins lucrativos que recolhe, analisa e disponibiliza informação sobre conflitos armados em várias partes do Globo (ACLED, 2021).

Como se pode ver pela captura de ecrã acima, o WebSIG dispõe de vários filtros indicados na lista à esquerda, tendo-se selecionado apenas, para efeito deste exemplo, os eventos de violência contra civis, representados sob a forma de círculos azuis no mapa. Quanto maior a dimensão, maior o número de eventos catalogados para aquela zona específica. Na tabela dos detalhes é possível ver a zona administrativa em que decorreu o incidente, a data, o subtipo de incidente e um breve relato do sucedido. Este WebSIG, apesar da dimensão Humanitária que lhe está associada, revela alguma carência a nível vetorial, podendo ter disponibilizado pelo menos uma camada vetorial extra para o nível das divisões administrativas imediatamente seguinte ao nacional, o que para o Myanmar corresponderia à divisão dos estados.

2.2.2. Harvard World Map

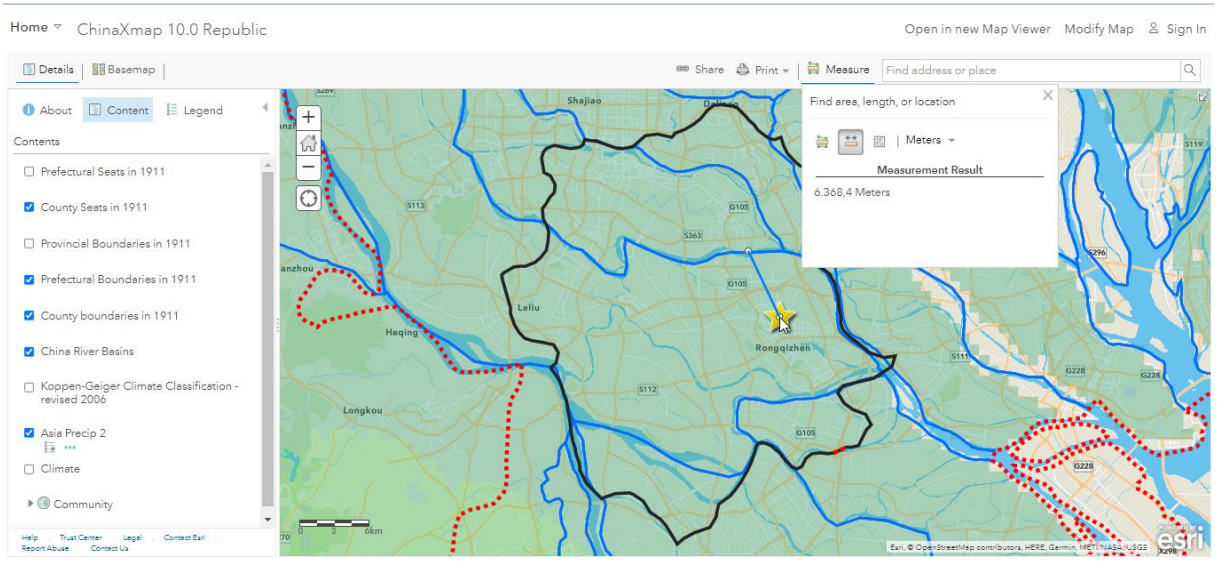

Fig. 2 – Mapa de precipitação do condado de Shunde Xian.

Tal como referido anteriormente, o Harvard World Map é uma plataforma SIG com fins educativos alojada na Cloud da ESRI (Harvard Map). Assim, é possível aos utilizadores da plataforma criarem os seus próprios WebSIGs carregando a informação geográfica que acharem pertinente. No exemplo acima apresenta-se um SIG designado "ChinaXmap 10.0 Republic", no qual estão disponíveis camadas vetoriais de geometria de ponto, linha e polígono, nomeadamente

as sedes de condado, as bacias hidrográficas e os limites administrativos, respetivamente. Dispõe também de camadas em formato matricial, como a relativa à precipitação em milímetros no continente asiático, numa gradação de cores de tons mais claros para mais escuros, indicando maior volume de precipitação. Este WebSIG goza de um maior poder de customização do que o anterior, podendo o utilizador não só escolher os objetos de cada camada que pretende visualizar através de um filtro aplicado sob a tabela de atributos, bem como personalizar ao seu gosto as cores e o estilo do traço no caso de objetos de geometria poligonal e linear, como até o símbolo usado para objetos de geometria de ponto. No exemplo acima optou-se pela análise do condado de Shunde Xian, na prefeitura de Guangzhou Fu, província de Guangdong. Os limites das prefeituras e dos condados encontram-se a vermelho e preto, respetivamente, estando as bacias hidrográficas a azul e as sedes dos condados representadas pelas estrelas amarelas. Graças à funcionalidade de medição de distâncias, é possível ter uma ideia da distância entre um ponto da margem do rio a norte da sede do condado, e do centro geográfico da própria sede, revelando uma distância de aproximadamente 6 Km, informação que poderá ter algum interesse numa situação de cheia numa das zonas da China com maior probabilidade de sofrer deste fenómeno, pelo menos a avaliar pela informação Raster disponível neste SIG. Apesar das capacidades de visualização similares às do ArcGIS, este SIG continua a não apresentar ferramentas de geoprocessamento que teriam sido úteis para esta análise, como a criação de máscaras em formato vetorial para zonas do condado com níveis de precipitação acima de um certo valor, ou até a nível da visualização, seria interessante a capacidade de filtrar os cursos de agua que passassem apenas no condado escolhido, ou até a criação de buffers em torno desses rios escolhidos de modo a melhor se perceber quais seriam as zonas de inundação em caso de cheia, evitando-se assim a geração de camadas temporárias no SIG.

2.2.3. National Map (do governo australiano)

Fig. 3 – Mapa das fazendas em zona de risco de incêndio.

O governo australiano disponibiliza o acesso gratuito ao seu webSIG, o "National Map", um SIG bastante completo no que toca à informação disponível para visualização, informação essa que se encontra dividida por categorias, como Agricultura, Energia ou Saúde, totalizando 17 categorias com informação tanto em formato vetorial como matricial (Australia National Map). Devido ao elevado volume de informação geográfica disponível, teria sido útil a possibilidade de filtrar a informação pelo seu formato bem como funcionalidades extra para alterar a cor e o traço de alguns objetos que não puderam constar do exemplo acima devido a cores padrão de baixo contraste com o mapa base. Atendendo ao clima tórrido que se vive na Austrália durante o Verão, optou-se pelo risco de incêndio como tópico de análise das capacidades deste SIG. Assim, e depois de uma consulta exaustiva do catálogo de informação geográfica disponível pertinente para o tópico em questão, optou-se pelas camadas de informação seguintes:

- Depósitos de água, representados por quadrados em cor turquesa;
- Serviços de emergência, nos quais se incluem Quartéis de Bombeiros de meio rural e serviços de transporte de doentes, representados pelos círculos RF e A, respetivamente;
- Fazendas, representadas pelos círculos a amarelo;
- Hospitais, representados pelos "H"s;
- Locais de fraca cobertura de rede móvel, representados pelos pontos a azul;
- Rios e afluentes, representados pelas linhas a azul;

- Vias de Comunicação, representadas pelas linhas a laranja e a cinza para as estradas principais e secundárias, respetivamente;
- Zonas com risco de incêndio na região sudoeste do país, representadas pela mancha a verde.

Dado que este WebSIG dispõe também de uma funcionalidade de medição de distâncias, ainda que pouco precisa, poderia ser usada para estimar as distâncias e os tempos necessários para os bombeiros ou os serviços de transporte de emergência chegarem a qualquer uma das dez fazendas visíveis no mapa em caso de incêndio, bem como os cursos de água ao longo do trajeto em caso de necessidade de abastecimento. Este SIG, tal como os anteriores, peca pela falta de ferramentas de geoprocessamento como análise de redes, o que neste caso teria sido útil para um cálculo mais preciso de rotas, ou ainda uma função de *clip* para os objetos vetoriais apenas para a zona de estudo, facilitando assim a manipulação do mapa.

2.3. Tecnologias e componentes dos WebSIGs anteriores

Tanto quanto foi possível apurar, os WebSIGs anteriores fizeram uso das seguintes tecnologias:

- Open Street Map
- Leaflet
- Carto / ArcGIS Online

Como já visto anteriormente, o Open Street Map constitui a mais ampla base de dados geográficos aberta (Fairhurst, 2020), disponível não só para consulta como também para edição. Neste caso, procedeu-se à sua integração nos webSIGs para efeitos de visualização, servindo assim de mapa base à informação geográfica visualizada.

O Leaflet JS é uma biblioteca JavaScript orientada para a visualização e estilização de informação geográfica, tanto vetorial como *raster*. É esta a componente tecnológica que, do lado do utilizador via Web Browser, irá fazer a sobreposição das várias camadas de informação e atribuir-lhes o estilo definido nos respetivos metadados (Agafonkin, 2021).

Por fim, o Carto DB e o ArcGIS Online funcionam como soluções SIG na nuvem, permitindo não só hospedar os WebSIGs como também tirar partido do poder de processamento das máquinas destas empresas para efeitos de geoprocessamento, operações que, para grandes quantidades de dados, se podem tornar morosas para processadores convencionais.

3. Modelo Conceptual do sistema

3.1. Esboço do modelo conceptual

O paradigma atual no que ao desenvolvimento Web concerne divide-se em duas estratégias de *design*: MVC (Model View Controller) e MVP (Model View Presenter).

A componente Model corresponde ao processamento dos dados bem como ao seu armazenamento na base de dados e respetivo acesso via *queries* SQL. É ainda esta componente que irá devolver os resultados das inquirições anteriores à base de dados através da componente View mas completamente desprovidos de qualquer simbologia cartográfica. Assim, esta componente funciona como o cerne de todo o processo, atualizando todas as instâncias da componente View com cada pesquisa submetida.

A componente View corresponde à apresentação gráfica dos dados obtidos anteriormente *renderizados* de acordo com as escolhas do utilizador submetidas na componente Controller. Esta componente é independente de qualquer sistema operativo, focando-se assim em apresentar a informação em formatos já consagrados no quotidiano web como HTML para informação alfanumérica ou WMS para informação geográfica, já abordada no capítulo anterior. Assim, esta componente está permanentemente preparada para responder a qualquer solicitação por parte do utilizador, atualizando-se instantaneamente para refletir as respetivas mudanças pedidas.

A terceira e última componente varia de acordo com a estratégia de *design*, mas na primeira existe uma componente Controller, a qual serve de intermediária entre o utilizador e o Model, submetendo a este último a respetiva pesquisa do utilizador. Na segunda estratégia, a terceira componente será o Presenter, servindo agora de intermediária entre a View e o Model, impedindo assim qualquer comunicação entre as duas últimas componentes.

Apesar da opção MVP ser menos complexa devido à centralização da lógica da aplicação na componente Presenter, optou-se pela abordagem MVC, pois havendo mais componentes distintas, a tarefa de testagem torna-se mais simples por se tratar de componentes mais simples. (Daodi et al., 2019)

À luz do exposto acima, a arquitetura pensada para este projeto foi a seguinte:

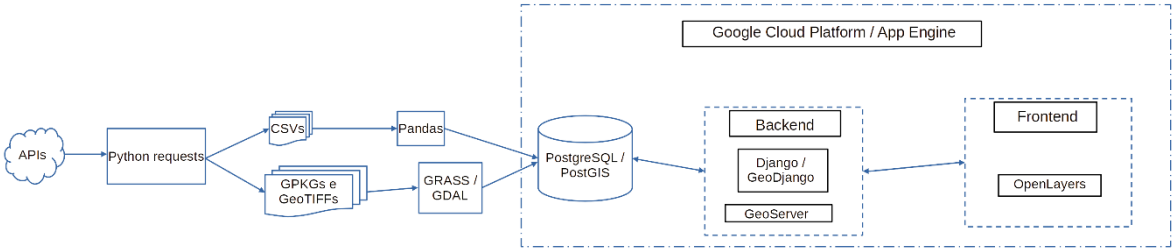

Fig. 4 – Diagrama de fluxo com a arquitetura do WebSIG proposto

3.2. Descrição detalhada do produto inicialmente pensado, de todas as suas componentes e tecnologias

Tal como os WebSIGs mencionados anteriormente, o sistema inicialmente pensado teria os seguintes componentes:

- Dados alfanuméricos e geoespaciais,
- Uma base de dados com capacidade de armazenamento de ambos os tipos mencionados acima,
- Um servidor onde a aplicação possa estar alojada e posteriormente acedida pelo utilizador via Internet.

3.2.1. Fontes e tipos de dados

Tal como mencionado acima, os dados inicialmente pensados seriam tanto de natureza alfanumérica como geoespacial, pelo que se começará por apresentar os do primeiro tipo. Inicialmente pensou-se em aceder a dados alfanuméricos provenientes de instituições fidedignas e com ampla experiência na recolha e tratamento destes dados para fins estatísticos. Essas organizações seriam:

- World Bank;
- UNSD (United Nations Statistics Department)

- FAO (Food and Agriculture Organization of the United Nations)
- HDX (Humanitarian Data Exchange)
- IFPRI (International Food Policy Research Institute)
- IRRI (International Rice Research Institute)
- UNICEF (United Nations Children's Fund)
- ILO (International Labour Organization)
- IMF (International Monetary Fund)
- WHO (World Health Organization)
- OECD (Organisation for Economic Co-operation and Development)
- UNEP (United Nations Environment Programme)
- UNHCR (United Nations High Commissioner for Refugees)
- WTO (World Trade Organization)

Devido ao elevado volume de dados disponibilizado por estas organizações, seria necessário arranjar uma forma eficiente de filtrar e adquirir a informação relevante para o contexto no qual a aplicação se inseria, daí se ter pensado em recorrer às respetivas APIs Web destas organizações. Uma API, ou do inglês *Application Programming Interface*, é um serviço oferecido, gratuitamente ou não, por uma entidade detentora de informação que a pretenda tornar acessível a um determinado grupo de utilizadores (Meng et al., 2017). Acaba por se traduzir numa ponte entre a base de dados dessa organização e o computador do utilizador, sendo esta ligação feita com recurso à internet. É a capacidade de aceder à informação via métodos HTTPS (Hypertext Transfer Protocol Secure) como o GET e de manipulação do URL (Uniform Resource Locator) para encaminhamento dos parâmetros de busca que atribui a estas APIs a arquitetura REST (Representational State Transfer). Tal arquitetura respeita as seguintes especificações:

- Separação entre a base de dados da aplicação e o *interface* do utilizador para aceder à informação,
- Como o protocolo base das API REST é o HTTP, um protocolo que não retém os dados de sessão do utilizador (*stateless)*, verifica-se uma maior performance em aplicações

externas que necessitem de aceder a grandes volumes de informação através deste tipo de APIs,

- Capacidade de armazenar temporariamente na máquina do utilizador informação previamente acedida (*web caching*),
- Possibilidade de escolha do formato da representação dos dados pretendidos. Ex: JSON, CSV ou XML. (Manuaba & Rudiastini, 2018)

A título de exemplo, pode referir-se a API do World Bank. (WorldBank)

Esta API disponibiliza o acesso a inúmeros indicadores em áreas como Economia, Saúde ou Capital Humano (Worldbank) para a maioria dos estados-membros das Nações Unidas, recolhidos ao longo de décadas. O acesso a esta informação seria feito através da edição do URL do banco mundial. Suponha-se que era pretendida a área agrícola como percentagem da área total do Myanmar e do Bangladesh para o ano de 2016. O URL a ser usado seria o seguinte: http://api.worldbank.org/v2/country/bgd;mmr/indicator/AG.LND.AGRI.ZS?date=2016&format=json

Dissecando o URL, obter-se-ia:

- http://api.worldbank.org/v2/country/: O tronco comum a qualquer pedido;
- bgd;mmr: os códigos ISO3 representativos do Bangladesh e Myanmar, respetivamente;
- AG.LND.AGRI.ZS: o código do indicador em causa
- Tudo o que vier a seguir ao ponto de interrogação são parâmetros adicionais para refinar a pesquisa. Neste caso:

  Date=2016 e format=json: Para indicar que se pretende o resultado do ano de 2016 e para que o resultado final venha em formato JSON, mais facilmente trabalhável em Python do que o XML, por exemplo.

Colocando diretamente a hiperligação acima num qualquer *browser* é possível aceder à informação pretendida em formato JSON, um formato de dados não associado diretamente a uma linguagem de programação específica mas assente em estruturas de dados familiares às linguagens que derivaram de C, como Python ou JavaScript, da qual deriva o nome JSON (JavaScript Object Notation). Estas estruturas de dados correspondem aos dicionários e às listas em Python. (JSON)

De forma a facilitar a visualização do conteúdo JSON, executou-se o seguinte procedimento em Python:

```
import requests, pprint
request = requests.get(
'http://api.worldbank.org/v2/country/bgd;mmr/indicator/AG.LND.AGRI.ZS?date=2016&format=json')
pprint.pprint(request.json())
```

Algoritmo 1 – Procedimento para obtenção da informação agrícola da API do World Bank.

O módulo Requests em Python permite o acesso a informação através de HTTPS fazendo uso do método GET discutido anteriormente, sendo necessário aplicar o método json() sobre o resultado obtido de modo a efetivamente se obter a informação pretendida nesse formato (Python Package Index). O módulo Pprint permite a apresentação vertical de listas e de dicionários, ao invés de na horizontal, facilitando assim a visualização do resultado final que se apresenta abaixo:

```
[{'lastupdated': '2021-09-15',
 'page': 1,
 'pages': 1,
 'per_page': 50,
 'sourceid': '2',
 'sourcename': 'World Development Indicators',
 'total': 2},
[{'country': {'id': 'BD', 'value': 'Bangladesh'},
  'countryiso3code': 'BGD',
  'date': '2016',
  'decimal': 1,
  'indicator': {'id': 'AG.LND.AGRI.ZS',
                'value': 'Agricultural land (% of land area)'},
  'obs_status': '',
  'unit': '',
  'value': 70.6323269570562},
 {'country': {'id': 'MM', 'value': 'Myanmar'},
  'countryiso3code': 'MMR',
  'date': '2016',
  'decimal': 1,
  'indicator': {'id': 'AG.LND.AGRI.ZS',
                'value': 'Agricultural land (% of land area)'},
  'obs_status': '',
  'unit': '',
  'value': 19.538188277087}]]
```

Fig. 5 – Saída gráfica dos resultados da execução do Algoritmo 1.

O resultado final corresponde a uma lista de dois elementos: um dicionário com metadados do pedido e uma lista de dicionários, um por país, nos quais se incluem o nome e código ISO do país, o indicador pretendido e claro, o valor para esse indicador.

Quanto aos dados geoespaciais, pensou-se em dados das seguintes categorias (indicando-se entre parêntesis o respetivo formato digital):

- Uso e ocupação do solo (*Raster*),
- Divisões Administrativas (Vetorial),
- População (*Raster*),
- Vias de comunicação (Vetorial),

- Cursos de água (Vetorial),
- Infraestruturas (Vetorial),
- Modelos digitais de terreno (*Raster*).

Os dados de uso e ocupação do solo seriam obtidos da GlobCover, uma iniciativa da ESA (European Space Agency) que a partir de imagens com resolução espacial de 300m obtidas pelo sensor MERIS (Medium Resolution Imaging Spectrometer) a bordo da missão ENVISAT (Environmental Satellite), produziu um mapa em formato vetorial para a totalidade do globo durante os períodos compreendidos entre dezembro de 2004 e junho de 2006, e entre janeiro e dezembro de 2009, contendo 22 classes de ocupação do solo, tal como constante do sistema de classificação da FAO. (Congalton et al., 2014) Versões a nível nacional seriam descarregadas a partir do geoportal da FAO (FAO), organização parceira da ESA nesta iniciativa.

Os dados populacionais foram obtidos a partir do World Population, um projeto sob a alçada da Universidade de Southampton orientado para o estudo da demografia, disponibilizando as suas previsões para cada país em produtos matriciais com uma resolução espacial de 100 ou 1000 metros, e temporal entre os anos de 2000 e 2020. O utilizador é também livre de escolher se quer os valores populacionais distribuídos pela totalidade do território do país em causa, ou apenas pelas áreas consideradas habitadas, de acordo com censos previamente realizados em cada país, e também com as estimativas populacionais das Nações Unidas (WorldPop).

A informação altimétrica foi obtida pela missão SRTM (Shuttle Radar Topography Mission), mais concretamente pelos seus dois sensores radar embarcados. Esta informação foi posteriormente processada pela USGS (United States Geological Survey), a maior agência americana focada no mapeamento dos recursos energéticos, hídricos e minerais desse país, resultando em modelos digitais de terreno divididos em três gamas de produtos com cobertura global mas com resolução espacial a variar entre 1 e 3 segundos de arco medidos no Equador. (USGS).

Quanto à informação vetorial, o OpenStreetMap é de facto único no que toca ao volume e disponibilidade deste tipo de dados para o mundo inteiro, tal como já abordado no capítulo anterior. Tendo isto em conta, os dados das vias de comunicação, cursos de água, infraestruturas e divisões administrativas seriam todos obtidos a partir da API do OSM.

3.2.2. Processamento dos dados

Uma vez obtidos, os dados acima necessitariam de ser devidamente preparados antes de inseridos numa base de dados. Para esse efeito pensou-se no pacote Pandas, uma biblioteca Python amplamente usada em análise de dados para fins estatísticos e capaz de manipular ficheiros CSV com centenas de milhares de células. (Pandas)

Quanto aos dados geoespaciais, optou-se pelas bibliotecas GDAL (Geospatial Data Abstraction Library) e GRASS (Geographic Resources Analysis Support System) por não só constituírem *software* aberto como também pela familiaridade já adquirida durante o 1º ano do MTIG. Ambas as bibliotecas dispõem de uma ampla gama de algoritmos para processamento tanto de dados vetoriais como matriciais, e representam peças extremamente importantes em pacotes abertos de informação geográfica como o OSGeo4W, uma iniciativa da OSGeo (Open Source Geospatial Foundation), e provavelmente o maior concorrente do ArcGIS da ESRI. (OSGeo)

3.2.3. Base de dados

Uma vez tratados, seria necessário inseri-los numa base de dados. Continuando na mesma ótica de manter tudo FOSS (Free and Open Source Software) optou-se pela PostgreSQL, uma base de dados relacional orientada a objetos. Para além de ser de uso livre, esta base de dados dispõe ainda de uma extensão designada PostGIS com capacidade para armazenar objetos geoespaciais. (PostgreSQL)

3.2.4. Backend

O pacote Django é uma das *frameworks* mais populares na área do desenvolvimento de aplicações Web. O Django está implementado em linguagem Python e é estruturado segundo o modelo MTV (Model, Template, View). A componente Model corresponde às estruturas de dados, nomeadamente às Classes, sendo cada instância de uma classe um atributo de todos os objetos pertencentes a essa mesma classe, não fosse o Python uma linguagem orientada a objetos. O Template é uma ferramenta que permite definir, ainda que de forma simplista, a componente visual da aplicação, recorrendo apenas a HTML para estruturar os elementos na página. Por fim o View permite a visualização de tudo o que foi construído com as duas componentes anteriores. (Holovaty & Kaplan-Moss, 2008)

Assim, utilizando-se um pacote suficientemente acreditado no mundo do desenvolvimento Web, como é o caso do Django, é possível a criação expedita de aplicações com funcionalidades avançadas como o caso de Sitemaps ou a autenticação de utilizadores alavancada por uma base de dados como a referida na secção anterior. Apresenta também vantagens ao nível da segurança, protegendo a aplicação de ataques à base de dados via expressões SQL duvidosas ou protegendo os futuros utilizadores contra *malware* disfarçado de elementos da própria aplicação Web. Por fim, esta *framework* permitirá o acesso à aplicação por parte de uma grande quantidade de utilizadores sem que a mesma colapse devido a súbitos e inesperados aumentos de tráfego. (Linos et al., 2019)

O GeoDjango surge como resposta à necessidade de apresentar elementos de cariz geoespacial, permitindo uma ligação à base de dados e a bibliotecas de manipulação de informação geoespacial como as duas referidas anteriormente, e ainda outras como a PROJ, uma biblioteca para a conversão entre diferentes sistemas de coordenadas, ou a GEOS, uma das dependências mais importantes do PostGIS e do GDAL, responsável pelas operações geoespaciais em formato vetorial, cujos resultados poderá depois comunicar às componentes do Frontend como o OpenLayers ou o Leaflet via GeoServer. (Django)

O GeoServer é, tal como implícito no nome, um servidor web com capacidade para não só apresentar ao utilizador da aplicação mapas previamente compilados a partir de informação geográfica na base de dados, como também para tornar acessível essa mesma informação ao utilizador via formatos padrão pela OGC como o WMS, WFS ou WCS. Será esta componente a ponte entre o Backend e o Frontend da aplicação. (GeoServer)

3.2.5. Frontend

Uma vez acessível nos formatos OGC previamente apresentados, será necessário criar um mapa dinâmico com o qual o utilizador possa interagir com os dados do WebSIG. Para este projeto optou-se pela biblioteca OpenLayers por oposição à anteriormente mencionada Leaflet, pois a primeira demonstra ser mais orientada para a produção de mapas mais complexos, dispondo de mais funcionalidades para fazer o mesmo trabalho e estando capacitada com a tecnologia AJAX, permitindo assim reduzir o esforço de processamento do lado do servidor, tal como já mencionado no capítulo anterior. (Nash et al., 2008)

3.2.6. Hosting

Devido aos constrangimentos de manutenção de um servidor físico para alojar a aplicação, ou seja, de uma máquina ligada 24 horas por dia, constantemente ligada à Internet, vulnerável a ataques, e com o poder de processamento necessário para produzir os mapas pedidos pelo utilizador, demasiado tempo acabaria gasto em tarefas que transcendem o âmbito deste projeto. Assim, e com a popularidade que o processamento na Cloud ganhou nos últimos anos, optou-se por uma solução PaaS (Platform as a Service), na qual seja possível hospedar e tirar partido do poder de processamento das máquinas de gigantes tecnológicos como a Microsoft ou Amazon para executar a aplicação Web sem qualquer constrangimento quer para o criador, quer para o utilizador. Assim, optou-se pela App Engine, uma das componentes da Google Cloud Platform, acima de tudo pelo Google Earth Engine, plataforma que permite o acesso aos mais variados grupos de dados matriciais temáticos, desde imagens de satélite, cartas de uso e ocupação do solo ou até modelos digitais de terreno . (Google Cloud Platform)

4. Implementação do modelo físico do sistema

A implementação do modelo proposto revelou alguns desafios tanto ao nível da tecnologia como dos dados escolhidos na secção anterior. Ir-se-á por isso numa primeira fase explorar as alterações adotadas nestas duas vertentes e subsequentes metodologias e processos Python de obtenção, tratamento e de *upload* desses dados para uma base de dados geoespacial.

4.1. Alterações ao inicialmente proposto

À exceção da HDX (Humanitarian Data Exchange), as fontes de dados alfanuméricos previamente mencionadas na secção 3.2.1 do capítulo anterior, apesar de dotadas de uma grande variedade de indicadores sócio-económicos, pecaram pela falta de granularidade no que concerne aos níveis administrativos sub-nacionais, invalidando assim a sua utilidade no âmbito deste projeto. A somar a este problema denotou-se ainda uma inconsistência na disponibilidade de dados geolocalizados na própria plataforma web da HDX para os países do continente africano, tomando como exemplo, no caso de shapefiles e geopackages, a Etiópia com 114 e o Egipto com 5.

Ao nível da informação matricial, nomeadamente as cartas de uso e ocupação do solo da GlobCover, revelaram-se insuficientes para mapear de forma aceitável as pequenas parcelas agrícolas da maioria dos países visados, dado que a maior parte destas teria sensivelmente 0.5 hectares e a unidade cartográfica mínima para cartas com uma resolução espacial de 300 m nunca seria inferior a 10 hectares. (FAO)

Assim, de modo a garantir-se a viabilidade deste projeto, optou-se pela redução da área de estudo, virando-se agora o foco para um país ao invés de um continente inteiro. Na escolha do país pesou não só a disponibilidade e variedade de dados alfanuméricos e geoespaciais, como também a pertinência que a sua situação político-social atual pudesse ter no contexto humanitário. Posto isto, optou-se pelo Myanmar pelos seguintes motivos:

- Crise Rohingya despoletada em 2017,
- Golpe de estado em 2021,
- Cheias anuais,

- Portais de informação específicos deste país.

Breve contexto humanitário do Myanmar

A relação entre os arracaneses e os rohingya residentes no estado de Rakhine no Myanmar começou a deteriorar-se a partir de 2012, quando o primeiro grupo, maioritário no estado em termos populacionais, começou a temer que o segundo o ultrapassasse nesse domínio, algo difícil de aceitar para uma nação maioritariamente budista (Kipgen, 2014). Cinco anos volvidos, os conflitos entre estes dois grupos intensificaram-se, culminando na morte de dezenas de milhares de membros desta minoria pelas forças militares birmanesas, e despoletando assim a migração de centenas de milhares para o sudeste do Bangladesh. De acordo com o Alto Comissariado das Nações Unidas para os Refugiados, à data de 31 de julho de 2019, mais de 742.000 rohingya terão fugido para o distrito bangladês de Cox's Bazar em busca de asilo (UNHCR), escapando assim a uma limpeza étnica que conta já com pelo menos 24.000 mortos. (Habib et al., 2018) No entanto, nem todos os membros desta etnia optaram pela migração para o país vizinho, tendo sido prontamente reencaminhados de forma forçada para campos de asilo no distrito de Sittwe, ainda no estado de Rakhine. Estas instalações revelaram-se frágeis na sua qualidade de construção, dimensionadas de forma a obrigar as famílias que lá habitam a partilhar um espaço minúsculo, e muitas delas construídas a cotas muito próximas da do mar, aumentando significativamente o risco de problemas sanitários para os ocupantes em caso de cheias. (Human Rights Watch) Este último detalhe é particularmente agravado pela época das monções, entre junho e setembro, típica dos países do Sudeste Asiático, e caraterizada por níveis de precipitação anormalmente elevados. (Sen Roy & Surinder) No entanto, os rohingyas não são os únicos prejudicados por este fenómeno. Ainda no estado de Rakhine, mais de 9.000 arracaneses tiveram de ser evacuados das suas casas devido à subida do nível dos rios Lemyo e Kaladan, na sequência da chuva intensa que se verificou em julho de 2019, época de monções.(The Irrawaddy)

Em fevereiro do ano passado, o governo incumbente encabeçado pela prémio nobel da paz Aung San Suu Kyi foi deposto pelas forças armadas birmanesas, que até hoje controlam pela força este país. Este golpe de estado traduziu-se não só no agravamento das condições de vida dos mianmarenses em geral, mas também dos 130.000 rohingya atualmente detidos nos asilos em

Sittwe, por serem considerados uma ameaça ainda maior a um regime desesperado pela manutenção de poder que não lhe foi incumbido pela população (Aljazeera).

Face a todas estas dinâmicas, o Myanmar pareceu de facto o país ideal para este projeto, tendo-se optado pelas seguintes fontes de dados:

- A MIMU (Myanmar Information Management Unit) é uma agência ao serviço da missão das Nações Unidas no Myanmar com o objetivo de produzir e disseminar informação relevante para os atores não-governamentais de cariz humanitário presentes nesse país. Esta iniciativa é apoiada pela Comissão Europeia, Programa das Nações Unidas para o Desenvolvimento, o Governo do Canadá, entre outros. (MIMU)
  Desta fonte selecionaram-se os seguintes dados com a respetiva extensão entre parêntesis:
  - Divisões Administrativas (shapefile);
  - Listas de "P-codes" para todas os níveis administrativos do país (xlsx). Estes códigos equivalem aos códigos postais portugueses, permitindo assim uma maior organização logística do país;
  - Dados populacionais por agrupamento de aldeias / bairro (xlsx);
  - Instalações de saúde (xlsx);
  - Instalações de ensino (shapefile);
  - Localização dos campos de asilo no estado de Rakhine (csv).
- Do OpenStreetMap (OSM) retirou-se informação vetorial de vias de comunicação e de cursos de água (geopackage).
- Do United States Geological Survey (USGS) retirou-se o modelo digital de terreno para o Myanmar de 1-arco de segundo.
- Do World Population retirou-se um produto *raster* com resolução espacial de 100 metros.
- Da United Nations Satellite Center (UNOSAT), uma agência das Nações Unidas encarregue da análise de imagens de satélite na sequência de desastres ambientais, retirou-se a informação vetorial relativa a cheias no Myanmar.

4.2. Metodologia e Processamento

O Myanmar encontra-se atualmente dividido em 5 níveis administrativos:

- 1º Nível: 7 estados / 7 regiões / 1 território da união (o qual contem a capital, Naypyidaw) / 5 zonas autónomas / 1 divisão autónoma,

- 2º Nível: 75 distritos,
- 3º Nível: 330 concelhos,
- 4º Nível: 469 freguesias / 13590 agrupamento de aldeias
- 5º Nível: 3470 bairros (exclusivos das freguesias) / 63214 aldeias (exclusivas dos agrupamentos de aldeias). (MIMU)

Atendendo aos 130.000 rohingya atualmente detidos no concelho de Sittwe, optou-se por esta área de estudo para este projeto, incluindo a sua capital, a freguesia de Sittwe, por sua vez dividida em 33 bairros, sem esquecer ainda as suas 88 aldeias espalhadas por 28 agrupamentos.

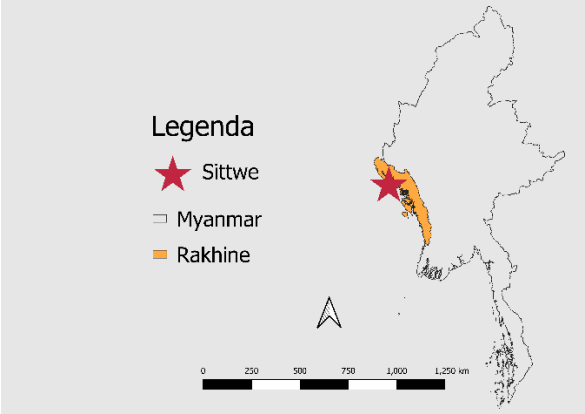

Fig. 6 – Localização de Sittwe no estado de Rakhine, Myanmar

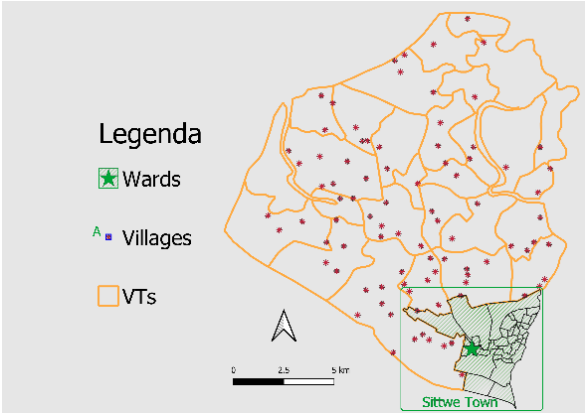

Fig. 7 – Agrupamentos, aldeias e bairros de Sittwe

4.2.1. Obtenção da Altimetria

Através do website https://earthexplorer.usgs.gov/ é possível descarregar mosaicos de informação altimétrica com resolução espacial de 1 arco de segundo ou aproximadamente 30 metros a latitude zero. Assim, submeteu-se um arquivo zip contendo quatro ficheiros shapefile do polígono do concelho de Sittwe que por sorte continha apenas 413 vértices, perto do limite de 500 imposto pela USGS. Esse ficheiro serviu de critério para a pesquisa do mosaico/s que melhor se encaixassem na extensão geográfica do polígono em questão, o qual acabou por ser o mosaico intitulado "n20_e092_1arc", designação atribuída ao canto inferior esquerdo do mesmo. De seguida o ficheiro GeoTIFF correspondente foi obtido através do Bulk Download, um software da autoria da USGS capaz de descarregar grandes volumes de dados matriciais. Uma vez carregado no QGIS, este mosaico de banda única apresenta-se em tons de cinza, em que cada pixel representa um valor de altitude em metros para uma área total de aproximadamente 900m^2. De acordo com os metadados da imagem, apesar desta apresentar uma resolução radiométrica de

16 bit para valores inteiros positivos e negativos, ou seja um intervalo de valores entre -32.768 e 32,767, os valores de pixel mínimo e máximo fixam-se em -14 e 879, respetivamente, e a média ronda o valor 26. No entanto estes valores refletem uma área muito superior à da de estudo, pelo que se procedeu ao recorte do mosaico através do algoritmo do GDAL "Clip raster by mask camada" tendo o polígono de Sittwe como máscara de entrada. Após inspeção dos metadados do resultado, constata-se que os valores mínimo e máximo se fixam em -10 e 36, respetivamente, e a média em cerca de 3 metros, algo coerente com a realidade no terreno atendendo a que se trata de uma zona costeira, e corroborando assim a ideia de que as inundações em tempo de monção nesta zona poderão mesmo ser devastadoras para os residentes de Sittwe.

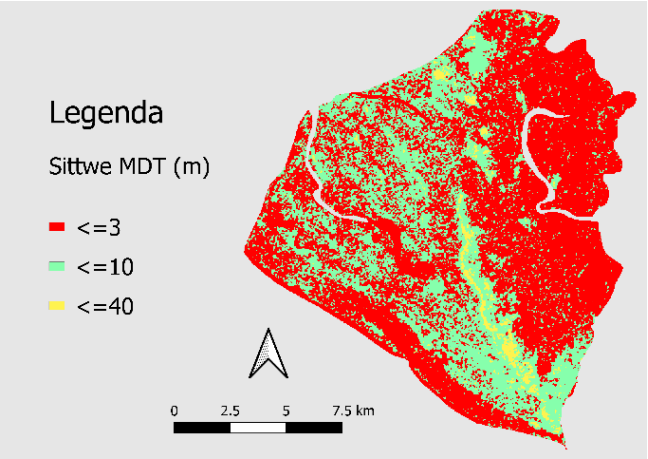

Fig. 8 – Composição colorida do Modelo Digital do Terreno de Sittwe

4.2.2. Obtenção dos dados populacionais

Através do projeto World Data obteve-se o produto matricial "Myanmar Population Unconstrained 2020 UN adjusted". Este *raster* apresenta uma resolução espacial de 100 metros medidos no equador, em que para cada pixel se apresenta uma estimativa do número de habitantes ajustado às previsões das Nações Unidas para 2020. Dos metadados da imagem é também possível concluir que o valor máximo ronda os 158 habitantes e o médio os 0.5. No entanto, tal como para a informação altimétrica, será necessário recortar a imagem pelo perímetro do concelho de Sittwe, pelo que se recorreu ao algoritmo anterior. O novo *raster* apresenta agora um valor de píxel médio próximo dos 5.5 habitantes e confirma que para a extensão deste projeto, a freguesia de Sittwe é de facto a zona mais densamente povoada de todo o concelho.

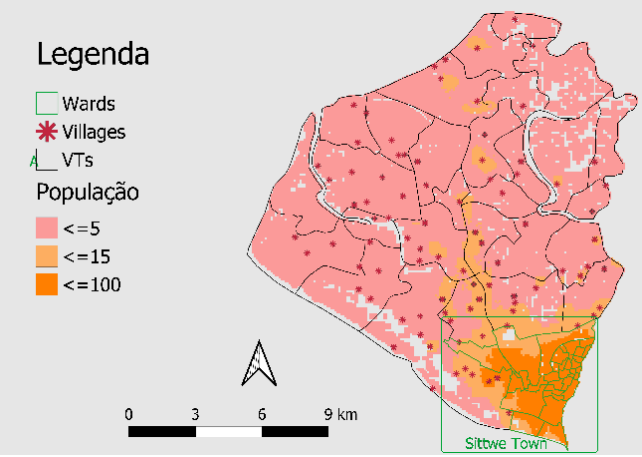

Fig. 9 – Distribuição populacional de Sittwe

4.2.3. Informação vetorial

Devido ao elevado volume de informação conseguido para o Myanmar, seria vantajoso cingi-lo à área de estudo, acima de tudo para reduzir ao mínimo indispensável o espaço ocupado na base de dados e assim facilitar toda e qualquer operação de pesquisa ou de geoprocessamento nela executada.

Outro aspeto importante quanto a este tipo de informação seria a uniformização do formato em que esta se encontra armazenada. Assim, optou-se pela geopackage por se tratar de um formato de dados geoespaciais:

- Aberto, na medida em que foi desenvolvido pelo OGC e é, tal como todas as outras convenções desta organização, vocacionado para um uso universal e sem restrições de acesso;
- É em si mesmo uma base de dados baseada em SQLite, com uma sintaxe genericamente similar à da PostgreSQL. Assim, permite o armazenamento de várias tabelas com geometrias diferentes e até a configuração do estilo relativo a cada camada, ao contrário do formato shapefile da Esri que permite armazenar apenas uma tabela por ficheiro.
- Outra vantagem da geopackage sobre a shapefile é a existência de um ficheiro único com extensão gpkg onde toda a informação acima se concentra, por oposição ao shp, o ficheiro que contem a informação geométrica de cada objeto, e que terá de vir acompanhado de pelo menos dois outros ficheiros, nomeadamente: um shx, correspondente ao índice que permite pesquisa de objetos, e um dbf, a tabela com os

atributos de cada objeto dessa shapefile. Adicionalmente poderão ainda constar do arquivo shapefile os ficheiros: sbn e sbx, xml, prj, e cpg, correspondentes ao índice espacial que permite pesquisa de objetos através da sua geometria, ao ficheiro dos metadados do arquivo, o ficheiro dos metadados do sistema de coordenadas, e o ficheiro relativo à codificação de carateres constante dos atributos do arquivo (ESRI). A ausência deste último ficheiro revelou-se outro fator a favor do formato geopackage, na medida em que a toponímia birmanesa aparecia distorcida na tabela de atributos em formato shapefile.

Posto isto, desenhou-se o procedimento python de nome "filterAndConvert2Geopackage", contido no Anexo A, e que para um dado diretório contendo, entre outros, ficheiros shp e csv, filtrasse os objetos que estivessem contidos num outro objeto de referência, sendo para o efeito deste projeto, o polígono do concelho de Sittwe, e por fim guardasse esse grupo de objetos num ficheiro geopackage. De forma a garantir uma execução eficiente e sem percalços, tomou-se em conta o seguinte na conceção deste algoritmo:

- O objeto de referência seria proveniente de uma geopackage, pelo que se começou por executar o algoritmo apenas para a camada dos concelhos do Myanmar sem qualquer filtro.
- Cada conjunto de dados deverá ter a sua própria pasta. Estas pastas deverão por sua vez ser colocadas num mesmo diretório, o qual será percorrido pelo algoritmo em busca de shapefiles ou CSVs.
- No final da conversão de cada ficheiro elegível, todos os ficheiros dessa pasta serão apagados.
- O sistema de coordenadas de todas as camadas será o WGS 84, com código EPSG 4326.
- Qualquer ficheiro csv terá os campos Longitude e Latitude e será relativo apenas a objetos de geometria de ponto.

O algoritmo seria o seguinte:

- Para cada pasta no diretório:
    - Para cada ficheiro da pasta:
        - Se a sua extensão for shp ou csv:
            - Se a sua extensão for shp:
                - Abrir a camada em modo de leitura
            - Se a sua extensão for csv:
                - Abrir o csv e guardar todos os seus registos numa variável
        - Criar um ficheiro geopackage onde se irão armazenar os objetos filtrados
        - Se o algoritmo estiver a ser executado em modo de filtragem:
            - Abrir a camada com o objeto de referência
            - Guardar o objeto numa variável
            - Se a extensão do ficheiro for csv ou se a geometria dos objetos dessa camada for Ponto ou Multi Ponto:
                - Adicionar à geopackage apenas os objetos geometricamente contidos no objeto de referência
            - Se a geometria dos objetos for de linha ou de polígono:
                - Adicionar à geopackage apenas os objetos parcialmente contidos no objeto de referência, mas apenas a parte de cada um totalmente contida no objeto de referência.
        - Se o algoritmo não estiver a ser executado em modo de filtragem:
            - Adicionar à geopackage todos os objetos da camada de entrada
    - Para cada ficheiro na pasta:
        - Apagar o ficheiro

4.2.4. Correção da localização de algumas aldeias

Para facilitar os processos seguintes, criou-se uma nova geopackage contendo as tabelas das aldeias, agrupamentos e bairros, por sua vez os níveis administrativos de maior relevância para este projeto.

Uma vez carregadas para o QGIS as camadas de informação já filtradas, constatou-se que algumas aldeias se encontravam fora do agrupamento que lhes havia sido atribuído na tabela de atributos. De modo a se determinar quantas e quais as aldeias nesta situação, adicionou-se uma segunda camada de estilo à das aldeias de modo a que se gerasse um pequeno buffer em torno dessas aldeias por forma a mais facilmente identificar todas os objetos nesta situação. Para essa camada de estilo capaz de gerar geometrias usou-se a seguinte expressão QGIS:

```
CASE WHEN
    NOT contains(
        geometry(
            get_feature('VT_layer_id', 'VT_PCODE', attribute('VT_Pcode'))
            ), $geometry)
    THEN buffer($geometry, 0.005, 8)
    END
```

Sendo:

- VT_layer_id = o identificador atribuído pelo QGIS à camada dos agrupamentos de aldeias,
- VT_PCODE = o campo com os códigos dos Agrupamentos constante dessa camada,
- VT_Pcode = o campo com os códigos dos Agrupamentos atribuídos a cada aldeia nessa camada

Ou seja, a expressão acima estabelece uma condição em que, para todos os objetos da camada das aldeias que não estejam contidos no agrupamento em que supostamente deveriam estar por força do valor que lhes foi atribuído para esse campo na tabela de atributos, lhes seja acrescentado um *buffer* de raio igual a 0.005 graus com oito segmentos.

Acrescentaram-se ainda etiquetas a todas os agrupamentos e aldeias com os nomes dos respetivos agrupamentos que lhes foram atribuídos na tabela de atributos, de modo a facilitar a visualização do fenómeno em causa.

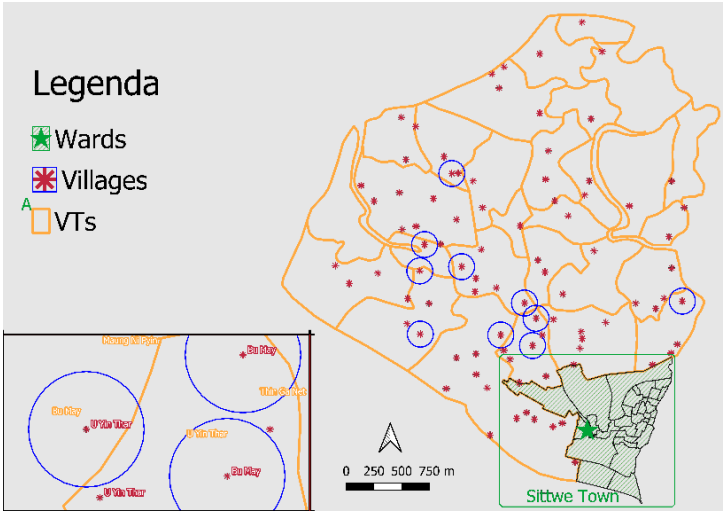

Fig. 10 – Buffer em torno das aldeias mal colocadas

Para que não surjam problemas futuros quando estes dados já estiverem inseridos na base de dados, é imperativo garantir a congruência entre as dimensões geométrica e alfanumérica. Atendendo à impossibilidade de determinar com precisão a localização real das aldeias em causa, optou-se por mudar na tabela de atributos o agrupamento de cada aldeia para aquele que de facto contem a geometria dessa aldeia. Assim, recorreu-se novamente a um processo python denominado "vil_correct", contido no Anexo B, com o seguinte algoritmo:

- Abrir em modo de escrita a geopackage que contem a tabela das aldeias e a dos agrupamentos
- Criar uma camada temporária a partir de um *spatial join* das duas tabelas para os objetos da tabela das aldeias que estejam geometricamente contidos na dos agrupamentos mas que apresentem valores diferentes para o campo 'VT', comum às duas tabelas, correspondente ao nome do agrupamento de aldeias atribuído. O resultado será uma tabela com 3 colunas – o código da aldeia, o código do agrupamento proveniente da tabela dos agrupamentos de aldeias, e o nome do agrupamento de aldeias também proveniente dessa mesma tabela – ficando os registos ordenados de forma crescente
- Abrir a camada das aldeias
- Selecionar apenas os registos dessa camada cujos valores Vill_Pcode – o identificador de cada aldeia - sejam idênticos aos da camada temporária.
- Para cada aldeia dessa seleção:
    - Obter da camada temporária a aldeia com o Vill_Pcode da camada atual
    - Mudar os atributos do agrupamento de aldeias e do Vill_Pcode da aldeia atual para os da aldeia selecionada anteriormente.

4.2.5. Anexação de campos com valores populacionais às camadas dos agrupamentos de aldeias e dos bairros

Como referido anteriormente, conseguiu-se um documento em formato Excel com os valores populacionais de cada agrupamento de aldeias e de cada bairro do Myanmar, mais concretamente o número de residentes por género, informação que, em conjunto com os dados populacionais em formato *raster* referidos anteriormente, serão crucias a este projeto. No entanto, dada a ausência de informação geoespacial nesse documento, decidiu-se adicionar esses dados à geopackage com as tabelas administrativas já existentes, pelo que se recorreu ao processo "acrescentar_valores_pop" - ver anexo C - para esse efeito. Na conceção deste algoritmo teve-se em conta o seguinte:

- Ambas as folhas deste ficheiro excel têm 11 colunas, sendo a coluna dos identificadores dos agrupamentos de aldeias e dos bairros a F.
- Atendendo ao número elevado de registos em cada folha, nomeadamente aos 3470 bairros e aos 13.590 agrupamentos de aldeias, previa-se que um algoritmo que se limitasse

a percorrer cada registo da folha de Excel até encontrar o respetivo agrupamento ou bairro seria à partida bastante ineficiente, a menos que por sorte, estivessem logo no topo de ambas as folhas do excel. Assim, optou-se por um algoritmo de pesquisa à partida mais eficiente do que o da pesquisa linear, nomeadamente o da pesquisa binária. Um algoritmo relativamente simples de implementar, este assenta numa lógica recursiva de divisão da lista de elementos original até se encontrar o elemento pretendido, estando essa lista já ordenada à partida, procedimento esse alcançado através do próprio excel (LeetCode).

O algoritmo seria o seguinte:

- Tomar o zero para índice inicial
- Se se tratar da primeira iteração:
  - Tomar o índice final como o último da lista
- Se o índice inicial for superior ao final:
  - O elemento procurado não se encontra na lista
  - Terminar essa chamada de função devolvendo *None*
- Se o elemento procurado for igual ao elemento no meio da porção da lista atual:
  - Encontrou-se o elemento pretendido
  - Terminar essa chamada de função devolvendo o índice do elemento
- Se o elemento procurado for superior ao elemento no meio da porção da lista atual:
  - Executar novamente o algoritmo mas mudando o índice inicial para o índice imediatamente a seguir ao do meio da porção atual
- Se o elemento procurado for inferior ao elemento no meio da porção da lista atual:
  - Executar novamente o algoritmo mas mudando o índice final para o índice imediatamente anterior ao do meio da porção atual

Assim, o algoritmo "acrescentar_valores_pop" seria:

- Abrir o ficheiro excel com os dados populacionais em modo de leitura
- Abrir a geopackage com os níveis administrativos em modo de escrita
- Para cada nível administrativo, bairros e agrupamentos de aldeias:
  - Guardar o nome do campo da camada administrativa que contém o respetivo identificador
  - Para cada coluna da lista (I, J e K) do excel, correspondentes aos valores populacionais totais, do sexo masculino e feminino, respetivamente:
    - Criar o respetivo campo e acrescenta-lo ao final da tabela administratriva
  - Criar um tuplo com os códigos desse nível administrativo da folha de excel
  - Para cada objeto da camada administrativa atual:
    - Obter através do algoritmo da pesquisa binária, o índice desse código no tuplo criado anteriormente

        - A partir desse índice, determinar a linha correspondente no Excel e preencher os campos anteriormente criados com os valores das colunas I, J e K

4.2.6. Inserção dos dados na base de dados

Uma vez corrigidos todos os dados anteriores, seria necessário inseri-los na base de dados criada para o projeto, criando uma tabela para cada conjunto de dados e dotando-a de um índice espacial que facilite a pesquisa destes objetos via comandos SQL. Assim, concebeu-se um processo python que de forma automatizada criasse essas mesmas tabelas a partir das camadas já existentes, preservando todos os campos e as respetivas estruturas de dados de cada um. O processo final, denominado "dataSource_to_dataBase", contido no Anexo D, pode ser descrito da seguinte forma:

- Abrir o ficheiro de texto com as credenciais de acesso ao PostgreSQL e guarda-las numa variável
- Com essas credenciais guardadas, aceder a uma base de dados já existente, como por exemplo a postgres que é criada aquando da instalação do PostgreSQL, e criar uma base de dados para o nosso projeto, de nome myanmar.
- Para cada conjunto de dados:
    - Abri-lo em modo de leitura
    - Criar uma lista vazia
    - Para cada campo da camada:
        - Adicionar à lista anterior um tuplo com o nome e o tipo SQL do campo
    - Criar uma tabela na base de dados a partir da lista anterior e da informação geométrica do campo da geometria da camada.
    - Criar um índice espacial para a tabela
    - Para cada objeto da camada de entrada:
        - Inseri-lo na tabela via expressão SQL

5. Considerações Finais

De forma geral, apesar da implementação do WebSIG ter ficado aquém do inicialmente proposto, o trabalho desenvolvido alcançou ainda assim alguns dos objetivos propostos, nomeadamente na extração, tratamento e apresentação da informação relevante para o projeto.

5.1. Conclusões do trabalho realizado

Começando pelo tratamento dos dados geoespaciais, considera-se que os procedimentos implementados em python não só servem adequadamente os respetivos propósitos para que foram concebidos, como são também generalizáveis a quaisquer outros conjuntos de dados vetoriais, permitindo alcançar os resultados desejados num espaço de tempo aceitável para uma linguagem de programação interpretada como é o caso do Python, atendendo a algoritmos como a pesquisa binária que tornam mais eficiente e célere uma parte importante do procedimento. Outro aspeto importante é o uso de pacotes python open source, como o caso do OGR que, por oposição ao ArcPy do ArcGIS, não restringem o acesso a algoritmos de geoprocessamento mediante o pagamento de licença do software base, fator importante para todo e qualquer trabalho futuro que venha a ser acrescentado a este projeto seja por que entidade for, com mais ou menos recursos financeiros ao seu dispor.

Assim, graças à informação obtida e devidamente tratada, pensa-se que este trabalho constitui um bom ponto de partida para um webSIG futuro de cariz humanitário para o Myanmar. Supondo que o Myanmar voltará a ser atingido este ano por cheias devastadoras na época das monções, que informação relevante poder-se-á tirar do webSIG proposto nesta dissertação? Analisando os dados de cheias obtidos para anos anteriores, tomou-se, a título exemplificativo, a informação obtida da UNOSAT datada de 14 de junho de 2016, poucos dias após a zona noroeste do Myanmar ter sido fortemente alagada, incluindo o concelho de Sittwe, a área de estudo deste projeto.

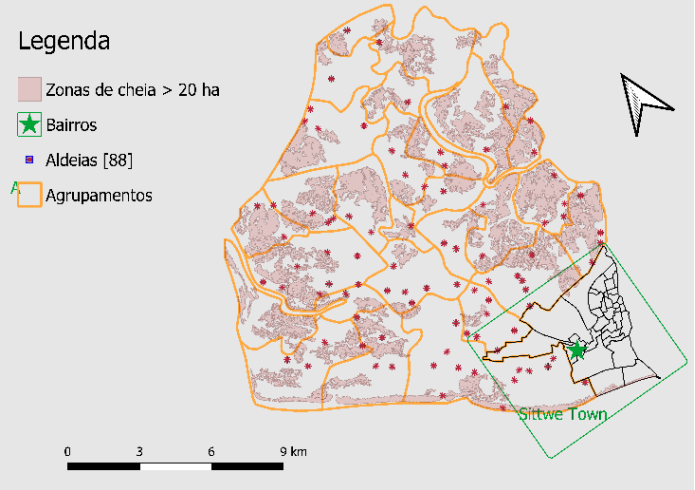

Fig. 11 – Cheias em Sittwe (14/6/2016)

Como se pode ver pelo mapa acima, há algumas aldeias muito próximas ou mesmo dentro de áreas afetadas pelas cheias, o que suscita a pergunta seguinte: Quantas pessoas terão sido afetadas?

Atendendo à dimensão humanitária da aplicação, dar-se-á prioridade aos refugiados alojados em campos de asilo em Sittwe, tentando-se perceber quais os campos mais próximos de zonas de inundação. Para esse efeito, selecionaram-se todos os campos a uma distância inferior a 0.001 graus (~100 metros) de qualquer área afetada por cheia via a seguinte expressão no QGIS:

Intersects(

buffer(

$geometry, distance:= 0.001, segments:= 5), aggregate(layer:='ST20160714', aggregate:='collect', expression:=$geometry, filter:="Area_hectars" > 20))

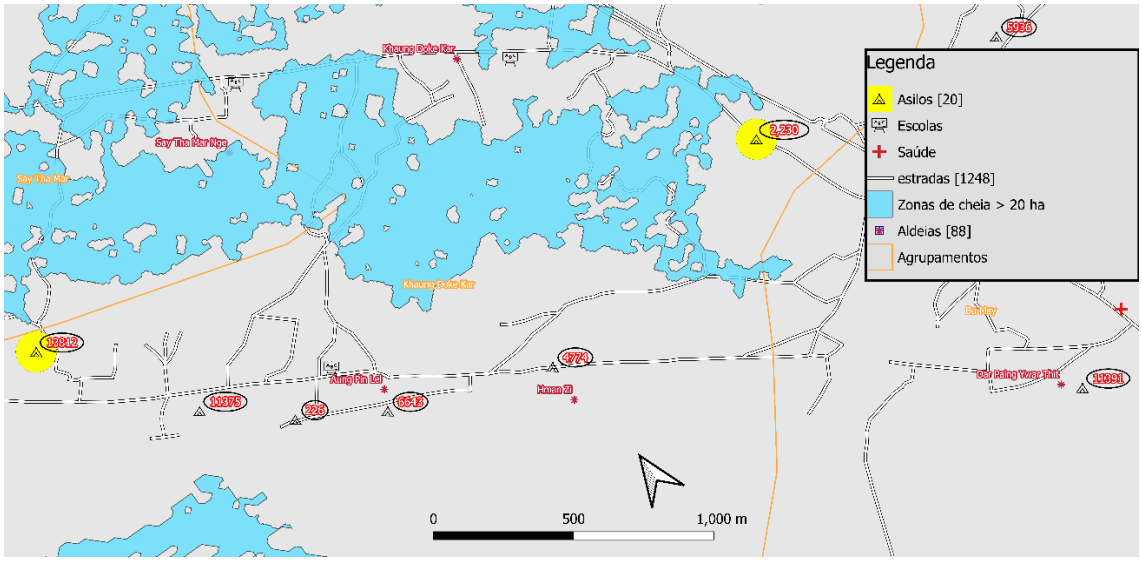

Fig. 12 – Asilos a menos de 100 metros de zonas de cheia a coloridos a amarelo

Conclui-se que dois dos vinte campos se encontram na situação em questão. Atendendo às 13812 pessoas presentes no asilo mais ocidental, dar-se-á prioridade aos seus residentes no que concerne ao seu encaminhamento para a instalação de saúde mais próxima, ou outra de grande dimensão onde possam ser distribuídos mantimentos como escolas. Assim, qual será a unidade de saúde mais próxima deste asilo por estrada atendendo a que alguns troços poderão estar inoperacionais devido a alagamento?

Para dar resposta a esta questão, começou por se usar o algoritmo do QGIS "Trajeto mais curto", que tal como o nome sugere gerou uma tabela com os troços da camada das estradas e as respetivas distâncias entre o campo anterior e cada instalação de saúde dessa camada. De seguida, para o resultado anterior, selecionaram-se a amarelo os troços que não intersetam nenhuma das zonas de inundação de área superior a vinte hectares com a seguinte expressão:

```
NOT intersects(
        $geometry, aggregate(
        layer:= 'ST20160714', aggregate:='collect', expression:=$geometry, filter:="Area_hectars" > 20))
```

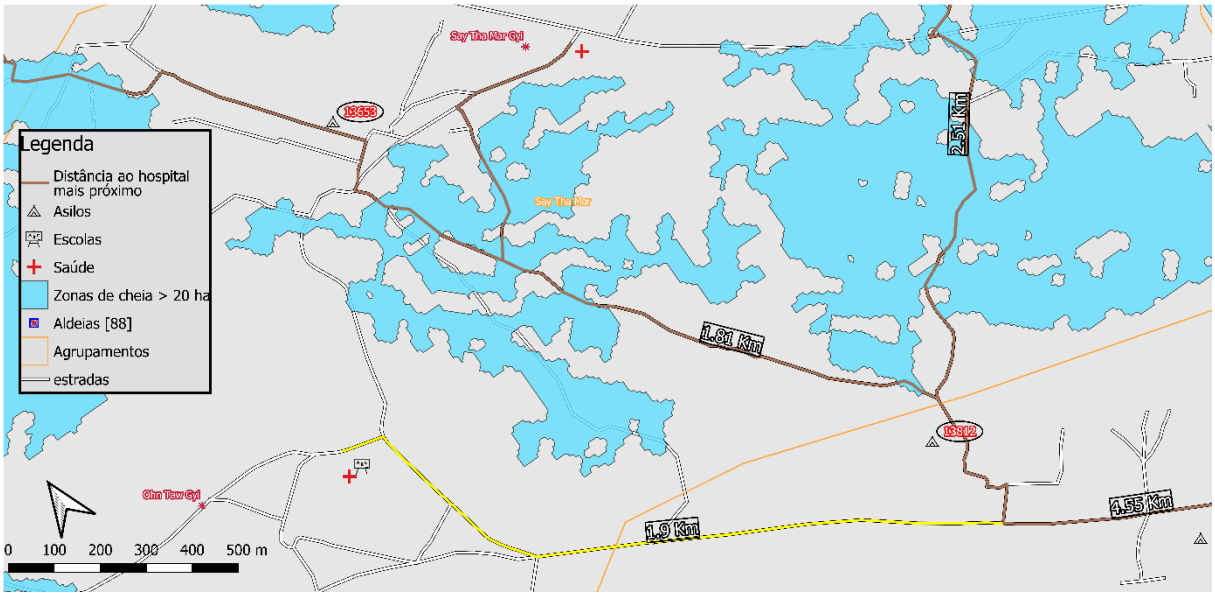

Fig. 13 – Distâncias aos hospitais mais próximos

Apesar do algoritmo anterior revelar que o hospital mais próximo estaria a 1.81 Km do campo em questão, esse caminho estaria inacessível devido às áreas alagadas circundantes, pelo que o segundo mais próximo seria o de 1.9 Km marcado a amarelo.

Este pequeno exemplo corrobora o impacto que um webSIG baseado no projeto desta dissertação poderia ter num contexto em que uma decisão errada pode custar a vida de muita gente.

5.2. Recomendações para trabalhos similares

O domínio dos Sistemas de Informação Geográfica engloba uma variedade de tópicos que, dada a sua componente tecnológica, se podem revelar bastante complexos. Assim, e dada a dimensão temporal para a execução de uma dissertação de mestrado, bem como o facto de esse trabalho ser levado a cabo apenas por uma pessoa, impõem-se algumas condições que não devem ser descuradas para que o mesmo possa ser entregue em tempo devido.

A primeira remete para a importância de um cronograma, uma ferramenta essencial a qualquer projeto com o intuito de que prazos estabelecidos sejam respeitados por todas as partes. Assim, é importante definir concretamente o âmbito do projeto de uma forma realista, procurando sempre satisfazer uma visão de *proof of concept* ao invés de uma megalómana que, apesar de válida, não se

adequa ao tempo disponível. É importante também não esquecer que a dimensão tecnológica dos SIG se irá traduzir em muitas horas de aprendizagem, tempo que terá de ser devidamente acautelado no cronograma geral.

Por fim, até a aprendizagem requer uma metodologia para que esta seja o mais eficiente possível. Uma parte importante dessa metodologia prende-se com as fontes de informação consultadas. Devido à mudança de paradigma que se verificou com o advento da Internet, em que o acesso à informação não se encontra mais barrado ao cidadão comum, torna-se importante filtrar essa informação de modo a se ir ao encontro do resultado pretendido. Assim, e para o efeito de tecnologias do mundo SIG como pacotes de Python ou visualizadores como o QGIS, sugere-se a consulta inicial dos respetivos manuais, muitas vezes encontrados nos sites oficiais destes programas, antes de uma pesquisa exaustiva no Google que, mesmo desaguando em domínios reputados como o StackOverflow ou o StackExchange, poderá conduzir o utilizador a um *rabbit hole* do qual não se obtém resultado algum.

5.3. Trabalho Futuro

Como referido anteriormente, esta dissertação constitui um bom ponto de partida para um WebSIG de cariz humanitário. No entanto, iterações futuras serão necessárias para se alcançar um protótipo de acesso universal via Internet, o qual exigirá um conhecimento sólido nas tecnologias de Backend e Frontend mencionadas no terceiro capítulo, como o Django ou o OpenLayers, respetivamente. Contudo, ir-se-á sugerir de seguida uma forma de simular a experiência do utilizador localmente através de software SIG já instalado via OSGEO, que poderá assim servir de inspiração a iterações futuras já com acesso à internet.

Primeiramente, sugere-se a conceção de algoritmos que complementem a informação de cheias e os dados populacionais já obtidos:

- Para a informação vetorial de cheias, poder-se-ia intersetar esta com o modelo digital do terreno já obtido, na expectativa de se desenvolver um modelo preditivo para as zonas de Sittwe mais propensas a alagamentos e os valores altimétricos associados.
- Quanto aos dados populacionais, e recordando-se o processo python que teve de ser implementado para corrigir as incongruências posicionais verificadas em várias das aldeias de Sittwe, sugere-se um algoritmo que intersete a camada dos agrupamentos de aldeias com a informação matricial populacional obtida, identificando-se assim, para cada agrupamento, as zonas de maior densidade populacional e portanto, um bom indicador

de que aí provavelmente se encontrarão aldeias desse agrupamento, podendo essa suposição ser posteriormente validada com imagens de satélite de alta resolução como as da Google, sendo também este último um ponto onde se poderia trabalhar no sentido da automatização.

Por fim, e de forma a possibilitar a utilizadores não versados em matéria SIG, sugere-se um protótipo implementado localmente, com recurso a *software* OSGeo em computadores com Windows. O QGIS apresenta-se como um excelente visualizador de informação geoespacial, pois permite não só a customização ao mais ínfimo detalhe de cada camada, como também a automatização da criação de mapas com esse mesmo grau de detalhe via respetivo pacote python, o PyQGIS, também presente no OSGEO. Este pacote apresenta também o acesso a um interface gráfico, permitindo assim a visualização de informação geográfica sem ter de se iniciar o QGIS propriamente dito. Posto isto, o modus operandi de um programa assim seria o seguinte:

- Ao iniciar o programa, o utilizador seria confrontado com o mapa do concelho de Sittwe, o mesmo que tem sido usado durante esta dissertação, no qual estariam visíveis as camadas dos agrupamentos de aldeias e dos bairros, sendo notificado via uma caixa de diálogo a clicar sobre um deles para obter mais informação.
- Após o *click*, o programa mudaria os limites do mapa visualizado para os do polígono selecionado anteriormente, apresentando uma caixa com as seguintes informações:
    - Número total de habitantes, e distribuições por sexo.
    - Número total de campos de asilo e respetivos números de agregados familiares e valores totais de alojados.
    - Área alagada do polígono escolhido.
    - Distância por estrada não alagada de todas as aldeias / asilos ao hospital e à escola mais próximos
- A informação acima seria obtida graças a *queries* executadas de forma instantânea ao nível da base de dados.

Bibliografia

ACLED. About ACLED. https://acleddata.com/about-acled/. Acedido em 11/11/2021.

Agafonkin, V. Leaflet - a JavaScript library for interactive maps. http://leafletjs.com/. Acedido em 13/11/2021.

Aljazeera. Myanmar's military coup prolongs misery for Rohingya in Rakhine. https://www.aljazeera.com/news/2022/1/6/rohingya-myanmar-restrictions-on-freedom-of-movement. Acedido em 28/01/2022

Australia National Map. https://nationalmap.gov.au/. Acedido em 12/11/2021.

Cameron, L., Demeyere, B., Henckaerts, J., La Haye, E., Niebergall-Lackner, H. (2015). The updated Commentary on the First Geneva Convention – a new tool for generating respect for international humanitarian law. International Review of the Red Cross, 97(900), 1209–1226.

Choudhury, N. (2014). World Wide Web and Its Journey from Web 1.0 to Web 4.0. International Journal of Computer Science and Information Technologies, Vol. 5 (6).

Congalton, R., Gu, J., Yadav, K., Thenkabail, P., Ozdogan, M. (2014). Global Land Cover Mapping: A Review and Uncertainty Analysis. Remote Sensing, 6(12), 12070–12093.

Daoudi, A., ElBoussaidi, G., Moha, N., Kpodjedo, S. (2019). [ACM Press the 34th ACM/SIGAPP Symposium - Limassol, Cyprus (2019.04.08-2019.04.12)] Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing - SAC '19 - An exploratory study of MVC-based architectural patterns in Android apps.

Django. GeoDjango Tutorial. https://docs.djangoproject.com/en/4.0/ref/contrib/gis/tutorial/#introduction Acedido em 18/12/2021.

ESRI. Shapefile file extensions. https://desktop.arcgis.com/en/arcmap/10.3/manage-data/shapefiles/shapefile-file-extensions.htm. Acedido em 28/01/2022

Fairhurst, R. (2020). Providing data to OpenStreetMap: a guide for local authorities and other data owners. https://blog.openstreetmap.org/wp-content/uploads/2020/07/Providing-data-to-OpenStreetMap.pdf. Acedido em 13/11/2021.

FAO. Assessment of the status of the development of the standards for the Terrestrial Essential Climate Variables. https://www.fao.org/3/i1237e/i1237e00.pdf Acedido em 30/01/2022.

FAO. FAO Geoportal. https://data.apps.fao.org/. Acedido em 16/12/2021.

Fee, E., Garofalo, M. (2010). Florence Nightingale and the Crimean War. American Journal of Public Health, 100(9), 1591–1591.

GeoServer. What is Geoserver? http://geoserver.org/about/ Acedido em 18/12/2021.

Gesesew, H., Berhane, K., Siraj, E. (2021) The impact of war on the health system of the Tigray region in Ethiopia: an assessment. BMJ Global Health.

Google Cloud Platform. Storing and Analyzing Your Data in Google's Cloud. https://cloud.google.com/files/articles/google-cloud_technical-article_overview-of-storage-options.pdf. Acedido em 18/12/2021.

Habib, M., Jubb, C., Ahmad, S., Rahman, M., Pallard, H. (2018). Forced migration of Rohingya : the untold experience. Ottawa, Ontario : Ontario International Development Agency, Canada

Harvard Map. Collaborating around location https://worldmap.maps.arcgis.com/home/index.html. Acedido em 11/11/2021.

Hess, R. (1998). A healing hegemony: Florence Nightingale, the British army in India and 'a want of … exercise'. The International Journal of the History of Sport, 15(3), 1–17.

Holovaty, A., Kaplan-Moss, J. (2008) The MTV Development Pattern. The Definitive Guide to Django: Web Development Done Right. Kinetic Publishing Services, LLC.

Human Rights Watch. An Open Prison without End: Myanmar's Mass Detention of Rohingya in Rakhine State. https://www.hrw.org/sites/default/files/media_2020/09/myanmar1020_web.pdf. Acedido em 29/01/2022.

JSON. Introducing JSON. https://www.json.org/json-en.html. Acedido em 15/12/2021

Kipgen, N. (2014). Addressing the Rohingya Problem. Journal of Asian and African Studies, 49(2), 234–247.

Laituri, M. e Kodrich, K. (2008). On Line Disaster Response Community: People as Sensors of High Magnitude Disasters Using Internet GIS. Sensors, 8(5), pp. 3037–3055

LeetCode. 704 Binary Search. https://leetcode.com/problems/binary-search/. Acedido em 25/01/2022

Li, S. e Gong, J. (2008). MASHUP: A NEW WAY OF PROVIDING WEB MAPPING/GIS SERVICES. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVII, Part B4.

Linos, G., Eirini, D., Panayiotis, T., Dimitris, M. (2019). Pythia: Identifying Dangerous Data-flows in Django based Applications. In 12th European Workshop on Systems Security (EuroSec '19), March 25–28, 2019, Dresden, Germany. ACM, New York, NY, USA.

Manuaba, I., Rudiastini, E. (2018). API REST Web service and backend system Of Lecturer's Assessment Information System on Politeknik Negeri Bali. Journal of Physics: Conference Series, 953.

Meng, M., Steinhardt, S., Schubert, A. (2017). Application Programming Interface Documentation. Journal of Technical Writing and Communication.

MIMU. About the MIMU. http://themimu.info/about-us. Acedido em 29/01/2022.

MIMU. Myanmar Administrative Structure: As per 2008 Constitution. https://themimu.info/sites/themimu.info/files/documents/Administrative_Structure_2008Constitution_20Mar2020.pdf. Acedido em 28/01/2022

Nash, E., Korduan, P., Abele, S., Hobona, G. (2008). Design Requirements for an AJAX and Web-Service Based Generic Internet GIS Client. 11th AGILE International Conference on Geographic Information Science(Italic). University of Girona, Spain

Neumann, A. (2017). Web Mapping and Web Cartography. Encyclopedia of GIS.

Open Geospatial Consortium Inc. (2006). OpenGIS® Web Map Server Implementation Specification.

Open Geospatial Consortium Inc. (2010). OpenGIS® Web Map Tile Service Implementation Standard.

OSGeo. About OSGeo4W. https://www.osgeo.org/projects/osgeo4w/. Acedido em 17/12/2021.

Pandas. About Pandas. https://pandas.pydata.org/about/. Acedido em 17/12/2021.

PostgreSQL. About PostgreSQL. https://www.postgresql.org/about/. Acedido em 17/12/2021.

Python Package Index. Requests 2.26.0. https://pypi.org/project/requests/. Acedido em 15/12/2021.

Sen Roy, N., Surinder K. (2000). Climatology of monsoon rains of Myanmar (Burma). International Journal of Climatology, 20(8), 913–928.

ShareTheMeal. How is my donation used? https://sharethemeal.org/en/faq.html Acedido em 30/01/2022.

Tan, X., Zhou, M., Zuo, X. e Cui, Y. (2008). [IEEE 2008 First International Conference on Intelligent Networks and Intelligent Systems (ICINIS) - Wuhan, Hubei, China (2008.11.1-2008.11.3)] 2008 First International Conference on Intelligent Networks and Intelligent Systems - Integration WebGIS with AJAX and XML Based on Google Maps. pp. 376–379.

The Irrawaddy. Nearly 9,500 People Affected by Rakhine State Floods. https://www.irrawaddy.com/news/burma/nearly-9500-people-affected-rakhine-state-floods.html. Acedido em 28/01/2022

UNHCR. Rohingya emergency. https://www.unhcr.org/rohingya-emergency.html. Acedido em 30/01/2022.

USGS. SRTM Data Products. https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-shuttle-radar-topography-mission-srtm-non. Acedido em 16/12/2021.

World Bank. Developer Information. https://datahelpdesk.worldbank.org/knowledgebase/topics/125589-developer-information. Acedido em 14/12/2021

World Bank. Indicators. https://data.worldbank.org/indicator/. Acedido em 14/12/2021

WorldPop. Top-down estimation modelling: Constrained vs Unconstrained. https://www.worldpop.org/methods/top_down_constrained_vs_unconstrained. Acedido em 16/12/2021.

Anexos

Anexo A : Procedimento Python "filterAndConvert2Geopackage"

Anexo B: Procedimento Python "vil_correct"

Anexo C: Procedimento Python "acrescentar_valores_pop"

Anexo D: Procedimento Python "dataSource_to_dataBase"

Anexo A: Procedimento Python "filterAndConvert2Geopackage"

```python
# - * - coding: utf-8 - * -
from osgeo import ogr, osr
import os, re, csv


def filterAndConvert2Geopackage(directory, filtrar):
    for x,y,z in os.walk(directory): #x,y,z correspondem ao diretório atual, lista de pastas dentro do diretório e lista de ficheiros dentro do diretório, respetivamente.
        for file in z:
            extensao = re.compile(r'\.([^\\]+)$').search(file).group(1).lower()
            if extensao == 'shp' or extensao == 'csv':
                info = re.compile(r'([^\\]+)\\([^\\]+)$').findall(x) #Lista com um tuplo (pasta, ficheiro a ser analisado)
                print(f'A analisar o ficheiro {info[0][1]} da pasta {info[0][0]}')
                #Abrir a camada de entrada
                in_source = fr'{x}\{file}'
                if extensao == 'shp':
                    in_driver = ogr.GetDriverByName('ESRI Shapefile')
                    in_dataSource = in_driver.Open(in_source, 0) #0 para abrir sem escrever
                    in_Layer = in_dataSource.GetLayer()
                    crs = in_Layer.GetSpatialRef()
                    in_LayerDefinition = in_Layer.GetLayerDefn()
                    in_Layer_geomType = ogr.GeometryTypeToName(in_Layer.GetGeomType())
                    #String com tipo de geometria. Ex: Multi Point
                #Se a extensão for CSV
                else:
                    with open(in_source, 'r', encoding='utf-8') as input_csv:
                        reader = csv.DictReader(input_csv, delimiter=',')
                        registos = tuple(reader) #Necessário guardar-se os registos numa variável pois após fechar o csv, já não se terá acesso aos registos a partir da variável reader. Cada elemento do tuplo será um dicionário para cada registo.
                        fieldnames = reader.fieldnames #No entanto tem-se acesso aos nomes dos campos a partir da variável fieldnames, mesmo depois de fechado o CSV
                    crs = osr.SpatialReference()
                    crs.ImportFromEPSG(4326)
                #Criar a camada de saída
                out_source = fr'{x}\{file[:file.index(extensao)]}gpkg'
                out_driver = ogr.GetDriverByName('GPKG')
                out_dataSource = out_driver.CreateDataSource(out_source)
                out_Layer = out_dataSource.CreateLayer(name=file[:file.index(f'.{extensao}')], srs=crs, geom_type=ogr.wkbUnknown)
                out_LayerDefinition = out_Layer.GetLayerDefn()

                #Iniciar a edição da GeoPackage:
                out_Layer.StartTransaction()

                #Pretende-se filtrar
                if len(filtrar) == 2:

                    #Abrir a camada com o objeto de estudo
                    clip_source = filtrar[0]
                    clip_driver = ogr.GetDriverByName('GPKG')
                    clip_dataSource = clip_driver.Open(clip_source, 0)
                    clip_Layer = clip_dataSource.GetLayer()
                    clip_Layer.SetAttributeFilter(filtrar[1]) #A funçao Clip aceita uma camada e não um objeto como parâmetro de entrada, daí a importância deste filtro de atributos
                    clip_feature = clip_Layer.GetNextFeature() #O método GetFeature() não iria respeitar o filtro anterior, daí se usar o GetNextFeature() para se iterar sobre uma seleçao de objetos
                    clip_feature_geom = clip_feature.GetGeometryRef()

                    #No caso de CSV ou geometria de ponto, terá de se verificar quais os objetos de entrada contidos no de estudo
```

```
53                  if extensao == 'csv' or in_Layer_geomType == 'Point' or
                    in_Layer_geomType == 'Multi Point':
54                      #Para shapefiles
55                      if extensao == 'shp':
56                          #Criar campos de atributos
57                          for i in range(in_LayerDefinition.GetFieldCount()):
58                              out_Layer.CreateField(in_LayerDefinition.GetFieldDefn(i))
59                          #Criar os objetos da camada final a partir da de entrada
60                          for inFeat in in_Layer:
61                              if clip_feature_geom.Contains(inFeat.GetGeometryRef())
                                == False:
62                                  continue
63                              else:
64                                  outFeature = ogr.Feature(out_LayerDefinition)
65                                  for i in range(out_LayerDefinition.GetFieldCount()):
66                                      fieldDefn = out_LayerDefinition.GetFieldDefn(i)
67                                      fieldName = fieldDefn.GetName()
68                                      outFeature.SetField(fieldName, inFeat.GetField(
                                        fieldName))
69                                  outFeature.SetGeometry(inFeat.GetGeometryRef())
70                                  out_Layer.CreateFeature(outFeature)
71                                  outFeature = None
72
73                      #Para CSVs
74                      else:
75                          #Criar campos dos atributos. Necessário determinar o tipo
                            de campo, acedendo ao valor de cada um no primeiro registo
                            da tabela para esse efeito.
76                          fieldnames_dic = {} #Dicionário com os nomes dos campo e os
                            respetivos tipos. Importante pois todos os valores de um
                            CSV são strings mas nem todos os campos serão desse tipo.
77                          for field in fieldnames:
78                              if re.compile(r'\d+\.\d+').search(registos[0][field]) !=
                                 None:
79                                  campo = ogr.FieldDefn(field, ogr.OFTReal) #Campo de
                                    números decimais
80                                  campo.SetPrecision(7)
81                                  out_Layer.CreateField(campo)
82                                  fieldnames_dic[field] = 'double'
83                              elif registos[0][field].isdecimal() == True:
84                                  campo = ogr.FieldDefn(field, ogr.OFTInteger64)
                                    #Campo de números inteiros
85                                  out_Layer.CreateField(campo)
86                                  fieldnames_dic[field] = 'int'
87                              else:
88                                  campo = ogr.FieldDefn(field, ogr.OFTString) #Campo
                                    de strings
89                                  campo.SetWidth(150) #Na eventualidade de haver
                                    strings muito longas, define-se um limite amplo
90                                  out_Layer.CreateField(campo)
91                                  fieldnames_dic[field] = 'string'
92                          #Criar os objetos da camada final a partir da de entrada
93                          for row in registos:
94                              ponto = ogr.CreateGeometryFromWkt(f"POINT ({float(row[
                                'Longitude'])} {float(row['Latitude'])})")
95                              if clip_feature_geom.Contains(ponto) == False:
96                                  continue
97                              else:
98                                  outFeature = ogr.Feature(out_LayerDefinition)
99                                  for field in fieldnames:
100                                     if fieldnames_dic[field] == 'double':
101                                         outFeature.SetField(field, float(row[field]))
102                                     elif fieldnames_dic[field] == 'int':
103                                         outFeature.SetField(field, int(row[field]))
104                                     else:
105                                         outFeature.SetField(field, row[field])
106                                 outFeature.SetGeometry(ponto)
```

```
107                                     out_Layer.CreateFeature(outFeature)
108                                     outFeature = None
109                         #No caso de geometria de polígono ou de linha, recorre-se a um clip.
110                         #Com o clip, a layer de saída herdará automaticamente os campos da
                            de entrada.
111                         else:
112                             in_Layer.Clip(clip_Layer, out_Layer)
113
114                         clip_dataSource = None
115
116                     #Sem filtrar.
117                     else:
118                         #Criar os campos a partir dos da shapefile de entrada
119                         for i in range(in_LayerDefinition.GetFieldCount()):
120                             out_Layer.CreateField(in_LayerDefinition.GetFieldDefn(i))

121                         #Criar os objetos da camada final a partir da de entrada
122                         for inFeat in in_Layer:
123                             outFeature = ogr.Feature(out_LayerDefinition)
124                             for i in range(out_LayerDefinition.GetFieldCount()):
125                                 fieldDefn = out_LayerDefinition.GetFieldDefn(i)
126                                 fieldName = fieldDefn.GetName()
127                                 outFeature.SetField(fieldName, inFeat.GetField(fieldName))
128                             outFeature.SetGeometry(inFeat.GetGeometryRef())
129                             out_Layer.CreateFeature(outFeature)
130                             outFeature = None
131
132                     #Terminar a edição da Geopackage e desligar as fontes de dados
133                     out_Layer.CommitTransaction()
134                     in_dataSource = None
135                     out_dataSource = None
136             #Apagar todos os ficheiros da pasta. Não apagará as GeoPackages criadas pois a
                lista de ficheiros não foi atualizada.
137             for i in z:
138                 os.unlink(fr'{x}\{i}')
139     #************************************************************
```

Anexo B: Procedimento Python "vil_correct"

```python
# - * - coding: utf-8 - * -
from osgeo import ogr

def vil_correct(in_source):
    in_driver = ogr.GetDriverByName('GPKG')
    in_dataSource = in_driver.Open(in_source, 1) #Abrir para editar a base de dados
    #Criar uma camada temporária com as aldeias fora dos agrupamentos lhes atribuídos na tabela de atributos
    misplaced_vils_layer = in_dataSource.ExecuteSQL(  #Spatial Join
    '''SELECT v.Vill_Pcode, vt.VT_PCODE, vt.VT
    FROM Villages AS v
    JOIN VTs as vt
    ON ST_Contains(vt.geom, v.geom)
    WHERE v.VT != vt.VT
    ORDER BY v.Vill_Pcode ASC;''')
    nr_fields = misplaced_vils_layer.GetLayerDefn().GetFieldCount()
    villages_layer = in_dataSource.GetLayerByName('Villages')
    villages_layer.StartTransaction()
    #Filtrar a camada das Aldeias pelos códigos das Aldeias mal colocadas
    villages_layer.SetAttributeFilter(f'Vill_Pcode IN ({",".join(tuple(str(feat.GetField(0)) for feat in misplaced_vils_layer))})')
    misplaced_vils_layer.ResetReading() #Importante fazer-se na eventualidade de se iterar sobre qualquer camada mais do que uma vez
    #Percorrer a seleçao anterior da camada das Aldeias e alterar para cada uma os seus atributos de de VT_Pcode e VT
    for village in villages_layer:
        village_code = village.GetField('Vill_Pcode')
        misplaced_vils_layer.SetAttributeFilter(f"Vill_Pcode = '{village_code}'")
        #Devem colocar-se plicas simples e não duplas em torno do atributo. Plicas duplas só para campos
        ms_village = misplaced_vils_layer.GetNextFeature() #Aldeia da tabela anterior
        village.SetField('VT_Pcode', ms_village.GetField('VT_PCODE'))
        village.SetField('VT', ms_village.GetField('VT'))
        villages_layer.SetFeature(village) #Reescreve os novos valores.
        misplaced_vils_layer.ResetReading()
    villages_layer.ResetReading()
    villages_layer.CommitTransaction()
    in_dataSource.ReleaseResultSet(misplaced_vils_layer) #Destruir a layer temporária criada
    in_dataSource = None

#*****************************************************************************
```

Anexo C: Procedimento Python "acrescentar_valores_pop"

```python
# - * - coding: utf-8 - * -
from osgeo import ogr
import openpyxl

def binarySearch (target, list_to_search, i_start = 0, i_end = 0, first_iter = True): # Pressupõe que a lista de entrada já está ordenada!!!
    if first_iter == True:
        i_end = len(list_to_search) - 1
    i_middle = round((i_end + i_start)/2)
    if i_start > i_end: #Número procurado não está na lista. Devolver None
        return None
    elif target == list_to_search[i_middle]:
        return i_middle
    elif target > list_to_search[i_middle]:
        return binarySearch(target, list_to_search, i_middle + 1, i_end, False)
    else:
        return binarySearch(target, list_to_search, i_start, i_middle - 1, False)
#*****************************************************************************

def acrescentar_valores_pop(excel_file, in_source):
    #Abrir o ficheiro Excel
    wb = openpyxl.load_workbook(excel_file, read_only=True) #Abrir apenas para ler pois já se sabe que não se vai editar o ficheiro, logo abertura é mais rápida
    folhas = wb.sheetnames
    #Abrir a Geopackage que contem as tabelas dos agrupamentos e dos bairros
    in_driver = ogr.GetDriverByName('GPKG')
    in_dataSource = in_driver.Open(in_source, 1)
    for l, layer_name in enumerate(('VTs', 'Wards')):
        #Definir o campo da camada que contem o código do agrupamento/bairro consoante a camada analisada
        if layer_name == 'VTs':
            campo_do_codigo = 'VT_PCODE'
        else:
            campo_do_codigo = 'Ward_Pcode'
        layer = in_dataSource.GetLayerByName(layer_name)
        layer.StartTransaction()
        campos_dic = {}
        for col in ('I', 'J', 'K'): #Colunas do Excel com os valores de população total, sexo masculino e feminino, respetivamente
            #Criar o respetivo campo na camada:
            campos_dic[col] = wb[folhas[l]][f'{col}1'].value
            layer.CreateField(ogr.FieldDefn(campos_dic[col], ogr.OFTInteger))
        excel_codes = tuple(i[0].value for i in wb[folhas[l]]['F2':f'F{wb[folhas[l]].max_row}'])
        #Preencher todos os registos da camada nesse campo
        for r in range(layer.GetFeatureCount()):
            #Encontrar a linha equivalente no Excel.
            #Coluna F será a que econtém os códigos em ambas as folhas do Excel
            #Devido ao número considerável de agrupamentos, usa-se o print seguinte em cada iteração apenas para descansar o utilizador quanto ao avanço do processo.
            print(f"A analisar o registo {r+1} de {layer.GetFeatureCount()}")
            registo = layer.GetNextFeature()
            codigo = registo.GetField(campo_do_codigo)
            row_nr = binarySearch(codigo, excel_codes) + 1 + 1 #+1 pois o slice do tuplo dos códigos começa na segunda linha, devido aos nomes dos campos ocuparem a primeira, Segudo +1 pois as linhas de Excel estão indexadas a 1 e não a 0.
            for col in ('I', 'J', 'K'):
                registo.SetField(campos_dic[col], wb[folhas[l]][f'{col}{row_nr}'].value)
            layer.SetFeature(registo)
            registo = None
        layer.ResetReading()
        layer.CommitTransaction()
    wb.close()
    in_dataSource = None
#*****************************************************************************
```

Anexo D: Procedimento Python "dataSource_to_dataBase"

```python
# - * - coding: utf-8 - * -
import psycopg2, os, re
from osgeo import ogr, osr
from psycopg2.sql import SQL, Identifier
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT



def dataSource_to_dataBase(country_directory):

    #Nome da futura base de dados
    dbname = re.compile(r'([^\\]*)$').search(country_directory).group(1).lower()
    #Guardar os parâmetros de ligaçao à base de dados que, por questões de segurança, não devem ser revelados no meio do código
    with open(r'C:\Users\Eu\UC\postgis-workshop-2018\nyc_connect_params.txt', 'r') as txt:
        connect_params = txt.read()

    #Para criar uma nova base de dados é necessária a conexão a uma preexistente.
    connection = psycopg2.connect(f'dbname=postgres {connect_params}')
    connection.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT) #Comando necessário para a criação de uma nova base de dados pois é necessário que mais nenhuma transaçao está a ocorrer em simultâneo. Desta forma a transação é logo efetivada, sem hipótese de ser revertida.
    cur = connection.cursor()
    cur.execute(SQL('CREATE DATABASE {};').format(Identifier(dbname))) #Usar os '%s' só para inserir valores. Os nomes de tabelas e de bases de dados têm de ser envoltos em instâncias da classe Identifier
    cur.close()
    connection.close()

    #Conectar à nova base de dados. As transações voltam a ser reversíveis até o commit final
    connection = psycopg2.connect(f'dbname={dbname} {connect_params}')
    cur = connection.cursor()
    cur.execute("CREATE EXTENSION postgis;") #Dotar a nova base de dados de capacidades de armazenamento de dados geoespacias
    extensions_dict = {'shp': 'ESRI Shapefile', 'gpkg': 'GPKG'}
    fieldTypes_dict = {'Integer': 'INT', 'Integer64': 'BIGINT', 'Real': 'REAL', 'String': 'TEXT'}#Dicionário com os nomes das estruturas de dados em sintaxe SQL
    #Criar uma tabela para cada conjunto de dados, adicionando os respetivos campos e objetos.
    for x,y,z in os.walk(country_directory):
        if len(z) == 0:
            continue
        table_name = re.compile(r'([^\\]*)$').search(x).group(1).lower()
        for i in z:
            if i.endswith('shp') or i.endswith('gpkg'):
                file = fr'{x}\{i}'
                extension = i[i.index('.')+1:]
                driver = ogr.GetDriverByName(extensions_dict[extension])
                dataSource = driver.Open(file, 0)
                layer = dataSource.GetLayer()
                srid = layer.GetSpatialRef().GetAttrValue('AUTHORITY', 1) #Devolve a string do código EPSG
                layerDefn = layer.GetLayerDefn()
                #Dar os campos da camada de entrada à tabela
                fields_tuples_list = []
                for i in range(layerDefn.GetFieldCount()):
                    fieldDefn = layerDefn.GetFieldDefn(i)
                    fieldName = fieldDefn.GetName()
                    fieldType = fieldDefn.GetFieldTypeName(fieldDefn.GetType())
                    fields_tuples_list.append((fieldName, fieldTypes_dict.get(fieldType, "VARCHAR"))) #Em caso de campo ser do tipo None, assume-se VARCHAR, que permite a inserçao de carateres alfanuméricos
                sql_fields_list = list(SQL("{} {}").format(Identifier(k[0]), SQL(k[1])))
```

```
                for k in fields_tuples_list) # Como mais à frente não é possivel ter
                p.ex.: "field_name VARCHAR", tem de se separar estas duas entidades,
                tendo em atnçao q os tipos de dados têm de ser instâncias da classe SQL
                e não Identifier. Importante para se prevenir a injeçao SQL.
54              geometria = SQL("{} {}").format(Identifier('geom'), SQL('GEOMETRY'))
55              query = SQL('CREATE TABLE {} ({}, {});').format(
56                  Identifier(table_name),
57                  SQL(', ').join(sql_fields_list),
58                  geometria)
59              cur.execute(query)
60              #Criar um índice espacial para a tabela. Bastante útil pois acelera a
                pesquisa de objetos cingindo-a a um extent reduzido.
61              create_sp_ix_query = SQL("CREATE INDEX {table_index} ON {tableName}
                USING GIST ({geom_field});").format(
62              table_index = Identifier(f'{table_name}_geom_idx'),
63              tableName=Identifier(table_name),
64              geom_field=Identifier('geom'))
65              cur.execute(create_sp_ix_query)
66              #Adicionar os objetos da camada de entrada à tabela
67              for feature in layer:
68                  wkt = feature.GetGeometryRef().ExportToWkt()
69                  query = SQL("INSERT INTO {table} ({fields}) VALUES ({percents},
                    ST_GeometryFromText(%s, %s));").format(
70                  table=Identifier(table_name),
71                  fields=SQL(', ').join(list(Identifier(i[0]) for i in
                    fields_tuples_list) + [Identifier('geom')]),
72                  percents=SQL(', ').join(list(SQL('%s') for j in range(len(
                    fields_tuples_list)))))
73                  query_values_list = list(feature.GetField(z[0]) for z in
                    fields_tuples_list) + [wkt, srid]
74                  cur.execute(query, tuple(query_values_list))
75              dataSource = None
76      #Tornar efetiva a transação.
77      connection.commit()
78      cur.close()
79      connection.close()
80  #*********************************************************************
```