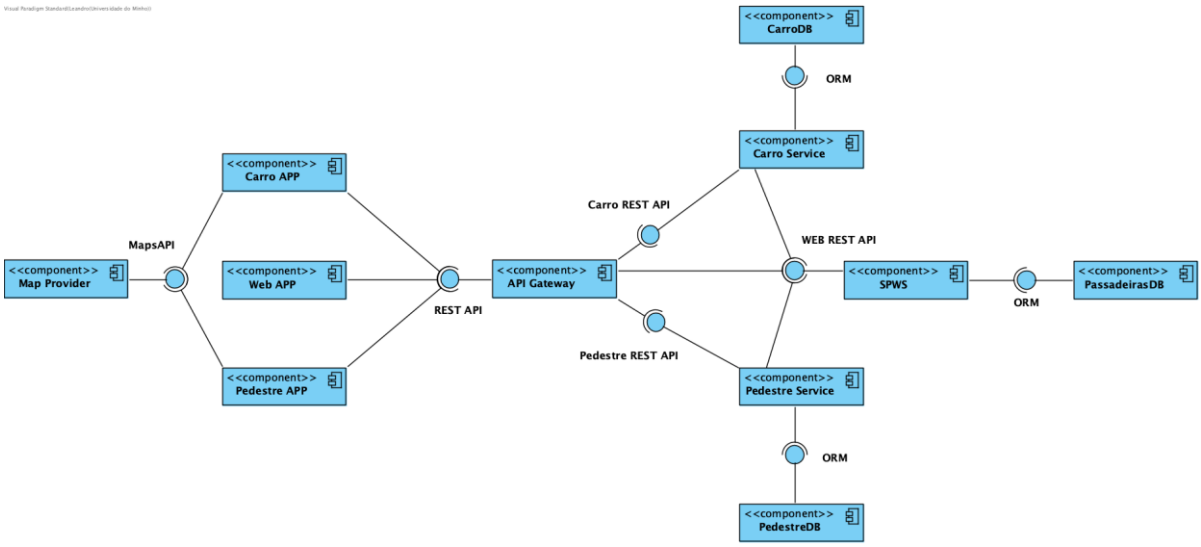
Leandro Costa pg41083 e Paulo Jorge pg17918

Proposta de arquitetura



Decidimos optar por uma arquitetura baseada em microserviços, ao invés de monolítica. Esta decisão teve como base o facto de o contexto do projeto poder ser sub-dividido em 3 principais componentes, cada um deles com requisitos próprios, cuja divisão poderá trazer benefícios. Apesar da criação de 3 serviços independentes entre si, estes podem estabelecer comunicação através de *endpoints* definidos, tratando-se assim de REST APIs.

A utilização de uma API *Gateway* como intermediário entre as aplicações e os serviços tem o intuito de se evitar possíveis API *chats* com diferentes serviços por parte do cliente para realizar uma tarefa. Através da utilização deste componente é possível criar novos *endpoints* personalizados que tratam de realizar os pedidos a diferentes serviços. Delegada a responsabilidade da gestão dos pedidos a este componente, deixa de existir o *delay* associado a múltiplos pedidos *http* do lado do cliente. Com apenas um pedido do lado do cliente podem ser invocados múltiplos serviços. Se fosse o browser (ou outra aplicação) a realizar múltiplos pedidos *http* a múltiplos serviços, poderia originar um API *chat* intensivo entre cliente e servidor, o que poderia vir a ser bastante demorado. Do lado da api *gateway*, a comunicação é bastante mais rápida uma vez que se trata de uma LAN *server side*.

De modo a usufruir ao máximo dos benefícios que uma arquitetura baseada em micro serviços oferece, foi decidida a utilização de uma base de dados para cada um dos serviços. Deste modo, evita-se conexões a uma mesma base de dados para obter informações completamente independentes.

No contexto específico deste trabalho, optou-se por criar um serviço dedicado à aplicação dos pedestres, outro para a aplicação de automóvel e outra para a aplicação web. O fluxo de acontecimentos idealizado passa por:

- a pedestre APP e carro APP possuem localização simulada alterada periodicamente e estas fazem um pedido GET inicial ao SPWS para obterem informação sobre a localização das passadeira presentes à volta de um raio (distância ainda indefinida)

- de modo a evitar pedidos desnecessários aos serviços, estes apenas comunicam a sua localização (POST ao pedestre e carro servisse) quando estão próximos da localização de uma passadeira.

- no caso do pedestre, quando é comunicada a localização (juntamente com o id da passadeira a que se encontra próximo), o pedestre *service* trata de fazer um pedido ao SPWS que irá incrementar o número de pedestres da passadeira em questão (*nPdestrians* na tabela das passadeiras).

- quando o pedestre deixa de estar próximo da passadeira, o pedestre *service* trata de realizar novamente um pedido ao SPWS para que este decremente o *nPedetrians* da passadeira correspondente.

- no caso do automóvel, da mesma forma que os pedestres, estes comunicam a sua localização apenas se se encontrar na proximidade de uma passadeira (cujas localizações foram obtidas a partir de um pedido GET ao SPWS). A comunicação da sua localização é feita simultaneamente ao carro service e SPWS. O pedido à API *gateway* trata de fazer um pedido POST ao carro *service* e SPWS. Desta forma evita-se que o cliente tenha que realizar dois pedidos, sendo este geridos pela API *gateway*. Ao receber o pedido, o SPWS verifica a coluna nPedestrians da passadeira que o veículo se encontra próximo e será enviada uma resposta diferente mediante este valor.

- quando o carro comunica a sua localização próxima de uma passadeira irá receber como resposta a informação sobre o estado da passadeira. Mediante o número de pedestres na passadeira em questão, este irá receber uma notificação com o estado de *Safe to cross* (nPedestrians = 0) ou *Pedestrian alert* (nPedestrians > 0).

Com o fluxo de acontecimentos definido, entendemos que as *business functions* a definir em cada um dos componentes são as seguintes:

Pedestre Service:

- *endpoint* para POST da localização do pedestre; - *endpoint* para obter informação sobre pedestre

Carro service:

- *endpoint* para POST da localização do veículo

- endpoint para obter informação sobre o veículo

SPWS:

- *endpoint* para comunicação da informação de novo pedestre nos arredores (resulta em aumento do número de pedestres)

- *endpoint* para a comunicação da informação de um veículo próximo de uma passadeira

- *endpoint* para obter informação sobre passadeiras

A comunicação da localização da posição por parte dos pedestres e veículos será de natureza *event based*, bem como o envio da notificação ao veículo no caso de um pedestre se encontrar próximo. As conexões à base de dados serão assíncronas de modo a não bloquear o *event loop* do *javascript* (irá ser utilizado Node js). O estilo arquitetural trata-se de natureza orquestral, uma vez que é feito um pedido a um certo serviço e estes, se necessitarem, irão desencadear comunicação com outros serviços.