Gestão de clientes de uma empresa de cloud computing

Bases de dados - Grupo 309

José Silva - up201705591@fe.up.pt
Nelson Gregório - up200900303@fe.up.pt
Rita Mota - up201703964@fe.up.pt

Prefácio

Os serviços de Cloud Computing e em demanda, *on demand*, têm-se tornado cada vez mais populares para pequenas e grandes empresas hospedaram as suas aplicações. Grande parte do sucesso deve-se à camada de abstração que introduziram, que permitiu aos programadores focaram-se mais no desenvolvimento e debugging da suas aplicações, sem se preocuparem com as infra estruturas em que estas irão correr. Apesar de serem direcionados para empresas, estes serviços também são altamente populares entre utilizadores individuais, sendo estes quem mais usufrui.

O trabalho realizado por este grupo visa aproximar um possível modelo que se utiliza atualmente na indústria. Existem vários exemplos dos quais nos inspiramos para ter um modelo mais fidedigno, como a **AWS**, *Amazon Web Services*, e a **GCP**, *Google Cloud Platform*. O leque de opções que apresentam aos possíveis clientes é muito vasto, como tal, de forma a cumprir os requerimentos do projeto foi necessário focarmo-nos em dois dos serviços oferecidos. O serviço de **instâncias virtuais**, em que um utilizador pode alugar uma máquina virtual ou um "espaço" para correr a sua aplicação, e o **serviço de armazenamento**, em que o utilizador pode alugar um espaço para armazenar dados.

O principal ponto de venda deste tipo de serviços é o facto de serem altamente personalizáveis ao gosto do cliente, sem que a experiência do utilizador seja comprometida com horas a finco de configuração. O feedback de anos na indústria permitiu a estes fornecedores de serviços criarem uma interface para o utilizador que não só é intuitiva, mas também oferece ferramentas de monitorização ultra detalhadas que quase tornou redundante o acesso à máquina.

Contexto

Cliente

Contém o respectivo email, password, nome, telefone, morada, saldo.

O email é um atributo único.

A password tem pelo menos 8 caracteres com obrigatoriedade de números, maiúsculas e símbolos e é cifrada antes de guardar.

Tem de ter associado, no mínimo, um cartão de crédito.

Cartão de crédito

Contém os dados do cartão de crédito, nomeadamente, nome do titular, número, CCV, validade e rede (banco).

País e Cidade

Indica em que centro de processamento de dados está localizado o serviço.

Sessão

Uma sessão é caracterizada por uma data de abertura e de fecho e está associada a uma instância. É criada sempre que o cliente arranca uma instância virtual.

É usada para controlar o período de tempo cobrado ao cliente.

Tipo de disco

Define o tipo de disco que o serviço de armazenamento usa, o custo mensal fixo e o custo por GB escrito (As leituras de disco são gratuitas). Escritas não são cobradas em armazenamentos associados a instâncias virtuais.

Tarifário

Indica o limite de transferência em TBs do serviço de armazenamento, o custo base e o custo por cada GB extra transferido se o limite for ultrapassado.

GPU

Uma VPS tem a opção reservar uma(s) placa(s) gráfica(s).

Cada modelo incorre um custo por hora enquanto uma sessão está aberta.

Sistema Operativo

Imagem do sistema operativo a correr numa VPS.

Tem um custo por hora caso o sistema não seja gratuito.

Backend

Software base sobre o qual uma APP corre. Por exemplo, Python (2.x/3.x) e módulos extra, Node.JS, Ruby (on Rails), entre outros.

Tem um custo por hora caso o sistema não seja gratuito.

Transferência

Guarda data, tamanho e nome do ficheiro transferido de/para o disco. Utilizado para fins de faturação.

Serviço

Um serviço é contratado e faturado mensalmente até ser cancelado.
Tem uma data de início e de cancelamento (que é nula até ser cancelado).
É cobrado no dia da data de início.

- **Armazenamento**

  Um serviço de armazenamento tem uma determinada capacidade.

  Tem associado um tipo de disco que usa, e um tarifário, ambos definidos pelo cliente no momento de criação.

  É cobrado em duas parcelas.

  Uma fixa, que depende da capacidade, do tipo de disco e do custo base do tarifário escolhido.

  Uma variável, consoante os GB escritos e os GBs transferidos caso o limite do tarifário seja ultrapassado.

- **Instância Virtual**

  Um serviço de instância virtual fornece ao cliente a capacidade de correr software sem se preocupar com a infraestrutura que este necessita.

  Tem um custo que arranque, cobrado por cada sessão iniciada, e um custo por hora (de acordo com as especificações da instância) enquanto uma sessão está aberta.

  A contratação deste serviço implica um serviço de armazenamento para os ficheiros usados/gerados na instância.

  - VPS

    Consiste em hardware virtualizado que é reservado para o cliente.

    Um cliente escolhe o número de vCPUs (CPUs virtuais), GBs de RAM, o(s) tipo(s) de GPU(s) (opcional) e o sistema operativo ao criar a VPS. Alteração de qualquer componente implica o cancelamento e recontratação do serviço.

  - APP

    Se um utilizador apenas quiser correr uma aplicação sem se preocupar com a escalabilidade e a manutenção do sistema, pode escolher este serviço. É necessário escolher o tipo de backend em que a aplicação correrá e o número máximo de instâncias que poderá iniciar.

Relação entre classes

Um **Cliente** quando se regista, precisa de obrigatoriamente utilizar um **cartão de crédito**, podendo ter mais que um associado à sua conta. Cada cartão de crédito só pode estar associado a um determinado Cliente.

A partir do momento que se encontra registado o cliente pode efetivamente começar a requisitar serviços. Os **serviços** são cobrados por mês. As **fórmulas** para cada serviço encontram-se na secção de **Lógica de faturação**.

Um serviço é **localizado** numa dada cidade, sendo que estas se encontram agrupadas pelo país a que pertencem. O preço numa dada cidade é definida por um conjunto de fatores de terceiros, como tal no mesmo país a variação de preço pode ser elevada.

O serviço de aplicação, APP, e de máquina virtual, VPS, são tipos de **Instâncias Virtuais**, ou seja, correm em hardware virtualizado, tendo um custo associado à sua iniciação e um custo por hora que é contabilizado por sessões e estas são cobradas no final do mês.

Como uma instância virtual não pode existir sem um disco associado, esta inclui obrigatoriamente um serviço de armazenamento. É usado para alojar o sistema operativo da VPS, que é fornecido sob a forma de imagens pré-feitas, assim como quaisquer outros ficheiros do cliente. De igual forma, a APP necessita de espaço para os ficheiros que vai correr (backend não conta como espaço ocupado) e outros ficheiros que possa gerar, podendo estes ser consultados no painel do serviço.

A faturação de um serviço de armazenamento é feita com base em dois fatores. O tarifário e o número de escritas(no caso de não haver instância virtual associada). O tarifário define o tráfego total disponível mensalmente, e caso seja ultrapassado é cobrada uma taxa por cada GB extra. O tráfego utilizado mensalmente é gerido mantendo registo de todas as **Transferência**s.

Lógica de faturação

APP:

(Backend > Custo por hora) *
(APP > número de instâncias) =
Custo por hora da APP

VPS:

(Sistema Operativo > Custo por hora) +
∑(GPU > Custo por hora) +
(VPS > nº vCPU cores) +
(VPS > GBs de RAM) * Preço por GB =
Custo por hora da VPS

Sessão:

(Sessão > Data de fecho - Data de abertura) (em horas) *
(Instância virtual > **Custo por hora (VPS ou APP)** +
(Instância virtual > Custo de arranque) =
Custo da sessão

Instância virtual:

∑ (cada sessão aberta no mês > **Custo da sessão**) num mês =
Custo mensal da instância

Armazenamento - Custo fixo:

(Tarifário > Custo base) +
(Tipo de disco > Custo mensal) =
(Armazenamento > Custo fixo mensal)

Armazenamento - Custo variável:

GBs transferidos = ∑(Transferências > Tamanho | Data.Mês = Mês de Faturação)

GBs escritos = GBs transferidos - ∑(Transferências > Tamanho | Leitura && Data.Mês = Mês de Faturação && (não é de instância virtual))

Se GBs transferidos > (Tarifário > Limite Transferência):
Custo transferência = [GBs transferidos - [(Tarifário > Limite Transferência) * 1000] * (Tarifário > Custo por GB extra)]

Custo transferência + [(Tipo de disco > Custo GB escrito) * GBs escritos] =
(Armazenamento > Custo variável mensal)

Armazenamento:

(Armazenamento > Custo fixo mensal) +
(Armazenamento > Custo variável mensal) =
Custo mensal do armazenamento

Serviço:

Custo mensal do armazenamento +
Custo mensal da instância =
Custo mensal do serviço

Faturação por mês:

∑ Serviço ativo [(Serviço > **Custo mensal**) + (Cidade > Taxa regional)] =
Custo total por mês para o cliente

Diagrama UML

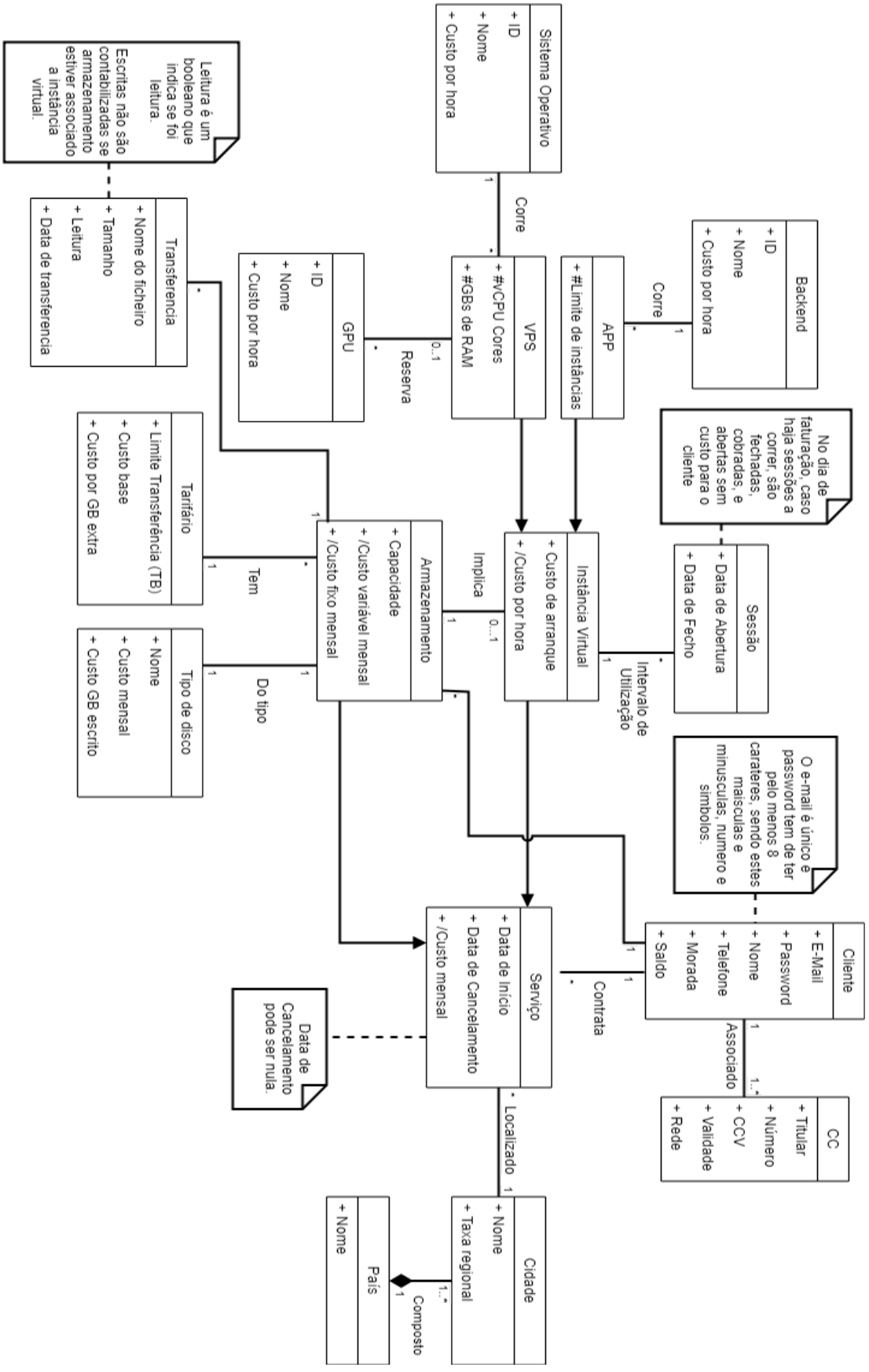
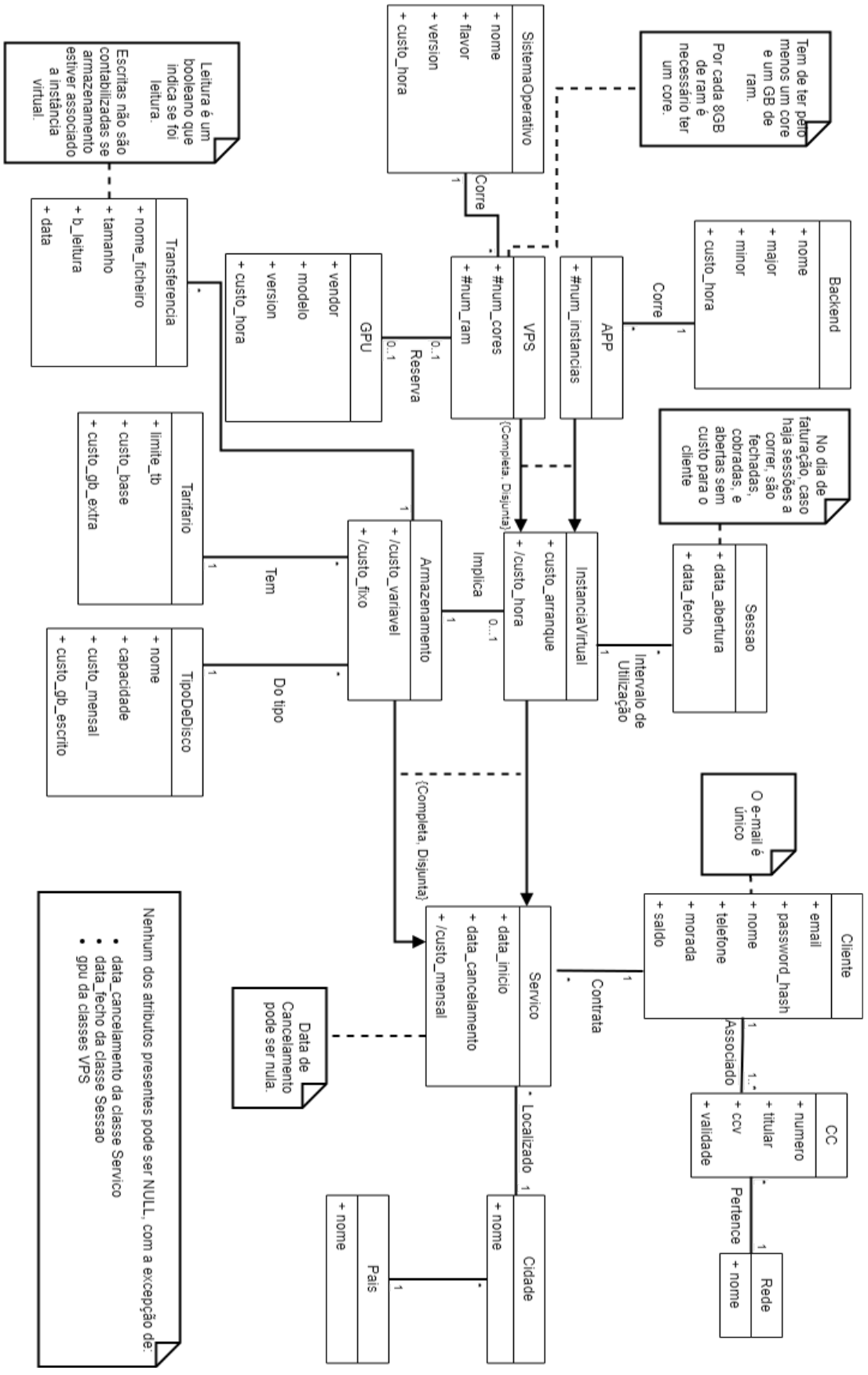


Diagrama UML (Revisto)



Modelo relacional

Cliente(email, password_hash, nome, telefone, morada, saldo)
email → password_hash, nome, telefone, morada, saldo
{email}$^+$ = {email, password_hash, nome, telefone, morada, saldo}
email é chave primária

CC(numero, titular, ccv, validade, rede → Rede, cliente → Cliente)
numero → titular, ccv, validade, rede, cliente
{numero}$^+$ = {numero, titular, ccv, validade, rede, cliente}
numero é chave primária
rede e cliente são chaves estrangeiras

Rede(id_rede, nome)
id_rede → nome
nome → id_rede
{id_rede}$^+$ = {id_rede, nome}
{nome}$^+$ = {id_rede, nome}
id_rede é chave primária

Pais(id_pais, nome)
id_pais → nome
nome → id_pais
{id_pais}$^+$ = {id_pais, nome}
{nome}$^+$ = {id_pais, nome}
id_pais é chave primária

Cidade(id_cidade, nome, pais → Pais)
id_cidade → nome, pais
nome, pais → id_cidade
{id_cidade}$^+$ = {id_cidade, nome, pais}
{nome_pais}$^+$ = {id_cidade, nome, pais}
id_cidade é chave primária
pais é chave estrangeira

Servico(id_serv, data_inicio, data_cancelamento, custo_mensal, cliente → Cliente, cidade → Cidade)
data_inicio, cliente → id_serv, data_cancelamento, custo_mensal, cidade
id_serv → data_inicio, cliente
{data_inicio, cliente}$^+$ = {data_inicio, cliente, id_serv, data_cancelamento, custo_mensal, cidade}

{id_serv}$^+$ = {id_serv, data_inicio, cliente, data_cancelamento, custo_mensal, cidade}
id_serv é chave primária
cliente e cidade são chaves estrangeiras

Tarifario(id_tarifario, limite_tb, custo_base, custo_gb_extra)
id_tarifario → limite_tb, custo_base, custo_gb_extra
limite_tb, custo_base, custo_gb_extra → id_tarifario
{id_tarifario}$^+$ = {id_tarifario, limite_tb, custo_base, custo_gb_extra}
id_tarifario é chave primária

TipoDeDisco(id_disco, nome, capacidade, custo_mensal, custo_gb_escrito)
id_disco → nome, capacidade, custo_mensal, custo_gb_escrito
nome, capacidade, custo_mensal, custo_gb_escrito → id_disco
{id_disco}$^+$ = {id_disco, nome, capacidade, custo_mensal, custo_gb_escrito}
id_disco é chave primária

Armazenamento(servico → Servico, custo_variavel, custo_fixo, tarifario → Tarifario, tipo_disco → TipoDeDisco)
servico → custo_variavel, custo_fixo, tarifario, tipo_disco
{servico}$^+$ = {servico, custo_variavel, custo_fixo, tarifario, tipo_disco}
servico é chave primária
servico, tarifario e tipo_disco são chaves estrangeiras

InstanciaVirtual(servico → Servico, custo_arranque, custo_hora, armazenamento → Armazenamento)
servico → custo_arranque, custo_hora, servico, armazenamento
{servico}$^+$ = {servico, custo_arranque, custo_hora, armazenamento}
serivico é chave primária
servico e armazenamento são chaves estrangeiras

Sessao(ins → InstanciaVirtual, data_abertura, data_fecho)
ins, data_abertura → data_fecho
{ins, data_abertura}$^+$ = {ins, data_abertura, data_fecho}
ins e data_abertura é chave primária composta
ins é chave estrangeira

Backend(id_backend, nome, major, minor, custo_hora)
id_backend → nome, major, minor, custo_hora
nome, major, minor → id_backend, custo_hora
{id_backend}$^+$ = {id_backend, nome, major, minor, custo_hora}
{nome, major, minor}$^+$ = {nome, major, minor, id_backend, custo_hora}
id_backend é chave primária

App(id_app → InstanciaVirtual, num_instancias, backend → Backend)

id_app → num_instancias, backend

$\{id\_app\}^+ = \{id\_app, num\_instancias, backend\}$

id_app é chave primária

backend é chave estrangeira

GPU(id_gpu, vendor, modelo, version, custo_hora)

id_gpu → vendor, modelo, version, custo_hora

vendor, modelo, version → id_gpu, custo_hora

$\{id\_gpu\}^+ = \{id\_gpu, vendor, modelo, version, custo\_hora\}$

$\{vendor, modelo, version\}^+ = \{vendor, modelo, version, id\_gpu, custo\_hora\}$

id_gpu é chave primária

SistemaOperativo(id_so, nome, flavor, version, custo_hora)

id_so → nome, flavor, version, custo_hora

nome, flavor, version → id_so, custo_hora

$\{id\_so\}^+ = \{id\_so, nome, flavor, version, custo\_hora\}$

$\{nome, flavor, version\}^+ = \{nome, flavor, version, id\_so, custo\_hora\}$

id_so é chave primária

VPS(id_vps → InstanciaVirtual, num_cores, num_ram, so → SistemaOperativo, gpu → GPU)

id_vps → num_cores, num_ram, so, gpu

$\{id\_vps\}^+ = \{id\_vps, num\_cores, num\_ram, so, gpu\}$

id_vps é chave primária

id_vps, so e gpu são chaves estrangeiras

Transferencia(nome_ficheiro, tamanho, b_leitura, data, armazenamento → Armazenamento)

armazenamento, data → nome, tamanho, b_leitura

$\{armazenamento, data\}^+ = \{armazenamento, data, nome, tamanho, b\_leitura\}$

armazenamento e data é chave primária composta

armazenamento é chave estrangeira

Análise de dependências funcionais e formas normais

Verificando as possíveis formas normais:

- ➔ Todos os atributos são atómicos e contêm apenas um elemento
- ➔ Não existem dependências parciais
- ➔ Não existem dependências transitivas
- ➔ Todas as dependências funcionais têm uma super chave do lado esquerdo

(Afirmações feitas com base nos fechos das relações incluídos na secção acima)

Com estas condições asseguradas, concluímos que o nosso modelo respeita as formas normais 1, 2, 3 e de Boyce-Codd, não havendo portanto nenhuma violação.

Não só se garantiu JOINs sem perdas, como também se removeu a possibilidade de quaisquer anomalias e , por fim, a lógica das FDs se encontram preservada.

Lista de restrições

Tal como indicado no UML, os únicos atributos que podem ser NULL são:

- ➔ data_cancelamento da classe Servico
- ➔ data_fecho da classe Sessao
- ➔ gpu da classe VPS

Considerar NOT NULL como restrição implícita em todos os atributos, excepto quando indicado o contrário.

Cliente:

- email tem de ser único entre clientes - PRIMARY KEY
- password_hash tem 64 caracteres - CHECK(LENGTH(password_hash) = 64)
- saldo tem o valor de 0 por defeito - DEFAULT 0
- saldo não pode ser negativo - CHECK(saldo >= 0)

CC:

- numero tem de ser único entre cartões - PRIMARY KEY
- ccv tem no máximo 3 algarismos - CHECK(ccv BETWEEN 0 AND 1000)
- validade tem de ser superior à data actual - CHECK(validade > DATE('now'))
- cliente é uma chave estrangeira para Cliente - FOREIGN KEY(cliente) REFERENCES Cliente(email)
- rede é uma chave estrangeira para Rede - FOREIGN KEY(rede) REFERENCES Rede(id_rede)

Rede:

- id_rede tem de ser único - PRIMARY KEY
- nome tem de ser único - UNIQUE

Pais:

- id_pais tem de ser único - PRIMARY KEY
- nome tem de ser único - UNIQUE

Cidade:

- id_cidade tem de ser único - PRIMARY KEY
- Não podem existir cidades com o mesmo nome no mesmo país - UNIQUE(nome, pais)
- pais é uma chave estrangeira para Pais - FOREIGN KEY(pais) REFERENCES Pais(id_pais)

Servico:

- id_serv tem de ser único - PRIMARY KEY
- data_cancelamento é NULL por defeito - DEFAULT NULL
- um cliente só pode criar um serviço a cada instante - UNIQUE(cliente, data_inicio)
- cliente é uma chave estrangeira para Cliente - FOREIGN KEY(cliente) REFERENCES Cliente(email)
- cidade é uma chave estrangeira para Cidade - FOREIGN KEY(cidade) REFERENCES Cidade(id_cidade)
- Só pode ser cancelado depois de ser iniciado - CHECK(data_cancelamento > data_inicio)

Tarifario:

- id_tarifario tem de ser único - PRIMARY KEY
- Não podem existir tarifários diferentes com os mesmos valores - UNIQUE(limite_tb, custo_base, custo_gb_extra)
- Todos os valores são positivos - CHECK(limite_tb > 0 AND custo_gb_extra > 0 AND custo_base > 0)

TipoDeDisco:

- id_disco tem de ser único - PRIMARY KEY
- Não podem existir tipos de disco diferentes com o mesmo nome e valores - UNIQUE(nome, capacidade, custo_mensal, custo_gb_escrito)
- Todos os valores são positivos - CHECK(capacidade > 0 AND custo_mensal > 0 AND custo_gb_escrito > 0)

Armazenamento:

- servico tem de ser único - PRIMARY KEY
- servico é uma chave estrangeira para Servico - FOREIGN KEY(servico) REFERENCES Servico(id_serv)
- Cada armazenamento é um serviço - UNIQUE
- tarifario é uma chave estrangeira para Tarifario - FOREIGN KEY(tarifario) REFERENCES Tarifario(id_tarifario)
- tipo_disco é uma chave estrangeira para TipoDeDisco - FOREIGN KEY(tipo_disco) REFERENCES TipoDeDisco(id_disco)
- custo_variavel tem de ser maior ou igual a zero e o custo_fixo tem de ser maior que zero - CHECK(custo_variavel >= 0 AND custo_fixo > 0)

InstanciaVirtual:

- servico tem de ser único - PRIMARY KEY
- servico é uma chave estrangeira para Servico - FOREIGN KEY(servico) REFERENCES Servico(id_serv)
- Cada instância é um serviço - UNIQUE
- armazenamento é uma chave estrangeira para Armazenamento - FOREIGN KEY(armazenamento) REFERENCES Armazenamento(servico)
- Todas as instâncias virtuais têm um armazenamento dedicado - UNIQUE
- custo_arranque e custo_hora têm de ser maiores que zero - CHECK(custo_arranque $>$ 0.0 AND custo_hora $>$ 0.0)

Sessao:

- Par (id_sessao, data_abertura) tem de ser único - PRIMARY KEY(id_sessao, data_abertura)
- id_sessao é uma chave estrangeira para InstanciaVirtual - FOREIGN KEY (id_sessao) REFERENCES InstanciaVirtual(servico)
- data_fecho é NULL por defeito - DEFAULT NULL
- Só pode ser cancelada depois de ser iniciada - CHECK(data_fecho $>$ data_abertura)

Backend:

- id_backend tem de ser único - PRIMARY KEY
- Não pode existir mais que uma backend com o mesmo nome, major e minor - UNIQUE(nome, major, minor)
- custo_hora tem de ser maior ou igual a zero - CHECK(custo_hora $>=$ 0.0)

App:

- id_app tem de ser único - PRIMARY KEY
- número mínimo de instâncias é um - DEFAULT 1
- número de instâncias tem de ser maior que zero - CHECK(num_instancias $>$ 0)
- id_app é uma chave estrangeira para InstanciaVirtual - FOREIGN KEY (id_app) REFERENCES InstanciaVirtual(servico)
- backend é uma chave estrangeira para Backend - FOREIGN KEY (backend) REFERENCES Backend(id_backend)

GPU:

- id_gpu tem de ser único - PRIMARY KEY
- Não pode existir mais que uma placa com o mesmo vendor, modelo e version - UNIQUE(vendor, modelo, version)
- custo_hora tem de ser maior que zero - CHECK(custo_hora $>$ 0.0)

SistemaOperativo:

- id_so tem de ser único - PRIMARY KEY
- Não pode existir mais que um SO com o mesmo nome, flavor e version - UNIQUE(nome, flavor, version)
- custo_hora tem de ser maior ou igual a zero - CHECK(custo_hora >= 0.0)

VPS:

- id_vps tem de ser único - PRIMARY KEY
- id_vps é uma chave estrangeira para InstanciaVirtual - FOREIGN KEY (id_vps) REFERENCES InstanciaVirtual(servico)
- so é uma chave estrangeira para SistemaOperativo - FOREIGN KEY(so) REFERENCES SistemaOperativo(id_so)
- gpu é uma chave estrangeira para GPU - FOREIGN KEY (gpu) REFERENCES SistemaOperativo(id_gpu)
- gpu é NULL por defeito - DEFAULT NULL
- número de cores e quantidade de ram tem de ser maior que zero - CHECK(num_cores > 0 AND num_ram > 0)
- Por cada 8 GB de RAM é necessário um core - CHECK((num_ram/num_cores) < 8.0)

Transferencia:

- Par (armazenamento, data) tem de ser único - PRIMARY KEY(armazenamento, data)
- armazenamento é uma chave estrangeira para Armazenamento - FOREIGN KEY(armazenamento) REFERENCES Armazenamento(servico)
- tamanho tem de ser maior que zero - CHECK(tamanho > 0)

Interrogações

1. Devolve todos os utilizadores, sem serviços abertos, que utilizaram pela última vez um serviço à mais de X dias.
2. Devolve todos os países com um datacenter a correr todas as versões disponíveis de um determinado Backend (NOME_BACKEND).
3. Devolve para cada país, a cidade com número de serviços a correr.
4. Devolve a duração média das instâncias virtuais de cada utilizador.
5. Devolve os utilizadores com serviço ativo no país (PAIS).
6. *****Devolve as aplicações a correr backend (BACK) entre as versões X.y e W.z.
7. Devolve todos os utilizadores que fizeram mais de X transferências num dado período de tempo (MIN_DATA até MAX_DATA).
8. *****Devolve o custo médio dos armazenamentos encerrados com tamanho entre X e Y.
9. Devolve a rede a partir da qual mais clientes est inscritos para utilizar qualquer tipo de serviço.
10. Devolve os utilizadores que durante o período de faturação (30 dias) ultrapassaram o limite de transferências do seu tarifário.

NOTA: Nas interrogações com "*****" foram incluídas duas versões da query, devido a terem performance parecida (dentro da quantidade de dados inseridos) e para depois ser esclarecido com a docente qual a melhor alternativa.

Gatilhos

1. **a) Fecha os serviços de um cliente quando este remove todos os seus cartões de crédito e não tem saldo disponível, evitando assim inconsistências em termos da faturação. (ON DELETE - AFTER)**

   A verificação mostra o número de clientes com serviços ativos, saldo a zero e apenas um cartão. Remove os cartões de um cliente nessas condições. Volta a mostrar o número de clientes que terá de ser menos um que o valor inicial.

   b) Bloqueia a inserção de um serviço de o cliente não tem saldo nem CCs associados. (ON INSERT - BEFORE)

   Correr a verificação do gatilho 1, depois correr verificação do gatilho X.
   O comportamento esperado é um erro com a mensagem "No funds to start the service, add a CC".

2. **Define o custo fixo derivado ao adicionar um novo tipo de armazenamento. (ON INSERT - AFTER)**

   É adicionado um serviço de armazenamento com atributos escolhidos de forma a dar um determinado custo mensal. Ao ser pedido o custo deste novo serviço, o valor deve refletir o valor determinado. É devolvido "CORRETO" ou "ERRADO" duas vezes para o custo fixo do armazenamento e para o custo mensal do serviço.

3. **Atualiza preço do armazenamento ao alterar as condições tarifário. (ON UPDATE - AFTER)**

   É mostrado o custo fixo X de um armazenamento com um tarifário Y.
   O custo base desse tarifário sofre um aumento de uma unidade.
   É mostrado novamente o custo fixo de Y que deve ser agora $X + 1$.