Fase 3 - Requisitos, Casos de Uso e Arquitetura

Pedro Carrega, nº49480 Vasco Ferreira, nº49470
Ye Yang, nº 49521

16 de Abril de 2020

1 Motivação para Dataset e Serviços

O dataset ecommerce foi escolhido pelo grupo devido ao grande número de eventos gerados e consequente informação produzida durante a utilização de uma loja de ecommerce. Informação esta que pode ser utilizada de diversas formas através de um grande número de variados serviços. Essa mesma informação poderá ser utilizada em vários contextos, sendo que escolhemos os seguintes 5 serviços que demonstram diferentes tipos de informação sobre o dataset:

- api/products/listCategories: Fornece todas as diferentes categorias presentes nos dados
- api/products/popularBrands: Fornece a contagem de eventos associados a cada marca
- api/products/salesByBrand: Lista o numero de vendas de cada marca
- api/products/salePrice: Calcula o valor médio de venda de uma determinada marca
- api/events/ratio: Apresenta a distribuição relativa de cada tipo de evento, havendo os possíveis valores: view, cart e purchase

Os serviços foram escolhidos de forma a que consigam fazer diferentes tipos de operações sobre os dados, desde serviços mais específicos e por isso com menos carga na base de dados, a serviços mais abrangentes e consequente aumento de carga. Foram também escolhidos pois todos os serviços fornecem dados úteis para serem explorados no contexto de lojas de ecommerce.

2 Diagrama de Casos de Uso

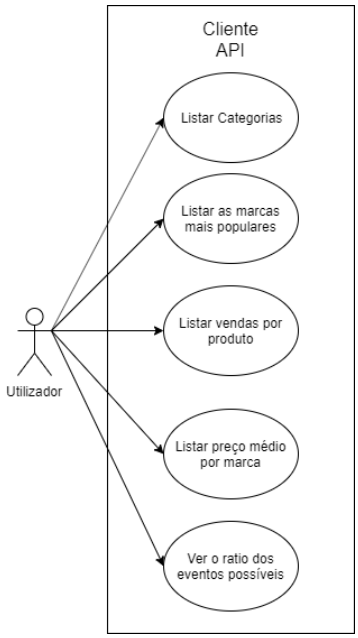

Figura 1: Diagrama de casos de uso

3 Requisitos

Requisitos Não Funcionais	**Descrição**
Portabilidade	Implementação de servidor em NodeJS e cliente em HTML de modo a facilitar o processo de mudança de plataforma do serviço
Legibilidade	A separação clara entre as camadas de apresentação, lógica de negócio e acesso à base de dados irá tornar o fluxo do sistema mais legível para os desenvolvedores
Estabilidade	A distribuição do sistema por diversas máquinas virtuais, que poderão estar distribuídas por diferentes fornecedores cloud e em diferentes Data Centers permitem obter um sistema estável
Elasticidade	O sistema deverá ser capaz de se adaptar à carga de trabalho através do provisionamento e desprovisionamento dos recursos de forma autónoma. Idealmente, de forma que em qualquer ponto do tempo, o sistema apenas utilize o número de máquinas necessárias de forma a corresponder à carga atual
Escalabilidade	Capacidade do sistema lidar com o crescimento de carga de trabalho. Pode-se associar à capacidade de elasticidade do sistema
Confiabilidade	Visto o sistema estar implementado na núvem, conseguimos garantir alta confiabilidade nos sistema, pois se um servidor no datacenter falhar, conseguimos facilmente migrar a VM para outro servidor funcional

Tabela 1: Requisitos Não Funcionais

Requisitos Funcionais	**Descrição**
Listar Categorias Disponíveis	Serviço que fornece aos clientes todas as categorias
Visualizar Popularidade das Marcas	Serviço que fornece cada marca associada com a sua popularidade
Visualizar Número de Vendas Individuais	Fornece o numero total de vendas de cada marca
Visualizar Preço Médio de Venda	Fornece o preço médio dos produtos vendidos de uma determinada marca
Visualizar Rácio de Tipo de Eventos	Fornece a percentagem de cada tipo de evento

Tabela 2: Requisitos Funcionais

4 Arquitetura da aplicação

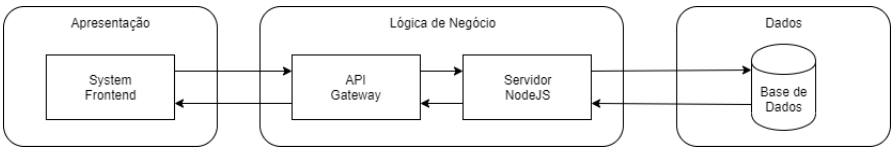

Figura 2: Arquitetura da aplicação

Após a definição da API com o Swagger na fase anterior, possibilitou a opção de exportar do API definido um Cliente (em Java) e um Servidor NodeJS.

4.1 System Frontend

O frontend irá dispor as funcionalidades anteriormente definidas. Pode ser extraído do ficheiro Swagger em formato HTML.

4.2 API Gateway

Este gateway irá ser encarregado de receber pedidos HTTP vindos do frontend e reencaminhá-los para o servidor NodeJS que tratará da lógica de negócio. Também irá encaminhar as respostas vindas do servidor para o frontend de modo disponibilizar os dados pretendidos.

4.3 Servidor NodeJS

O servidor exportado do Swagger irá receber os pedidos a partir da API Gateway, e processá-los de acordo com a funcionalidade pretendida. A lógica de negócio é aqui tratada, realizando as queries necessárias para a base de dados. Ao receber os dados, processa-os de acordo com a funcionalidade, e encaminha o resultado para a gateway.

4.4 Base de dados

A base de dados irá conter o conteúdo dos ficheiros .csv do nosso data set, que são acedidos pelo servidor de modo a poder efetuar as leituras necessárias para produzir uma resposta para o cliente.

5 Arquitetura técnica

Para garantir os requisitos não funcionais mencionados a cima devemos ter em atenção os serviços da cloud escolhidos, pois proporcionam vários aspetos fundamentais dos requisitos.

A utilização dos serviços disponibilizados pela AWS permite cumprir alguns dos requisitos não funcionais mencionados, nomeadamente a Escalabilidade e a Elasticidade. A implementação da API através da API Gateway permite que este serviço da cloud trate automaticamente da carga de pedidos HTTP recebida, não sendo necessário a criação de scripts especializados para distribuição de carga. A base de dados a ser implementada na Amazon DynamoDB também permite escalabilidade e também eficácia na resolução de queries, tratando da carga de queries automaticamente tal como os restantes serviços da cloud.

A Confiabilidade do sistema também é melhorada com a instalação em serviços na cloud, visto estas terem deteções automáticas de falhas de servidores, possibilitando a migração do sistema virtualizado para outro servidor funcional.

A Estabilidade é garantida nas várias ferramentas usadas na instalação do sistema. O API Gateway da AWS tratará automaticamente da carga de pedidos HTTP recebida, não sendo necessário a criação de scripts para mitigação de carga.

Já a Estabilidade é garantida devido a ter a Lógica de negócio distribuída por várias máquinas virtuais que cumprem a função de servidores.

Dado que o API é definido em Swagger, a implementação do cliente e servidor são facilmente traduzidos para uma outra linguagem exportável, garantindo a Portabilidade. Visto que a lógica de negócio vai ser implementada num servidor NodeJS, a maior parte do código desenvolvido poderá ser instalado noutra plataforma Cloud sem grandes mudanças, sendo as principais diferenças o acesso à base de dados da nova plataforma e a comunicação com a API Gateway. A camada de apresentação também não sofraria de mudanças de grau elevado, sendo que apenas a comunicação para a API Gateway mudaria.

6 Lançamento em Kubernetes

O scripts de deployment do sistema foram separados em 4 devido à necessidade dos clusters e node groups estarem ativos, antes de proceder aos próximos passos. Existe também uma secção de edição de ficheiro manual, o que fez a quebra entre o terceiro e o quarto script.

Antes da execução dos scripts é necessário verificar a existência dos seguintes repositórios, roles e policies e caso existam, precisam de ser **eliminados**:

1. Repositórios com nomes *products* e *events*
2. AWS Role com nome eksServiceRole
3. AWS Policy com nome ALBIngressControllerIAMPolicyEcommerce

Para a execução dos scripts são necessárias as seguintes ferramentas:

- AWS CLI
- eksctl
- kubectl

A região a escolher para o lançamento poderá ser qualquer um, porém recomendamos a região eu-west-1. Esta região terá de ser a mesma nos argumentos de todos os scripts que requeiram a mesma.

6.1 deploy1.sh

O primeiro script de deployment recebe 3 argumentos na seguinte ordem:

1. A região onde a Stack e o Cluster vão ser lançados (ex.: **eu-west-1**)
2. O nome da Stack que terá de ser único (nenhuma outra Stack na CloudFormation da conta pessoal poderá ter o mesmo nome) para o script funcionar corretamente (ex.:**ecommerce-stack**)

3. O nome do Cluster que também terá de ser único (ex.:**ecommerce-cluster**)

A criação da stack e do cluster irá demorar cerca de 10 a 20 minutos até ficarem ativos, após o qual poderemos proceder à execução do segundo script. O estado do script pode ser verificado com o seguinte comando:

- *aws eks describe-cluster –name CLUSTER_NAME* , mudando CLUSTER_NAME para o nome do cluster dado nos argumentos

A execução do segundo script só deve ser feita quando o estado do cluster estiver em **ACTIVE**.

6.2 deploy2.sh

O segundo script recebe os mesmos argumentos que o primeiro, todos na mesma ordem. Neste script vão ser criados os node groups e o pull das imagens dos serviços.

Para realizar o pull, irá ser pedido para inserir os credenciais IAM que dão acesso aos repositórios por nós criados. Estes credenciais encontram-se no ficheiro **credenciais.txt** juntamente com a região onde os repositórios se encontram.

No fim