Apresentar a forma como as questões identificadas anteriormente podem ser satisfeitas com o novo sistema, utilizando a linguagem de interrogação do sistema NoSQL

Do ponto de vista do cliente deve ser possível:

* (RE1) - Ver o histórico das suas viagens num dado período. (histórico\_viagens)

db.Clientes.find({$and: [{\_id:id\_input},{“bilhetes.data\_partida”: {$gte : ISODate(data\_p\_input)},{”bilhetes.data\_chegada”: {$lte: ISODate(data\_c\_input)}]},{“bilhetes.origem”:1,”bilhetes.partida”:1}).pretty();

* (RE2) - Consultar o montante gasto num dado período. (total\_gasto)

db.Clientes.aggregate( {$match: { $and: [{“bilhetes.data\_partida”: {$gte : ISODate(data\_p\_input)},{”bilhetes.data\_chegada”: {$lte: ISODate(data\_c\_input)}]}} , {$group: {total: {$sum: ”$billhetes.preco”}}},{$project: {total: 1}});

* (RE3) - Visualizar a origem, destino, duração, preço, lugar e data da viagem associada ao seu bilhete. (detalhes\_viagem)

db.Clientes.find({“bilhetes.id”: bilhete\_input},{“bilhetes.id”:1,”bilhetes.origem”:1,”bilhetes.destino”:1,”bilhetes.duração”:1, “bilhetes.preco”:1, “bilhetes.lugar”:1, “bilhetes.classe”:1,”bilhetes.data\_partida”:1}).pretty();

* (RE4) - Pesquisar as viagens disponíveis num dado período entre duas estações. (viagens\_between)

db.Viagens.find( {$and: [{data\_partida: data\_input},{data\_chegada: data\_input2}, {origem:o\_input},{destino:d\_input}]}} .pretty();

* (RE5) - Consultar a lista de lugares livres para uma viagem.

???

* (RE6) - Consultar o horário específico duma determinada estação.

JUNTAR ESTAS DUAS?

db.Viagens.find($and:[{origem=origem\_input},{data\_partida:{$gte: new Date()}}],{origem:1,destino:1,data\_partida:1}).pretty();

db.Viagens.find($and:[{destino=origem\_input},{data\_chegada:{$gte: new Date()}}],{origem:1,destino:1,data\_chegada:1}).pretty();

* (RE7) - Consultar lista de estações existentes.

???????????

db.Viagens.aggregate({$lookup:{from:Viagens,localField:origem,foreignField:destino,as:estacoes}, { $replaceRoot: { newRoot: { $mergeObjects: [ { $arrayElemAt: [ "$estacoes", 0 ] }, "$$ROOT" ] } } }, { $project: { estacoes: 1 } } ])

Do ponto de vista do administrador da aplicação deve ser possível:

* (RE8) - Consultar as viagens realizadas por um determinado comboio num dado período. (viagens\_comboio\_between)

db.Viagens.find({comboio=comboio\_id}).pretty();

* (RE9) - Saber quais os passageiros que viajaram entre duas estações num dado período. (clientes\_between\_estacoes)

db.Viagens.find({$and:[{“bilhetes.cliente”:{$ne: null}},{data\_partida:{$gte: data\_p\_input}},{data\_chegada:{$lte: data\_c\_input}}]},{“bilhetes.cliente”:1}).pretty();

* (RE10) - Saber quais os passageiros que participaram numa dada viagem. (clientes\_na\_viagem)

db.Viagens.find( {\_id:viagem\_input} , {“bilhetes.cliente”:1} ).pretty();

* (RE11) - Verificar quantos bilhetes foram vendidos num determinado período.

db.Viagens.aggregate( [ { $match: { “bilhetes.cliente”: { $ne: null } } }, { $count: "bilhetes\_vendidos" } ] );

* (RE12) - Calcular o valor total faturado num determinado período. (total\_faturado)

db.Viagens.aggregate( [ { $match: {$and: [{ “bilhetes.cliente”: { $ne: null }},{data\_partida:{$gte:data\_p\_input}},{data\_chegada:{$lte:data\_c\_input}}]}}, {$project: {\_id:0,preco:1}},{ $sum: "total\_faturado" } ] );

Do ponto de vista de um cliente, este deve conseguir:

* (RC1) - Inserir os dados para efetuar o registo.
* (RC2) - Inserir dados para a compra de um novo bilhete.
* (RC3) - Atualizar a informação inserida aquando do registo.

Do ponto de vista do administrador da aplicação, deve ser possível:

* (RC4) - Inserir uma nova viagem.
* (RC5) - Inserir dados sobre um novo comboio, assim como os lugares existentes neste.
* (RC6) - Adicionar novas estações.
* (RC7) - Atualizar os dados relativos a uma viagem.