UNIVERSI
DADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO
SCC 0240 BASES DE DADOS

Inclusão Digital em Escolas

Docente: Profa. Dra. Elaine Parros Machado de Souza
PAE: André Moreira Souza

Alunos	#USP
Ciro Grossi Falsarella	11795593
Eduardo Garcia de Gaspari Valdejão	11795676
Gabriel Rosati Bryan Tavares	11355831
Henrico Lazuroz Moura de Almeida	12543502
Victor Lucas de Almeida Fernandes	12675399

São Carlos - 16 de Abril de 2023

Sumário

São Carlos - 16 de Abril de 2023

Introdução

Com a digitalização do mundo, a educação ficou cada vez mais dependente da internet e de dispositivos digitais, tais como celulares, tablets e computadores pessoais. Porém, a realidade brasileira é outra. Parte dos alunos não possuem acesso ao mundo digital, seja porque não têm acesso às ferramentas necessárias dentro de casa, ou porque as escolas não são capazes de prover tais tecnologias. Nesse contexto, a escassez de recursos resulta na marginalização de parte da população, precarizando o ensino básico no país. Para atender a essa demanda, propomos a execução de um sistema colaborativo de compartilhamento de hardware para inclusão digital em escolas.

Nesse sistema, as escolas se cadastram e ficam responsáveis por cadastrar os seus alunos. Nesse primeiro momento, o foco principal é em descobrir o grau de acesso dos alunos ao digital. Além do mais, as escolas poderão usar de posts para expor as suas demandas para a digitalização. Assim, construiremos um feed de rolagem com solicitações e demandas das diferentes escolas para a inserção no digital.

Com o intuito de auxiliar, surgem os colaboradores, que podem oferecer seus itens para as escolas. Dessa forma, aqueles que quiserem ajudar, podem procurar no feed por demanda aparelhos semelhantes aos que querem doar. Alternativamente, os colaboradores podem usar posts para exporem os produtos que tem disponíveis, construindo um feed de doações. Destarte, aqueles que desejam doar equipamentos podem descobrir quais instituições estudantis eles conseguem ajudar.

Para facilitar o processo de escolha entre os diferentes colaboradores e escolas, existirá um sistema de avaliação. Após cada doação, ambas as partes deverão dar uma nota para a outra parte. Dessa forma, será possível escolher para quem os produtos serão enviados com confiança.

Portanto, o sistema facilita que escolas sejam digitalmente incluídas por meio da exibição das necessidades tecnológicas das escolas, dos sobressalentes de colaboradores e da classificação dos usuários. Sendo assim, o software conectará as peças tecnológicas sobressalentes com quem mais precisa delas.

Modelo Entidade-Relacionamento

Levantamento de requisitos

Cada usuário é identificado por um **e-mail** (chave única), além de possuir **nome** (no máximo 60 caracteres alfanuméricos) e **telefone** (string de 13 caracteres numéricos). Também apresenta um **endereço**, composto por **CEP** (8 caracteres), **número** (inteiro) e **complemento** (máximo 20 caracteres alfanuméricos). Os **usuários** são classificados pelo atributo **cargo**, que pode receber os seguintes valores: **administrador** ou **usuário comum**, de forma que o segundo é especializado para uma **escola** ou um **colaborador**.

Um **usuário** pode **avaliar** e receber **avaliações** de outros **usuários** com uma **nota** (1 a 5 estrelas). A partir de um **chat**, é possível que os **usuários** enviem e recebam **mensagens** com um **texto** (no máximo 256 caracteres), identificadas por uma **data** (data e hora de envio) e os **usuários** presentes na relação. Todo **usuário** possui um **perfil público** com **nome** (no máximo 30 caracteres), onde há sua **avaliação geral** (média das avaliações recebidas) e **número de transações** (calculado de forma derivada a partir das movimentações de itens). Todo **usuário** é capaz de fazer um **post**, que é especializado entre **doação** e **solicitação**, sempre representando um **item**. Cada **post** é identificado pela **data** (data e hora de publicação), e possui **título** (no máximo 30 caracteres) e **imagem** (arquivo jpg correspondente ao produto). Nos **posts**, o **usuário** pode acrescentar **comentários**, agregando informações e opiniões sobre os **itens**. Cada um deles é identificado pela **data** (data e hora de publicação) e possui um **conteúdo** (no máximo 256 caracteres). Além disso, **usuários** realizam **transações** que movimentam **itens**, identificadas por uma **data** (data e hora da transação).

As **escolas**, além de todos os atributos de **usuário**, possuem **código MEC** (8 dígitos), para garantir veracidade, e um **responsável**, campo composto por **CPF** (11 caracteres), **nome** (máximo de 60 caracteres) e **telefone** (string de 13 caracteres numéricos). Cada **escola** realiza o **cadastro** dos seus **alunos**.

Um **aluno** é identificado a partir de seu **CPF** (11 caracteres), e possui dados como **nome** (no máximo 60 caracteres alfanuméricos), **telefone** (13 caracteres numéricos), **endereço** (igualmente composto como anteriormente citado), **e-mail** (no máximo 60 caracteres

alfanuméricos), **data de nascimento** (data com dia e ano) e **idade** (calculada de forma derivada a partir da **data de nascimento**). Os **alunos** recebem **itens** provenientes das **doações** gerenciadas pelo sistema.

Cada **colaborador** é especializado como **pessoa física** ou **pessoa jurídica**, e apresenta um **documento** diferente de acordo com a especialização. No primeiro caso, possui um **CPF** (11 caracteres), além de **data de nascimento** (data com dia e ano) e **idade** (calculada de forma derivada a partir da **data de nascimento**). Já no segundo caso, possui **CNPJ** (14 caracteres) e **razão social** (30 caracteres). Os **colaboradores** são responsáveis pela **doação** de itens.

Todo **item** é identificado por seu **número serial** (máximo 30 caracteres alfanuméricos). Ademais, está relacionado a um **doador** (usuário colaborador do sistema) e um **usuário atual** (usuário escola/aluno do sistema), e possui um **valor** (inteiro) e uma **descrição** (máximo de 256 caracteres alfanuméricos). Os **itens** são obrigatoriamente divididos em **categorias**, que possuem um **nome** (no máximo 60 caracteres alfanuméricos) e uma **descrição** (máximo de 256 caracteres alfanuméricos).

O **administrador** pode **remover usuários comuns**, **posts** e **comentários** assim como verificar **perfis públicos** para garantir que as normas do sistema não sejam violadas. Também é responsável por **autenticar escolas** para que elas possam entrar no sistema, com intenção de reduzir fraudes. Os **administradores** também **gerenciam** as **categorias dos itens**, podendo criar, remover ou modificar elas.

Principais funcionalidades

Todos os usuários podem acessar o perfil público de outros usuários, adicionar comentários em seus posts e enviar mensagens uns para os outros por meio de chats e realizarem avaliações mútuas. Além disso, os usuários podem fazer posts, que divulgam itens para doações ou explicitem itens necessitados.

As escolas se responsabilizam por cadastrar os seus alunos e manter seus dados atualizados. Ela pode remover alunos caso seja interesse. O usuário escola pode observar o feed dos itens disponíveis para doação em sua região.

Colaboradores podem observar o feed dos itens solicitados em sua região. A partir disso podem realizar doações, que devem ser armazenadas no sistema, para que posteriormente o usuário possa verificar suas movimentações.

Os administradores podem remover usuários comuns que julgarem incoerentes, além de também poderem excluir posts. É preciso que um administrador autentique uma escola para o perfil dela ser validado como possível instituição beneficiada pelo projeto. Apenas administradores podem alterar a categoria de um item e criar novas categorias.

Em suma, escolas e colaboradores devem nutrir respectivamente os feeds de itens solicitados e itens disponíveis por meio de posts. Ambos devem manter seus perfis atualizados e avaliarem a outra parte depois de realizadas transações. A escola fica então como responsável por tudo aquilo que envolve os alunos, desde o cadastramento até a entrega do item. Para os administradores, cabe a função de selecionar posts e perfis que não deveriam fazer parte do sistema.

Análise de Ciclos

01. Ciclo Escola-Transação-Aluno-Item

Há a presença de um ciclo com dependências entre as entidades Escola, Aluno, Item e Transação, através dos relacionamentos Cadastra, Recebe, Acessa e Registra Movimentação. Não foram encontradas alternativas para esse ciclo, haja vista que os relacionamentos são funcionalidades distintas e devem ser separados. O ciclo representa dependência, porque apenas alunos cadastrados pelas escolas podem receber itens. Além disso, o ciclo não pode ser quebrado por conta da dependência existencial da Transação com Item.

02. Ciclo Usuário

Há a presença de dois ciclos de dependência na entidade Usuário, através dos autos-relacionamentos Avalia e Mensagem. Os ciclos não podem ser removidos, uma vez que isso implicaria na remoção das funcionalidades. Os ciclos representam dependência, haja vista que os relacionamentos acontecem entre diferentes usuários. Para que não haja **inconsistência** no sistema, a aplicação deve **restringir** que o usuário possa apenas **avaliar** e **enviar mensagem** para um usuário diferente.

03. Ciclo Escola-Administrador-Categoria-Item-Aluno

Há a presença de um ciclo entre as entidades Escola, Aluno, Item, Administrador e Categoria, através dos relacionamentos Remove (serve também para o Autentica), Gerencia, Divide, Recebe e Cadastra. **Não foram encontradas possíveis inconsistências** geradas por esse ciclo, haja vista que os relacionamentos são funcionalidades distintas e devem ser separados.

04. Ciclo Usuário-Post-Comentário

Há a presença de um ciclo de dependência entre as entidades Usuário, Post e Comentário, através do relacionamentos Faz, Refere-se e Cria. Não foram encontradas alternativas para esse ciclo, haja vista que os relacionamentos são funcionalidades distintas e devem ser separados. O ciclo não pode ser quebrado por conta de dependência existencial do comentário com relação ao usuário. Para que não haja **inconsistência** no sistema, a

aplicação deve **restringir** que o usuário só possa **comentar** no post de outro usuário.

05. Ciclo Usuário-Perfil Público-Administrador

Há a presença de um ciclo de dependência entre as entidades Usuário, Perfil Público e Administrador, através dos relacionamentos Possui, Verifica e Remove (serve também para o relacionamento Autentica). Não foram encontradas alternativas para esse ciclo, haja vista que os relacionamentos são funcionalidades distintas e devem ser separados. O ciclo representa dependência, pois um perfil público só pode ser verificado caso exista e pertença a um usuário, por conta da dependência existencial do mesmo.

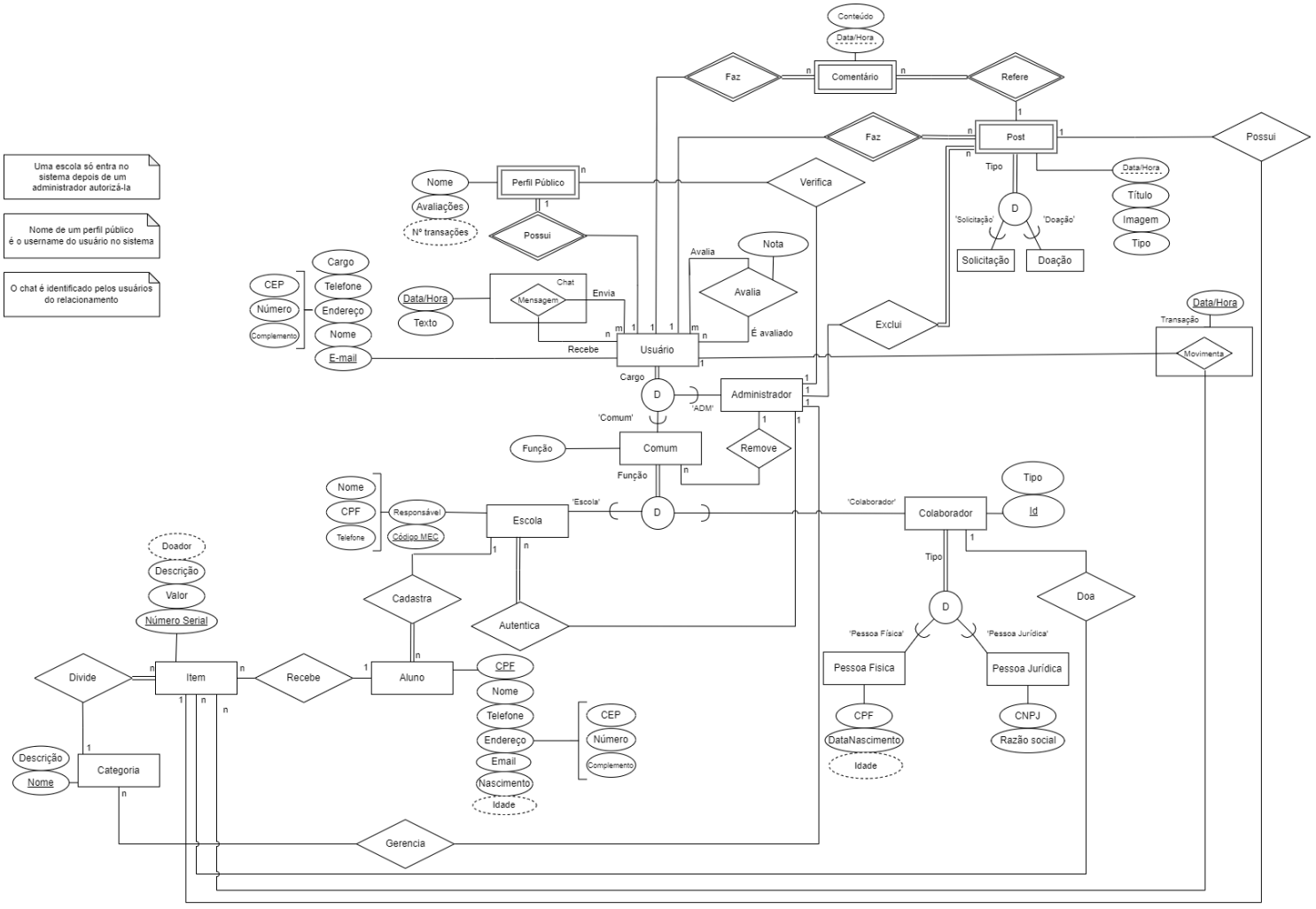
06. Ciclo Colaborador-Item-Categoria-Administrador

Há a presença de um ciclo entre as entidades Colaborador, Item, Categoria e Administrador, através dos relacionamentos Doa, Divide, Gerencia e Remove (serve também para o relacionamento Autentica). Não foram encontradas alternativas para esse ciclo, haja vista que os relacionamentos são funcionalidades independentes e devem ser separados.

07. Ciclo Usuário-Post-Item-Aluno-Escola

Há a presença de um ciclo entre as entidades Usuário, Post, Item, Aluno e Escola, através dos relacionamentos Cria, Possui, Recebe e Cadastra. **Não foram encontradas possíveis inconsistências** geradas por esse ciclo, haja vista que os relacionamentos são funcionalidades distintas e devem ser separados.

Diagrama


Uma escola só entra no sistema depois de um administrador autorizá-la
Nome de um perfil público é o username do usuário no sistema
O chat é identificado pelos usuários do relacionamento
Conteúdo
Data/Hora
Comentário
Faz
Refere
Post
Possui
Tipo
Data/Hora
Título
Imagem
Tipo
'Solicitação'
'Doação'
Solicitação
Doação
Nome
Avaliações
Nº transações
Perfil Público
Verifica
Possui
Nota
Avalia
É avaliado
Chat
Mensagem
Envia
Recebe
Data/Hora
Texto
Cargo
Telefone
Endereço
Nome
E-mail
CEP
Número
Complemento
Usuário
Exclui
Data/Hora
Transação
Movimenta
Cargo
'ADM'
'Comum'
Administrador
Remove
Função
Comum
Nome
CPF
Telefone
Responsável
Código MEC
Escola
'Escola'
'Colaborador'
Colaborador
Tipo
Id
Doador
Descrição
Valor
Número Serial
Cadastra
Autentica
Doa
Divide
Item
Recebe
Aluno
CPF
Nome
Telefone
Endereço
Email
Nascimento
Idade
CEP
Número
Complemento
'Pessoa Física'
'Pessoa Jurídica'
Pessoa Física
Pessoa Jurídica
CPF
DataNascimento
Idade
CNPJ
Razão social
Descrição
Nome
Categoria
Gerencia

Mudanças relacionadas a primeira entrega

Diagrama

Realizamos algumas alterações em nosso diagrama em relação a primeira entrega. Algumas delas são correções do que tínhamos feito de errado, outras são alterações para melhor funcionamento do sistema.

01. Usuário Comum

a. Criamos uma nova entidade chamada Usuário Comum que herda da entidade usuário, com atributo seletor ‘função’.
b. As entidades Escola e Colaborador agora herdam de Usuário Comum.
c. A entidade Administrador agora herda de Usuário, com atributo seletor ‘cargo’.
d. Tanto a generalização de Usuário como a de Usuário Comum são obrigatórias

02. Post & Comentário

a. Ambas as entidades têm agora o atributo ‘Data / Hora’ como chave parcial
b. Post agora tem um atributo seletor de tipo para sua generalização
c. O relacionamento refere se torna obrigatório para comentário

03. Transação

a. Transação agora é uma agregação entre usuário e item
b. A agregação tem como chave a sua data/hora

04. Colaborador

a. Declaramos a chave ID como uma cópia do CNPJ caso pessoa jurídica e CPF caso pessoa física
b. Declaramos o atributo seletor ‘tipo’ para a sua generalização
c. As Pessoas Jurídicas agora também possuem o atributo de ‘Razão Social’

05. Item

a. Ajustamos para que seu identificador não fosse mais sintético e passasse a se referir ao seu código serial

Documento

Além de alterações em nosso diagrama, alteramos parte de nossa documentação para que ela representasse melhor o que está proposto.

01. Levantamento de requisitos

a. Adicionamos os tipos dos atributos das entidades
b. Adicionamos os relacionamentos entre as diferentes entidades.

02. Funcionalidades

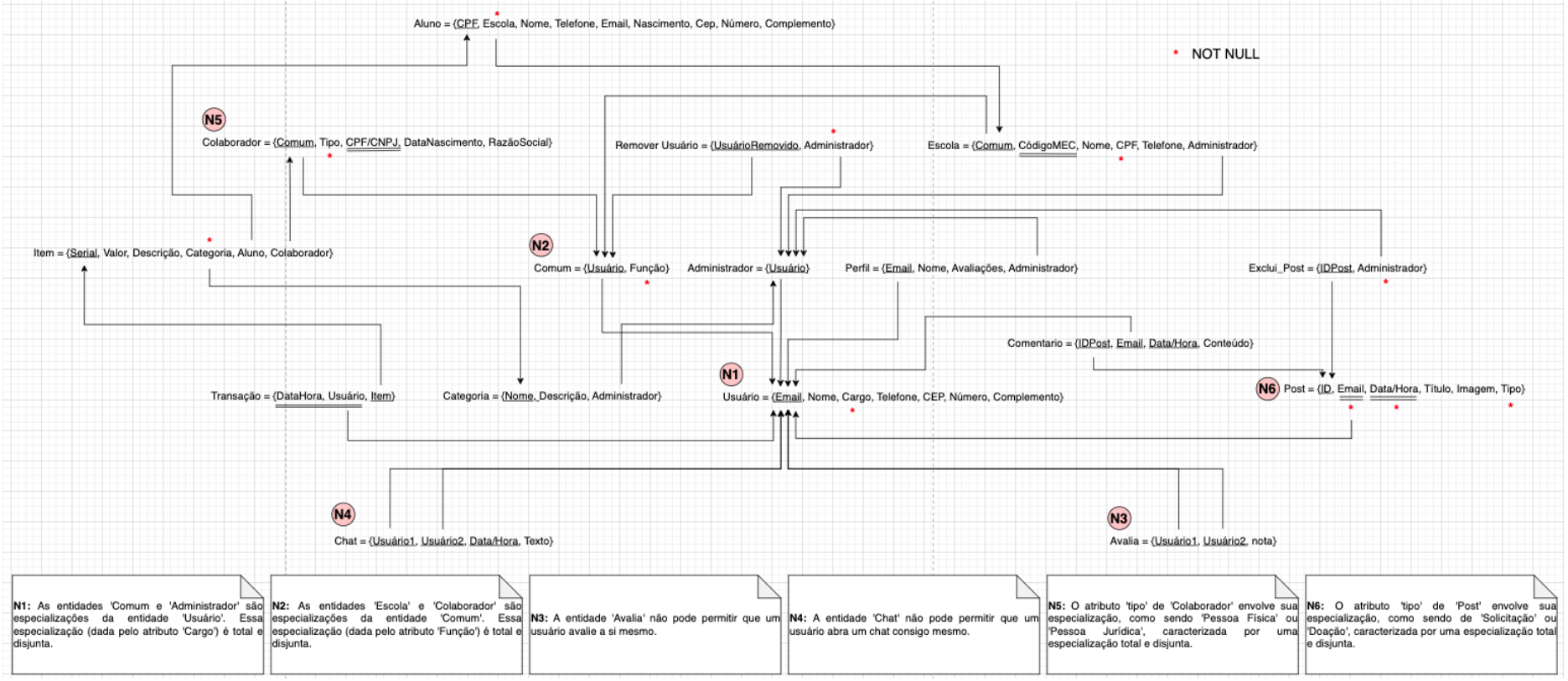
a. Especificação do relacionamento do aluno
b. Resumo geral de como o sistema vai funcionar

03. Ciclos

a. Adicionamos a restrição de desenvolvimento ao ciclo Usuário
b. Adicionamos a restrição de desenvolvimento ao ciclo Usuário-Post-Comentário
c. Documentamos o ciclo Usuário-Post-Item-Aluno-Escola

Modelo Relacional

Esquema Relacional



São Carlos - 16 de Abril de 2023

Discussão sobre os mapeamentos propostos

01. Especialização da entidade Usuário

Solução Adotada: foi mapeada com uma tabela para a entidade genérica ‘Usuário’ e duas tabelas para as entidades específicas ‘Administrador’ e ‘Comum’.

Vantagens: escolhemos essa solução pois tanto a entidade genérica quanto as entidades específicas possuem diversos relacionamentos com outros conjuntos de entidades. Assim, o mapeamento dessas relações foi facilitado.

Desvantagens: essa solução de mapeamento não garante a especialização total da entidade ‘Usuário’, assim como a disjunção entre as entidades específicas. Também exige um custo computacional de memória elevado.

Alternativas: outra solução possível seria agrupar todas as entidades específicas em uma tabela só, porém geraria um aumento de dificuldade ao mapear os relacionamentos dessas entidades. Também poderíamos criar tabelas somente para as entidades específicas, no entanto, não escolhemos essa solução por conta das diversas relações e atributos que a entidade genérica ‘Usuário’ possui.

02. Especialização da entidade Usuário Comum

Solução Adotada: foi mapeada com uma tabela para a entidade genérica ‘Comum’ e duas tabelas para as entidades específicas ‘Escola’ e ‘Colaborador’

Vantagens: escolhemos essa solução pois tanto a entidade genérica quanto as entidades específicas possuem relacionamentos com outros conjuntos de entidades. Assim, o mapeamento dessas relações foi facilitado.

Desvantagens: essa solução de mapeamento não garante a especialização total da entidade ‘Comum’, assim como a disjunção entre as entidades específicas. Também exige um custo computacional de memória elevado.

Alternativas: outra solução possível seria agrupar todas as entidades específicas em uma tabela só, porém geraria um aumento de dificuldade ao mapear os relacionamentos dessas entidades. Também poderíamos criar tabelas somente para as entidades específicas, no

entanto, não escolhemos essa solução por conta da relação de remoção que a entidade genérica ‘Comum’ possui.

03. Especialização da entidade Colaborador

Solução Adotada: Mapeamos a entidade ‘Colaborador’ em uma única tabela, visto que as entidades específicas não possuem relacionamentos com outros conjuntos de entidade, e têm poucos atributos específicos.

Vantagens: Criando-se apenas uma tabela, é economizada memória de armazenamento, ainda garantindo a especialização total da entidade.

Desvantagens: Por possuir relacionamentos e atributos de especialização, é necessário que haja tratamento, em nível de aplicação, da integridade do sistema, para que o mesmo seja condizente ao mapeamento criado e para que os valores armazenados sejam válidos.

Alternativas: Alternativamente, seria possível a criação de uma tabela para cada especialização da entidade, contornando a necessidade de tratamento em nível de aplicação, porém, sendo mais custoso em questão da memória de armazenamento.

04. Especialização da entidade Post

Solução Adotada: Foi mapeada a entidade genérica 'Post' e suas especializações em uma única tabela. Esse mapeamento se mostrou viável uma vez que as especialidades da entidade Post não possuem atributos próprios nem participam de relacionamentos.

Vantagens: Haja vista que as entidades específicas não possuem atributos, a criação de apenas uma tabela economiza memória de armazenamento e poder de processamento. O mapeamento garante especialização total e disjunção.

Desvantagens: Não foram notadas desvantagens na solução adotada.

Alternativas: Uma solução alternativa seria a criação de uma tabela única para cada especialização, todavia consumiria mais memória de armazenamento e poder de processamento, pois, para cada consulta na tabela genérica Post, também seria necessário consultar a tabela específica Além disso, a solução não garantiria disjunção e especialização total.

05. Agregação Chat

Solução Adotada: criamos uma nova tabela para a agregação ‘Chat’, com chave composta por ‘Usuário 1 + Usuário 2 + Data/Hora’.

Vantagens: é possível obter o histórico de mensagens entre os usuários.

Desvantagens: exige custo computacional de memória por criar mais uma tabela.

Alternativas: não encontramos outras soluções para esse mapeamento, visto que não existe uma chave identificadora específica da agregação, assim como atributos do relacionamento gerador da mesma.

06. Agregação Transação

Solução Adotada: criamos uma nova tabela para a agregação ‘Transação’, com a chave primária sendo ´Item´ e chave secundária sendo composta por ‘Data/Hora’ + ‘Usuário’.

Vantagens: é possível obter o histórico de itens movimentados por um usuário.

Desvantagens: exige custo computacional de memória por criar mais uma tabela.

Alternativas: não encontramos outras soluções para esse mapeamento.

07. Relacionamento ‘Avalia’ entre ‘Usuário’ - N:M

Solução Adotada: Mapeamos o relacionamento em uma tabela ‘Avalia’, armazenando o usuário que avalia e o usuário que é avaliado, como uma única chave composta.

Vantagens: A solução permite que o relacionamento do tipo “muitos-para-muitos” seja preservado.

Desvantagens: A criação de uma nova tabela aumenta o custo de memória.

Alternativas: Não foram identificadas alternativas para esse mapeamento.

08. Relacionamento ‘Exclui’ entre 'Post' e 'Administrador' - 1:N

Solução Adotada: Houve a criação da tabela “ExcluiPost”, com a chave primária sendo a chave estrangeira 'Post'. Essa solução

alternativa foi adotada por conta de que o relacionamento ocorre com pouca frequência.

Vantagens: Essa solução evita o excesso de atributos nulos na tabela “Post”. Além disso, a cardinalidade do relacionamento é preservada.

Desvantagens: Há maior uso de memória de armazenamento e poder de processamento com a utilização dessa solução.

Alternativas: A alternativa mais viável seria a criação de um atributo “RemovidoPor”, na entidade “Post”. A desvantagem desta alternativa é o excesso deste atributo com valor null, já que a remoção de um Post é pouco frequente.

09. Relacionamento ‘Doa’ entre ‘Colaborador’ e ‘Item’ - 1:N

Solução Adotada: adicionamos um atributo ‘Colaborador’ na tabela ‘Item’.

Vantagens: essa solução garante que um colaborador pode estar associado a diversas doações, porém cada item possui somente um doador. Dessa forma, a cardinalidade da relação é preservada.

Desvantagens: possível presença de valores nulos, por conta do relacionamento possuir participação parcial.

Alternativas: uma solução alternativa seria a criação de uma tabela para o relacionamento ‘Doa’. No entanto, ocasionaria a replicação desnecessária de dados.

10. Relacionamento ‘RemoverUsuário’ entre ‘Administrador’ e ‘Comum’ - 1:N

Solução Adotada: Foi criada uma tabela para o relacionamento ‘Remove’, sendo o usuário a chave primária e o administrador um atributo não nulo. Essa solução foi escolhida pois é uma operação pouco recorrente, mas que deve ter seu histórico registrado.

Vantagens: Essa solução evita o excesso de atributos nulos na tabela ‘Comum’, pois não armazena um administrador para usuários não removidos.

Desvantagens: Cria uma nova tabela para armazenar os usuários removidos, aumentando o consumo de memória.

Alternativas: seria possível criar um atributo ‘RemovidoPor’ na tabela ‘Comum’, que se ligaria a relação ‘Administrador’, porém

essa alternativa não compensa pois a operação de remoção ocorre poucas vezes.

11. Relacionamento ‘Verifica’ entre ‘Administrador’ e ‘Perfil’ - 1:N

Solução Adotada: Inserimos um atributo ‘Administrador’ à entidade de ‘Perfil’, que representa o administrador que verificou o perfil público, permitindo com que o mesmo administrador seja responsável por várias verificações, mas que cada perfil possua apenas um verificador.

Vantagens: A vantagem se dá pela preservação da cardinalidade da relação.

Desvantagens: possível presença de valores nulos, por conta do relacionamento possuir participação parcial.

Alternativas: Alternativamente, poderia ser criada uma tabela separada, que relacionasse cada perfil com seu respectivo administrador, porém criando redundância dos dados armazenados, pela recorrência da criação de perfis.

12. Relacionamento ‘Cadastra’ entre ‘Escola’ e ‘Aluno’ - 1:N

Solução Adotada: Houve a criação do atributo 'Escola' na entidade 'Aluno', que representa o relacionamento "Cadastra". Este atributo é chave estrangeira da chave primária de ‘Escola’, garantindo a participação total de 'Aluno' nesta relação.

Vantagens: Por haver participação total, todos os alunos devem ser cadastrados por uma escola. Por isso, este campo nunca será null e não há a necessidade de separar essa relação em uma tabela alternativa, economizando memória.

Desvantagens: Não foram notadas desvantagens na solução adotada.

Alternativas: Uma alternativa seria mapear essa relação em uma tabela separada, porém seria mais custoso computacionalmente em relação à memória, sem prover vantagens.

13. Relacionamento ‘Autentica’ entre ‘Administrador’ e ‘Escola’ - 1:N

Solução Adotada: adicionamos um atributo ‘Administrador’ na tabela ‘Escola’.

Vantagens: essa solução garante que um administrador pode estar associado a diversas autenticações, porém cada escola possui somente um administrador que a autenticou. Dessa forma, a cardinalidade da relação é preservada. Além disso, garante a participação total do relacionamento ao exigir que o administrador seja não nulo.

Desvantagens: não foram identificadas desvantagens em mapear desta maneira.

Alternativas: uma solução alternativa seria a criação de uma tabela para o relacionamento 'Autentica'. No entanto, ocasionaria a replicação desnecessária de dados.

14. Entidade Fraca 'PerfilPúblico' 1:1

Solução Adotada: A entidade recebeu sua própria tabela, uma vez que possui atributos próprios, porém por ser identificada apenas pelo 'Usuário' dono do mesmo, a chave é Email que identifica que cada 'Usuário' tem apenas seu 'PerfilPúblico' único.

Vantagens: Ao tomar essa solução é garantido que 'PerfilPúblico' é uma entidade fraca, uma vez que só pode ser identificada pela usuário Owner da mesma. Nesse caso, é garantida a participação total, uma vez que só existe um perfil se o usuário também existir.

Desvantagens: Por PerfilPublico estar mapeado em uma entidade diferente de Usuario, há maior custo de junção ao selecionar.

Alternativas: Uma alternativa seria juntar as relações PerfilPublico e Usuario em uma mesma entidade, porém a desnormalização permitiria possíveis anomalias de inserção, atualização e remoção.

15. Entidade Fraca 'Comentário'(de 'Faz' e 'Refere') - 1:N

Solução Adotada: Mapeamos a entidade 'Comentário' em uma tabela própria, como se fosse uma entidade forte, tendo a chave primária composta pelos atributos 'IDPost', ´Email´ e ´Data/Hora´.

Vantagens: O mapeamento preserva a característica de entidade fraca do 'Comentário', necessitando de uma entidade 'Post' e 'Usuário', assim como a cardinalidade e participação total do relacionamento.

Desvantagens: Não foram identificadas desvantagens em mapear desta maneira.

Alternativas: Não foram identificadas alternativas para esse mapeamento.

16. Entidade Fraca ‘Post’

Solução Adotada: Foi-se adotado um identificador sintético como chave primária da tabela 'Post', como se fosse uma entidade forte. Por isso, a chave composta modelada no MER foi utilizada como chave secundária.

Vantagens: A entidade 'Post' continua sendo uma entidade fraca de Usuário, haja vista que todo Post necessita de um Usuário (not null) e possui uma data/hora. Além disso, a cardinalidade é respeitada e é garantida a participação total do relacionamento.

Desvantagens: Não foram notadas desvantagens em mapear desta maneira.

Alternativas: Não foram identificadas alternativas para esse mapeamento.

17. Identificador sintético para ‘Post’

Solução Adotada: transformamos a chave primária da entidade ‘Post’ que era composta por ‘Usuário + Data/Hora’ em chave secundária não nula, e criamos um identificador sintético para ser chave primária.

Vantagens: essa solução evita a replicação da chave primária composta em todos os relacionamentos da entidade ‘Post’, além de facilitar a identificação dela por depender somente de um número inteiro.

Desvantagens: o identificador sintético não atribui significado semântico para a chave.

Alternativas: outra solução possível seria não criar o identificador sintético e replicar a chave primária original nos relacionamentos da entidade ‘Post’.

18. Atributo derivado ‘NroTransações’

Solução Adotada: O atributo não foi armazenado diretamente na base de dados uma vez que é um dado mutável, que se altera toda vez que uma nova transação é feita. Sendo assim, ao receber uma doação ou doar um item esse valor é recalculado, além de não ser essencial para outros relacionamentos.

Vantagens: O cálculo em tempo real do número de transações garante maior precisão para tal informação e reduz a possibilidade de inconsistências em relação a movimentação de itens.

Desvantagens: O atributo deve ser recalculado sempre que necessário o que pode exigir um pouco mais de processamento.

Alternativas: Uma alternativa viável seria armazenar esse dado diretamente na base de dados, nesse caso não foi optado.

19. Atributo derivado ‘Doador’ de ‘Item’

Solução Adotada: O atributo foi armazenado diretamente na base de dados, como ‘Colaborador’ na entidade ‘Item’, já que é uma informação essencial para outros relacionamentos.

Vantagens: Armazená-lo diretamente preserva a cardinalidade da relação.

Desvantagens: Não foram identificadas desvantagens em mapear desta maneira.

Alternativas: A alternativa seria a criação de uma tabela própria de ‘Doação’, que relacione o item doado a seu doador, mas que aumentaria a memória de armazenamento utilizada, por redundâncias criadas pela recorrência de itens doados.

20. Atributo derivado ‘Idade’

Solução Adotada: O atributo derivado 'Idade' não foi mapeado na tabela 'Usuário', por conta de não ser um dado relevante para qualquer relacionamento da aplicação.

Vantagens: Por ser um atributo anualmente mutável, o não mapeamento garante que o mesmo será calculado com base no ano corrente, evitando possíveis erros.

Desvantagens: O atributo precisa ser calculado em tempo real toda vez que solicitado, consumindo tempo de processamento da aplicação.

Alternativas: A alternativa mais viável seria mapear este atributo na tabela de 'Usuário', porém geraria inconsistência se não fosse atualizada anualmente.

Mudanças relacionadas a segunda entrega

Modelo Relacional

Realizamos algumas alterações em nosso diagrama em relação a segunda entrega. Algumas delas são correções do que tínhamos feito de errado, outras são alterações para melhor funcionamento do sistema.

01. Colaborador
   a. O atributo “CPF/CNPJ” foi-se tornado em chave secundária da entidade Colaborador. Essa correção foi feita pois CPF/CNPJ é único por registro.

02. Escola
   a. O atributo “CodigoMec” foi-se tornado em chave secundária da entidade Escola. Essa correção foi feita pois CodigoMec é único por registro.
   b. O atributo “Administrador” foi-se tornado em chave estrangeira, referente ao atributo Usuário da entidade Administrador.

03. Transação
   a. O atributo “item” foi-se tornado chave primária da entidade Transação, para que não haja uma transação com mesmo item.
   b. Os atributos “usuario” e “data/hora” foram tornados em chave secundária da entidade Transação.

04. Comentário
   a. Os atributos “IDPost”, “Email” e “Data/Hora” foram tornados em chave primária da entidade Comentário.

Discussão sobre os mapeamentos propostos

Realizamos algumas alterações nas discussões sobre os mapeamentos em relação a segunda entrega. Algumas delas são correções do que tínhamos feito de errado, outras são alterações para melhor funcionamento do sistema.

1. **Agregação Transação**
    a. O documento da agregação de Transação foi alterado para se adequar aos ajustes realizados no modelo relacional, referentes às chaves primárias e secundárias da entidade.
2. **Relacionamento 'Avalia' entre 'Usuário' - N:M**
    a. Foi-se removido o trecho "além de não cobrir a participação total presente no relacionamento.".
3. **Relacionamento 'Doa' entre 'Colaborador' e 'Item' - 1:N**
    a. Foi-se adicionada a desvantagem: "possível presença de valores nulos".
4. **Relacionamento 'Verifica' entre 'Administrador' e 'Perfil' - 1:N**
    a. Foi-se adicionada a desvantagem: "possível presença de valores nulos".
5. **Entidade Fraca 'PerfilPúblico' 1:1**
    a. Foi-se adicionada a desvantagem: "maior custo de junção".
    b. Foi-se adicionada a alternativa: "juntar as entidade PerfilPublico e Usuario".
6. **Entidade Fraca 'Comentário'(de 'Faz' e 'Refere') - 1:N**
    a. O documento da entidade fraca Comentário foi alterado para se adequar aos ajustes realizados no modelo relacional, referentes à chave primária da entidade.

Implementação

Esta seção é focada nas especificações da aplicação implementada para o projeto de Inclusão Digital nas Escolas.

A aplicação foi escrita em Python, utilizando a biblioteca Psycopg2 para comunicação com o banco de dados Postgres.

Além disso, foram implementados *scripts* para criação das tabelas (esquema.sql), alimentação inicial da base (dados.sql) e consultas personalizadas (consultas.sql).

Estrutura de Diretórios

A aplicação foi estruturada nos seguintes diretórios:

- **Application**: contém o arquivo com o código do projeto e a função main (app.py);
- **Docker**: contém o arquivo *docker-compose.py*, responsável por fornecer o banco de dados da aplicação;
- **Sql**: contém os arquivos de script para criação (esquema.sql), alimentação (dados.sql) e consulta (consultas.sql) da base de dados.

Scripts de Consulta

Os scripts de consulta a seguir foram criados e executados utilizando o ambiente Oracle.

C1. Consultar as categorias de itens doados em um certo período.

```sql
-- Categorias doadas em determinado período
SELECT COUNT(CATEGORIA.NOME) AS QTDE, CATEGORIA.NOME AS CATEGORIA
FROM TRANSACAO
JOIN ITEM ON (TRANSACAO.ITEM = ITEM.SERIAL)
JOIN CATEGORIA ON (ITEM.CATEGORIA = CATEGORIA.NOME)
WHERE
    TRANSACAO.DATA_HORA BETWEEN
        TO_TIMESTAMP('2014-07-03 00:00:00', 'YYYY-MM-DD HH24:MI:SS.FF') AND
        TO_TIMESTAMP('2023-02-01 00:00:00', 'YYYY-MM-DD HH24:MI:SS.FF')
GROUP BY CATEGORIA.NOME
ORDER BY CATEGORIA.NOME;
```

Código em SQL para a consulta 1.

Nessa consulta selecionamos todos os itens que foram doados, por meio de um *join* entre as tabelas de "TRANSACAO" e "ITEM", e obtemos suas informações de categoria, a partir de outro *join* com a tabela "CATEGORIA". Podemos, então, filtrar os itens doados a partir de seu período de doação, indicado pela operação de *between*, dadas as datas limites desejadas (2014-07-03 e 2023-02-01).
Dessa forma, obtivemos o resultado a seguir, onde a tabela apresenta a quantidade "QTDE" de itens de categoria "CATEGORIA" doados durante o período indicado:

QTDE	CATEGORIA
1	Eletrônicos
1	Livros

Resultado consulta 1.

Essa consulta é obtida a partir das tabelas de exemplo mencionadas, em especial as tabelas "TRANSACAO" e "ITEM", onde assimilamos itens doados por meio de seus códigos, e verificamos suas categorias e datas de transação:

	ITEM	USUARIO	DATA_HORA
1	d0i2dj4u2f	email3@gmail.com	03/07/14 07:49:17,213000000
2	da423j5	email6@gmail.com	23/03/20 05:29:15,666000000

Tabela de transações.

	SERIAL	VALOR	DESCRICAO	CATEGORIA	ALUNO	COLABORADOR
1	d0i2dj4u2f	234	alguma coisa cara	Eletrônicos	123456789	email3@gmail.com
2	da423j5	1	livro de 1 real	Livros	234567891	email3@gmail.com

Tabela de itens.

C2. Consultar doações feitas para escolas com menos de 3 alunos.

```sql
-- Doações para escolas com menos de 3 alunos
SELECT ITEM.CATEGORIA AS CATEGORIA, ESCOLA.NOME AS ESCOLA, ALUNO.NOME AS ALUNO
FROM TRANSACAO
    JOIN ITEM ON (TRANSACAO.ITEM = ITEM.SERIAL)
    JOIN ALUNO ON (ITEM.ALUNO = ALUNO.CPF)
    JOIN ESCOLA ON (ALUNO.ESCOLA = ESCOLA.COMUM)
WHERE
    (SELECT COUNT(*) FROM ALUNO WHERE (ALUNO.ESCOLA = ESCOLA.COMUM)) < 3
ORDER BY
    TRANSACAO.DATA_HORA;
```

Código em SQL para a consulta 2.

Nessa consulta selecionamos todos os itens que foram doados, por meio de um *join* entre as tabelas de “TRANSACAO” e “ITEM”, e obtemos o aluno que recebe o item e sua escola, a partir de outros dois *joins* com as tabelas “ALUNO” e “ESCOLA” . Podemos, então, filtrar as doações, para obtermos apenas as escolas relacionadas que possuem menos de 3 alunos, a partir de uma *nested querry*.
Dessa forma, obtivemos o resultado a seguir, onde a tabela apresenta a categoria “CATEGORIA” do item doado ao aluno “ALUNO” pertencente à escola “ESCOLA”:

CATEGORIA	ESCOLA	ALUNO
Eletrônicos	Nome Escola	Nome Aluno
Livros	Nome Escola	Nome Aluno2

Resultado consulta 2.

Essa consulta é obtida a partir das tabelas de exemplo mencionadas, em especial as tabelas “ALUNO” e “ESCOLA”, onde assimilamos os alunos e suas escolas, e verificamos a quantidade de alunos que estudam na mesma:

	CPF	ESCOLA	NOME	TELEFONE	EMAIL	NASCIMENTO	CEP	NUMERO	COMPLEMENTO
1	123456789	email@gmail.com	Nome Aluno	16912345678	aluno@gmail.com	02/10/00	123456789	7	casa
2	234567891	email@gmail.com	Nome Aluno2	16923456789	aluno2@gmail.com	09/02/87	1234567890	8	apto15
3	345678910	email5@gmail.com	Nome Aluno3	16934567891	aluno3@gmail.com	23/07/04	2345678901	9	casa
4	456789101	email5@gmail.com	Nome Aluno4	16945678910	aluno4@gmail.com	15/11/02	3456789012	10	apto223 bloco04

Tabela de alunos.

	COMUM	CODIGO_MEC	NOME	CPF	TELEFONE	ADMINISTRADOR
1	email@gmail.com	12345678	Nome Escola	123456789	1699999999	email2@hotmail.com
2	email5@gmail.com	23456789	Nome Escola2	234567891	1655555555	email2@hotmail.com

Tabela de escolas.

C3. Consultar o chat entre usuários, envolvendo um post.

```
-- Chat de usuários de determinado post
SELECT
    C.USUARIO1 AS CRIADOR_POST, C.USUARIO2 AS MATCH, C.TEXTO AS CHAT
FROM
    CHAT C
    JOIN POST P ON P.EMAIL = C.USUARIO1;
```

Código em SQL para a consulta 3.

Nessa consulta selecionamos todos os usuários que iniciaram um chat a partir de um post, por meio de um *join* entre as tabelas de “CHAT” e “POST”, a partir do email do usuário criador do post. Obtemos, então, ambos usuários envolvidos no chat, assim como sua conversa em questão.
Dessa forma, obtivemos o resultado a seguir, onde a tabela apresenta os emails do criador do post “CRIADOR_POST” e do outro usuário envolvido no match “MATCH”, e o chat “CHAT” entre eles:

CRIADOR_POST	MATCH	CHAT
email3@gmail.com	email@gmail.com	bla bla bla bla
email@gmail.com	email3@gmail.com	bla bla bla bla bla

Resultado consulta 3.

Essa consulta é obtida a partir das tabelas de exemplo mencionadas, "POST" e "CHAT", onde assimilamos os chats aos posts criados:

	USUARIO1	USUARIO2	DATA_HORA	TEXTO
1	email@gmail.com	email3@gmail.com	02/07/14 06:14:00,742000000	bla bla bla bla bla
2	email3@gmail.com	email@gmail.com	02/07/14 06:14:00,742000000	bla bla bla bla

Tabela de chats.

	ID	EMAIL	DATA_HORA	TITULO	IMAGEM	TIPO
1	129	email@gmail.com	01/07/14 10:06:27,898000000	Alguma coisa	fhasdlufhehp	Doacao
2	129309	email3@gmail.com	22/03/20 04:33:01,122000000	Outra coisa	fdifyasofe	Solicitacao

Tabela de posts.

C4. Consultar as maiores avaliações de todos os usuários.

```
-- Usuário e sua maior avaliação (Left Outer Join)
SELECT
    U.NOME AS NOME, MAX(A.NOTA) AS MAX
FROM
    USUARIO U
    LEFT JOIN AVALIA A ON U.EMAIL = A.USUARIO2
GROUP BY
    U.NOME;
```

Código em SQL para a consulta 4.

Nessa consulta selecionamos todos os usuários e suas avaliações, executando um *left join* entre as tabelas "USUARIO" e "AVALIA", obtendo uma lista de todos os nomes de usuários, seguidos de suas avaliações, filtradas pelo operador *max*, podendo conter usuários ainda não avaliados.
Dessa forma, obtivemos o resultado a seguir, onde a tabela apresenta o nome "NOME" do usuário e sua avaliação máxima "MAX":

NOME	MAX
Nome Pessoa	7
Nome Pessoa4	
Nome Pessoa5	
Nome Pessoa6	
Nome Pessoa9	
Pessoa Nome	
Nome Pessoa3	9
Meu Nome	

Resultado consulta 4.

Essa consulta é obtida a partir das tabelas de exemplo mencionadas, “USUARIO” e “AVALIACAO”, onde assimilamos os usuários às suas avaliações recebidas:

	USUARIO1	USUARIO2	NOTA
1	email@gmail.com	email3@gmail.com	4
2	email3@gmail.com	email@gmail.com	7
3	email4@gmail.com	email3@gmail.com	9

Tabela de avaliações.

	EMAIL	NOME	CARGO	TELEFONE	CEP	NUMERO	COMPLEMENTO
1	meuemail@gmail.com	Meu Nome	Comum	11932585543	923458	1234	lar doce lar
2	email9@gmail.com	Nome Pessoa9	Comum	16916161616	161616161	16	apto16
3	email@gmail.com	Nome Pessoa	Comum	16999999999	111222	1	casa
4	email2@hotmail.com	Pessoa Nome	Administrador	16988888888	10101010	2	ap01
5	email3@gmail.com	Nome Pessoa3	Comum	1677777777	111222333	2	casa
6	email4@gmail.com	Nome Pessoa4	Administrador	1666666666	222333444	3	apto02
7	email5@gmail.com	Nome Pessoa5	Comum	1655555555	333444555	204	casa
8	email6@gmail.com	Nome Pessoa6	Comum	1644444444	444555666	1024	apto22

Tabela de usuários.

C5. Consultar os usuários que foram avaliados por todos os colaboradores.

```sql
-- Usuários avaliados por todos os colaboradores (Sql Nested + Divisão Relacional)
SELECT U.NOME
FROM USUARIO U JOIN AVALIA A ON (A.USUARIO2 = U.EMAIL)
WHERE NOT EXISTS (
    (
        SELECT UA.EMAIL
        FROM USUARIO UA JOIN COMUM C ON (UA.EMAIL = C.USUARIO)
        WHERE UPPER(C.FUNCAO) = 'COLABORADOR'
    )
        MINUS
    (
        SELECT A.USUARIO1
        FROM AVALIA A
        WHERE U.EMAIL = A.USUARIO2
    )
)
GROUP BY U.NOME
```

Código em SQL para a consulta 5.

Nessa consulta selecionamos todos os usuários avaliados, por meio de um *join* entre as tabelas “USUARIO” e “AVALIA”, e os filtramos por meio de uma divisão relacional entre *nested querrys*, onde verificamos a diferença entre a quantidade de usuários com a tag de colaborador (obtida pelo *join* entre as tabelas “USUARIO” e “COMUM”) e a quantidade de usuários que avaliam um único usuário. Assim, todos os usuários que obtiverem a mesma quantidade de avaliadores com a tag de colaborador, em relação ao total de colaboradores, pertencem à tabela resultante.
Dessa forma, obtivemos o resultado a seguir, onde a tabela apresenta o nome “NOME” dos usuários avaliados por todos os colaboradores:

```
NOME
----------------
Nome Pessoa
```

Resultado consulta 5.

Essa consulta é obtida a partir das tabelas de exemplo mencionadas, "AVALIACAO" e "COLABORADOR", onde assimilamos os colaboradores às suas avaliações dadas:

	USUARIO1	USUARIO2	NOTA
1	email@gmail.com	email3@gmail.com	4
2	email3@gmail.com	email@gmail.com	7
3	email4@gmail.com	email3@gmail.com	9
4	email6@gmail.com	email@gmail.com	2

Tabela de avaliações.

	COMUM	TIPO	CPF_CNPJ	DATA_NASCIMENTO	RAZAO_SOCIAL
1	email6@gmail.com	Pessoa Juridica	423795245	22/12/99	Nem Sei
2	email3@gmail.com	Pessoa Fisica	28473295	02/08/00	(null)

Tabela de colaboradores.

Aplicação

Descrição

A interface da aplicação foi projetada em linha de comando, porém com fácil utilização pelo usuário, através de mensagens de ajuda e uma simples interação.
A linguagem de programação utilizada foi Python 3, junto com a biblioteca Psycopg2, para integração com o SGBD Postgresql.

Requisitos

Para executar a aplicação, é necessário estar em um sistema operacional Windows, Linux ou MacOS, com Python 3 instalado e configurado no PATH do sistema.

Além disso, é necessário possuir Docker instalado e configurado na máquina, para que seja possível iniciar e executar o banco de dados.

Por fim, é necessário instalar a biblioteca Psycopg2, através do seguinte comando:
pip3 install psycopg2-binary

Utilização

Para executar a aplicação, primeiramente é preciso iniciar o banco de dados. Para isso, é necessário estar com o **Docker** executando e rodar o seguinte comando, no diretório Docker do projeto:
docker-compose up

Para executar a aplicação, basta rodar o seguinte comando no diretório raiz do projeto:
python3 Application/app.py

Desta forma, a aplicação iniciará e solicitará para que o usuário entre com um dos comandos disponíveis:

- **registrar**: funcionalidade para registrar um novo usuário no sistema. Serão requisitados os dados referentes a entidade **Usuário**, como “email” e “nome”;
- **buscar**: funcionalidade para retornar a média das notas de avaliações de um determinado usuário, informado pelo usuário;
- **sair**: funcionalidade para encerrar a aplicação.

Conclusão

Foi-se concluído que, com um projeto bem estruturado de banco de dados, respeitando as definições semânticas do projeto, desde a construção do MER ao SQL, a construção da aplicação final foi muito facilitada, possibilitando que as funcionalidades primordiais do projeto fossem executadas sem maiores dificuldades.