Engenharia Web 19/20 – Arquitetura do projeto Sidewalk Monitoring System

José Pereira (A82880), Ricardo Petronilho (A81744)

3 de Maio de 2020

1 INTRODUÇÃO

Neste relatório apresenta-se a proposta de Arquitetura para o projeto Sidewalk Monitoring System, proposto pela equipa docente da unidade curricular de Engenharia Web.

Inicialmente será apresentada uma modelação dos dados/domínio do problema. De seguida a listagem dos micro serviços necessários para esta arquitetura. Por fim, apresentação/explicação da arquitetura proposta para o Sidewalk Monitoring System.

2 MODELAÇÃO DO PROBLEMA (DOMÍNIO)

Na fase anterior do projeto, implementação estrutural do projeto em **WebRatio**, o modelo lógico proposto, que reflete o modelo de domínio, foi o seguinte.

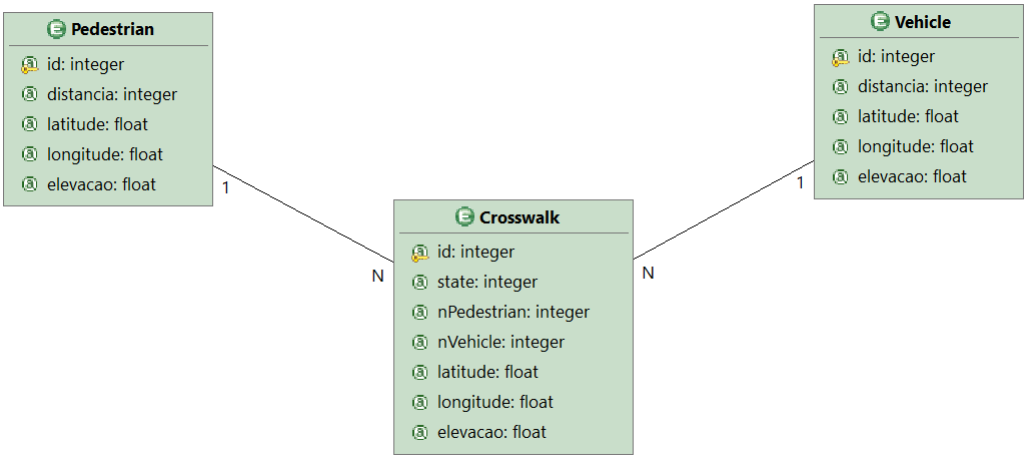

Figure 1. Modelo lógico inicial.

O modelo lógico faz sentido quando se trata de um sistema simplista que apenas interage com uma única base de dados. No entanto nesta segunda fase, na qual foi desenvolvida a arquitetura, o grupo apercebeu-se que tal não se verifica uma vez que o sistema está dividido em vários **micro-serviços, sendo que cada um deles necessita de uma base de dados privada (ou de tabelas privadas na mesma base de dados)** tornando assim impossível a utilização do modelo lógico proposto em cima.

Na próxima secção, juntamente com a descrição de cada micro-serviço será referido o modelo lógico **redefinido** que cada um utiliza.

3 MICRO SERVIÇOS

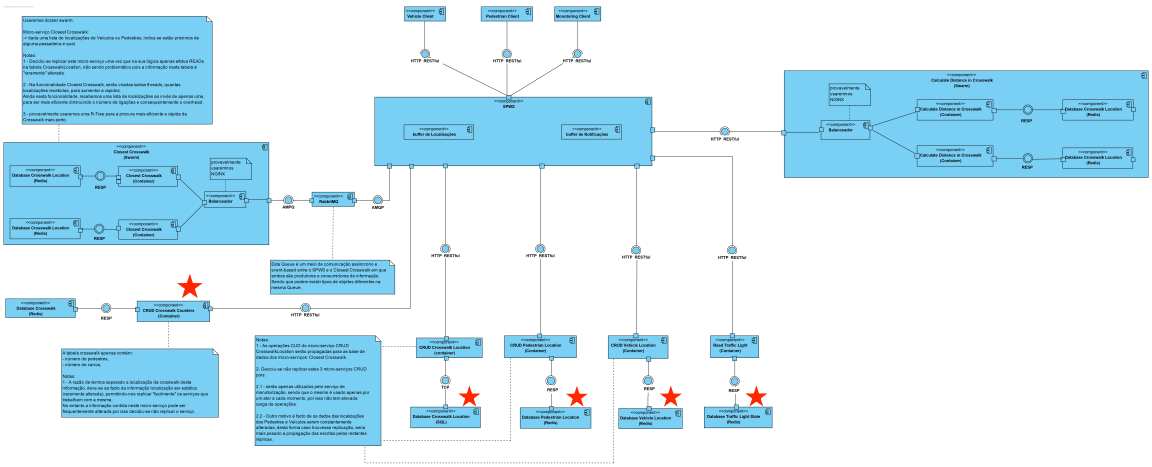

Figure 2. Diagrama de componentes.

Nesta secção são descritos os micro-serviços integrados na arquitetura, bem como as decisões que levaram à sua criação e estruturação dos mesmos.

Foram definidos os seguintes micro-serviços:

- Closest Crosswalk (redis, replicado)
- CRUD Crosswalk Counters (redis)
- CRUD Crosswalk Location (SQL)
- CRUD Pedestrian Location (redis)
- CRUD Vehicle Location (redis)
- Read traffic light (redis)
- Calculate distance in Crosswalk (redis, replicado)

3.1 Closest Crosswalk

O micro-serviço Closest Crosswalk é responsável por **indicar a passadeira mais próxima de uma dada localização**.

Visto que a lógica de procura da respetiva passadeira pode ser um processo exaustivo e com elevada complexidade o grupo identificou possíveis algoritmos eficientes sendo que o ideal seria o uso de R-Trees. No entanto existem outras alternativas mais simples (mas possivelmente menos eficientes) tais como a separação do total de passadeiras em N grupos e utilizar N Threads que exaustivamente efetuam a procura.

Concluiu-se que a **resposta em tempo útil** deste micro-serviço é fundamental para a notificação atempada tanto dos pedestres como dos veículos desta forma para aumentar a rapidez de processamento e **assegurar que com o aumento de carga o micro-serviço continua a ter tempos de resposta eficientes ou mesmo que não falha** decidiu-se utilizar um cluster usando a tecnologia Docker Swarm. Sendo o mesmo um conjunto de máquinas virtuais ou físicas que implementam um cluster em que uma máquina denominada por **master**, após pré-configurada, faz a gestão automática dos micro-serviços e respetivas réplicas, fornecidos pelas restantes máquinas denominadas por **slaves**. O número concreto de réplicas e máquinas a utilizar não está, ainda,

definido uma vez que é necessária a realização de **testes de carga** quando existir uma implementação prática do micro-serviço. Para a realização do **balanceamento de pedidos entre as réplicas** o grupo decidiu integrar a tecnologia NGINX no cluster.

Tal como referido na secção anterior, foi necessária a alteração do modelo lógico para a distribuição e replicação dos dados nos micro-serviços. O micro-serviço Closest Crosswalk necessita apenas da localização de todas as passadeiras, desta forma o modelo lógico utilizado apenas contém as coordenadas (latitude, longitude e altitude) das passadeiras. Concluiu-se que a localização de cada passadeira são dados, pela sua natureza, **alterados momentaneamente**, desta forma serão efetuadas, **maioritariamente, leituras** sobre os mesmos por isso a replicação destes dados torna-se facilitada. Sendo que **cada réplica do micro-serviço contém uma réplica dos dados** no seu armazenamento de dados privado. Caso exista alteração dos dados, mesmo que raramente, **a alteração é propagada para todas as réplicas**. A tecnologia a utilizar para o armazenamento de dados é o Redis uma vez que é rápida por conter os dados em RAM.

3.2 CRUD Crosswalk Counters

O micro-serviço CRUD Crosswalk Counters é responsável pelas funcionalidades **CRUD**(Create, Read, Update e Delete) da tabela Crosswalk referente aos contadores de *pedetrians/vehicles*.

Tal como já foi referido anteriormente, o modelo lógico proposto inicialmente foi afetado por forma a distribuir a lógica por micro serviços. Dessa forma, a tabela com a informação da Crosswalk presente neste micro-serviço, ao contrário do que se pode observar na figura 1, apenas contém os atributos **id** da crosswalk, e os contadores de *pedestrians* e *vehicles* diários, não contendo a localização, e estado da traffic light da mesma. A razão pela qual foi feita esta divisão de atributos deve-se ao facto da localização da crosswalk ser "raramente"alterada, no entanto, frequentemente **acedida**, ou seja, caso o read dessa informação ficasse neste micro-serviço iria "pesá-lo"ainda com mais pedidos. Do mesmo modo, a informação do estado da *traffic light* foi separada desta informação (a informação da localização e estado da traffic light da crosswalk, bem como os micro-serviços responsáveis por essas informações serão abordados mais aditante).

O armazenamento de dados neste micro-serviço efetua-se utilizando um Redis, devido à necessidade de rapidez nos tempos de resposta. No entanto, o Redis consiste num sistema *non-blocking*, isto é, escritas não bloqueiam leituras, não sendo crítico (ao contrário por exemplo das notificações dos clientes) para o sistema, pois caso o número de *pedestrians*/*vehicles* apresentado na monitorização não seja o mais atualizado, o mesmo será no próximo refresh da mesma.

3.3 CRUD Crosswalk Location

O micro-serviço CRUD Crosswalk Location é responsável pelas funcionalidades **CRUD**(Create, Read, Update e Delete) da tabela Crosswalk referente à localização da crosswalk.

Tal como já foi dito anteriormente, esta informação foi separada tanto da informação dos contadores, bem como da traffic light. A razão pela qual foi separada, deve-se à "raridade"(se pensarmos que se cria uma crosswalk a cada semana, isso será pouco para o sistema) de escritas efetuadas da localização das crosswalks, bem como do elevado número de leituras efetuadas da mesma. Assim evita-se um único micro serviço CRUD de toda a informação da crosswalk, o qual ficaria muito "pesado".

Outra responsabilidade deste serviço será de em cada alteração/adição de uma nova crosswalk (localização), propagar esses dados para todos os serviços onde exista a informação de localização das mesmas, tais como, **Closest Crosswalk** e **Calculate Distance in Crosswalk**.

A base de dados utilizada será **SQL**, visto que, se pretende persistir as localizações das crosswalks.

3.4 CRUD Pedestrian Location & CRUD Vehicle Location

A razão pela qual abordar-se-á estes dois micro serviços em conjunto, deve-se ao facto de serem bastante semelhantes, no entanto, para informações/dados diferentes, sendo respetivamente para a informação do **Pedestrian** e **Vehicle**.

Os micro-serviços **CRUD Pedestrian Location** & **CRUD Vehicle Location** são responsáveis pelas funcionalidades **CRUD** (Create, Read, Update e Delete) da informação (localização e id) respetivamente do **Pedestrian** e **Vehicle**.

Nestes micro serviços utilizou-se Redis, que apesar de ser *non-blocking*, ou seja, as escritas não bloqueiam as leituras, significa que dados podem ser lidos durante escritas tal como já foi referido anteriormente. Dado que, a localização dos **Pedestrians** e **Vehicles** consiste num dado crítico, ou seja, a consistência/integridade do seu valor torna-se importante, desta forma, pode-se pensar ser arriscado utilizar Redis, pois quando houvesse uma leitura de uma localização a mesma pode não ser a mais atualizada.

No entanto, apesar da localização ser um dado crítico, quando existe necessidade de notificar o **pedestrian** ou **vehicle**, efetua-se o mesmo no momento em que se identifica o "perigo". Isto é, após o micro serviço **Closest Crosswalk** referido anteriormente , identificar que uma das entidades encontra-se em "perigo", notifica-o nesse mesmo momento, e só de seguida escreve a informação, para que outros serviços ou funcionalidades a possam usar. Desta forma, a utilização de Redis neste dado crítico não influencia nessas situações.

3.5 Read Traffic Light

O micro-serviço Read Traffic Light é responsável por **indicar o estado do traffic light de uma determinada passadeira**.

Tal como referido na secção 3.3, inicialmente a informação do taffic light estava anexada num único serviço que continha, também, os contadores de pedestres e veículos. No entanto esse serviço seria **sobrecarregado uma vez que era utilizado constantemente** tanto para leituras na monotorização bem como para escritas quando se detetava a existência de pedestres ou veículos numa passadeira e ainda eram efetuadas leituras para implementar o requisito de notificar o estado do traffic light tanto aos pedestres como veículos.

Na implementação deste último requisito os atores devem ser notificados em tempo útil assim a separação em dois micro-serviços, sendo um responsável pelo CRUD dos contadores e outro apenas para leitura do traffic light torna-se necessária para que exista **balanceamento de carga, ficando a leitura do estado do traffic light mais disponível e por isso com tempo de resposta menor**.

Visto que o estado do traffic light, pela sua natureza, é constantemente alterado, torna-se complexa a replicação destes dados, no entanto para que a leitura seja mais rápida decidiu-se utilizar a tecnologia Redis.

3.6 Calculate distance in Crosswalk

O micro-serviço Calculate distance in Crosswalk é responsável por indicar a distância de um pedestre ou veículo a uma passadeira dado a localização dos atores e o identificador da respetiva passadeira.

Visto que este micro-serviço é apenas utilizado na monotorização, decidiu-se enviar apenas o identificador da passadeira e não a localização da mesma pois podem ser efetuados **muitos pedidos de cálculos de distância para a mesma passadeira (enquanto está a ser observada) sendo o identificador menos dados a ocupar na rede**.

Desta forma este micro-serviço necessita de armazenar a localização de todas as passadeiras, sendo o mesmo feito utilizado a tecnologia Redis.

Tal como na secção 3.1 concluiu-se que a localização de cada passadeira são dados, pela sua natureza, **alterados momentaneamente**, desta forma serão efetuadas, **maioritariamente, leituras** sobre os mesmos por isso a replicação destes dados torna-se facilitada. Sendo que **cada réplica do micro-serviço contém uma réplica dos dados** no seu armazenamento de dados privado. Caso exista alteração dos dados, mesmo que raramente, **a alteração é propagada para todas as réplicas**.

Analogamente a replicação do micro-serviço será realziada utilzando Docker Swarm.

4 SERVIÇO SPWS

A arquitetura implementada consiste numa *Orchestration*, onde o **SPWS** é o **Maestro**, gerindo os micro serviços referidos anteriormente.

O SPWS recebe os dados dos clientes (**Pedestrians** e **Vehicles**), isto é, as suas localizações, e começa por enviar as mesmas para o micro serviço **Closest Crosswalk**, no entanto, para uma *queue* que existe entre ambos, de forma a tornar esta funcionalidade **asíncrona e event-based**, isto é, o **SPWS** não fica à espera da resposta do mesmo.

Após o **Closest Crosswalk** determinar se as localizações recebidas estão no raio de crosswalks, envia a mesma para a queue.

O **SPWS** recebe as mensagens da queue, enviando imediatamente a notificação para os **Pedestrians** e **Vehicles**, sendo que neste último, ainda envia separadamente a notificação do estado da traffic light. O objetivo de enviar as notificações separadas consiste em não "prender"/fazer esperar os **vehicles** dessa mesma notificação, pois para os mesmo, é mais crucial verificar a presença de pedestres do que o estado da traffic light.

Usufruindo do facto do envio das localizações por parte dos atores são num intervalo de tempo curto, aproveita-se o *Http request*, para, caso haja, uma notificação, ser imediatamente enviada no response, sendo desta forma a comunicação assíncrona visto que não fica à espera.

Ao mesmo tempo que é efetuada a notificação das entidades é atualizada a localização das mesmas através dos micro-serviços CRUD Pedestrian/Vehicle Location pois esses dados são necessários na monotorização.

Concluindo, o SPWS fornece as seguintes funcionalidades:

- update location: envio da localização por parte dos clientes.
- verificação de perigo: dada localização das entidades, verificar se as mesmas estão no raio de uma crosswalk.
- notifiação: as entidades em caso de perigo são avisadas do mesmo (**Pedestrian** e *vehicle*.
- monitorização de uma crosswalk: consulta da localização/distância dos **Pedestrians** e **Vehicles** na vinhança.
- CRUD da informação das crosswalks (localização).