UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Departamento de Ciências de Computação – SCC

SCC0240 – Banco de Dados

Projeto de Banco de Dados: Tutor Me

Professora: Elaine Sousa

Leonardo Vinicius de Almeida - 10392230

Lucas Xavier Leite - 10783347

São Carlos

2022

Sumário

1 Introdução

A plataforma Tutor Me fornece uma plataforma de tutoria voluntária que promove a integração digital. Nela, usuários podem se cadastrar como tutores voluntários ou alunos, abrangendo diversos tópicos em diferentes níveis de complexidade. O foco da plataforma é a tutoria de conceitos de computação de nível básico, promovendo o aprendizado de como ter o computador e o celular fazer parte das funções básicas do dia a dia, como ensinar alguém a fazer ligações, usar redes sociais, como usar serviços de streaming multimídia, entre outros.

Desde acessar e-mails até usar operações avançadas em planilhas, a ideia da plataforma é criar um ambiente em que qualquer um, não importa o nível de experiência, possa aprender alguma nova habilidade.

2 Modelo Entidade Relacionamento

2.1 Levantamento de Requisitos

A plataforma tem como objetivo o uso por diversos tipos de **usuários**, armazenando suas informações pessoais: **nome, CPF, endereço, telefone, data de nascimento** e **idiomas falados**, assim como dados para uso dentro da plataforma: **e-mail, senha, data do último login, foto do perfil (apenas endereço), data da última edição do perfil, tipo de usuário** e **dados do cartão** para pagamento, contendo **número do cartão, data de validade, código de segurança, CPF** e **nome do titular**.

Identificado pelo atributo **tipo**, todo usuário deve fazer parte de uma, e apenas uma, das seguintes categorias, cada qual contendo informações específicas no âmbito de suas atividades no sistema:

- **Aluno**: sobre eles, guardam-se informações sobre seus **interesses** e seu **tipo**, definindo se ele é **assinante**. Alunos podem participar de **cursos** e dar seu feedback a estes por meio de **avaliações**, com **nota**, e um **comentário** em determinada **data e hora**. Também podem manter **conversas** com outros alunos, sobre as quais armazenamos a **data e hora** em que cada mensagem foi enviada, bem como os **alunos** envolvidos. Também podem **avaliar** os **tutores** de cursos dos quais participam, fornecendo uma **nota** e uma **avaliação** textual. Por fim, alunos podem acessar apenas **recursos** específicos dos cursos dos quais estão participando.

- **Tutor**: os dados básicos são suas **habilidades**. Podem ser de duas categorias específicas, identificadas por atributo **tipo**: **voluntários**, encarregados de prestar serviços de **tutoria** em cursos, e **especialistas**, que são responsáveis por **criar cursos** e **gerenciar recursos**, assim como **fornecer tutoria personalizada** para alunos com acesso a recursos pagos. Enquanto que, sobre o primeiro, é relevante apenas sua **motivação** para prestar o serviço, sobre o segundo, os dados relevantes para o sistema são o **valor da taxa cobrada**, seu **currículo acadêmico** e seus **dados bancários** para pagamento, contendo números da **agência, conta** e **banco**. Sobre suas tutorias personalizadas, é importante também armazenar **data** e **hora** de eventuais **agendamentos**.

- **Administrador**: responsável por **cadastrar tutores**, os dados armazenados são apenas o **nível de acesso** e seu **log de atividades**, contendo alterações feitas dentro do sistema, as quais também podem ser derivadas do próprio banco de dados.

Sobre os **cursos**, que configuram a parte central da plataforma, temos como dados: **código identificador**, que é gerado de forma sistemática pela plataforma, sendo uma combinação da área do conhecimento e um número serial, **título, categoria, descrição, nível de dificuldade** e sua **média de avaliações**, calculada a partir de seu relacionamento com alunos. Além dos relacionamentos já citados com os diversos tipos de usuários, é importante citar também que cursos possuem recursos.

Recursos são elementos para auxiliar a tutoria de voluntários nos cursos e possuem **nome, descrição** e **tipo**, devendo ser categorizados como um dos dois tipos específicos: **pago** ou **comum**. É importante notar também que os recursos de um curso só estão disponíveis aos alunos que participam dele, e que recursos pagos só estão disponíveis para alunos assinantes e alunos que pagaram seu preço único.

Os recursos que podem ser cadastrados e encontrados na plataforma são:

- **Recurso comuns** na forma de **vídeotutoriais** e **guias**, contendo o primeiro informação sobre sua **duração**, e o segundo, sobre o **formato** no qual é oferecido.
- **Recursos pagos** são acessíveis apenas para **alunos assinantes** ou que pagaram seu **preço único**, e podem ser **tutorias personalizadas** ou **atividades práticas**.
- As **atividades práticas** têm uma certa **duração** e são compostas por **questões** que devem ser respondidas pelos **alunos** como parte de seu aprendizado. Cada questão contém uma **pergunta** e uma série de **alternativas**, bem como uma indicação de qual é a **alternativa correta**.
- **Tutorias personalizadas** são recursos que indicam que determinado curso permite que um **aluno** crie um **agendamento** com um **especialista** sobre determinado **assunto**. A plataforma armazena para um determinado **agendamento** os valores de **data e hora**, bem como o **aluno** e **especialista** envolvidos.

2.2 Principais Funcionalidades

- Usuário:
  - Cadastro de usuário;
  - Navegação por recursos e cursos da plataforma;
  - Avaliação de cursos e tutores;
  - Conversar com outros alunos;

  - Participação de tutorias personalizadas;

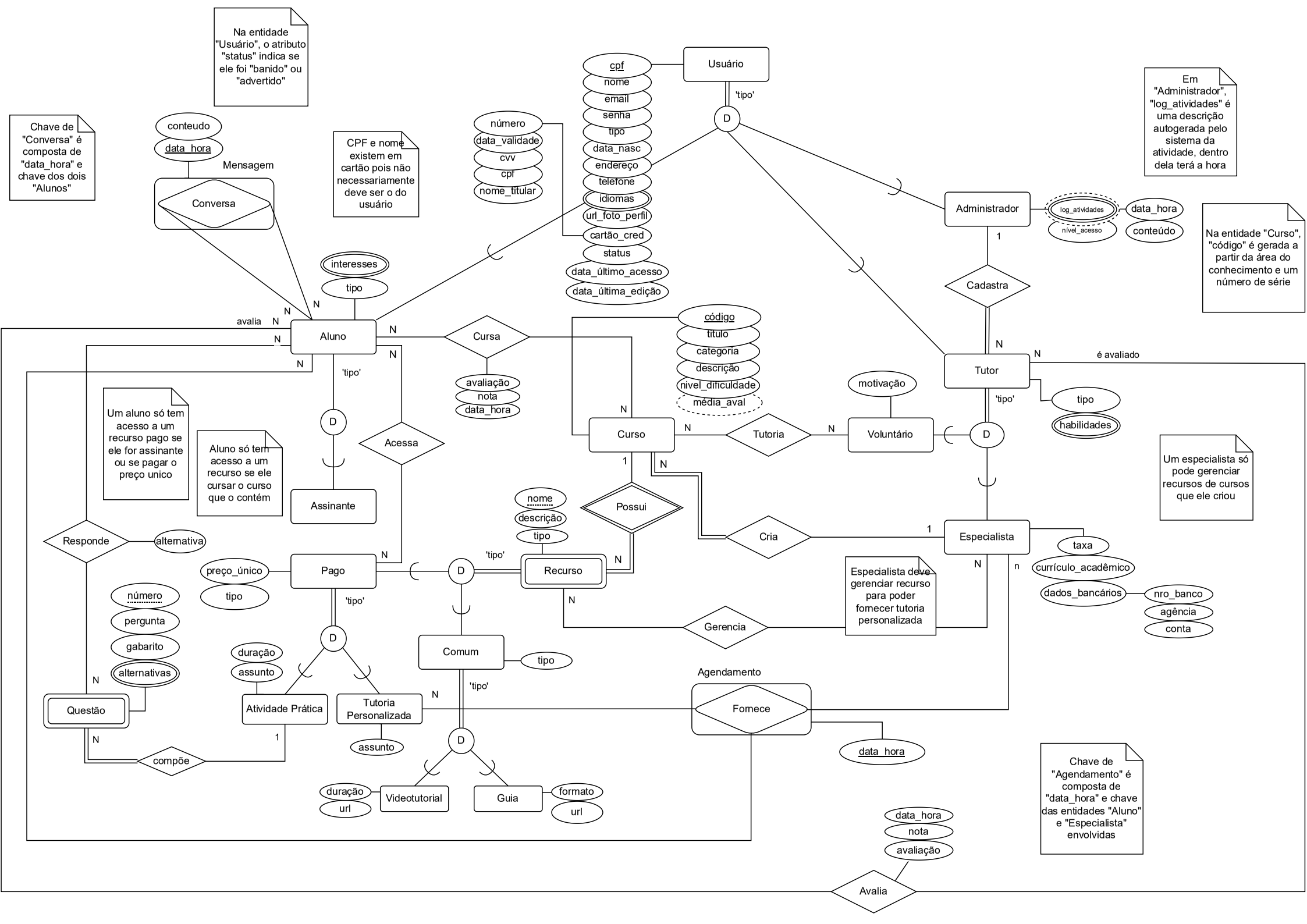
- Tutor:
  - Conexão com usuários para fornecer ajuda em áreas específicas;
  - Ministrar cursos (para tutores especializados);
  - Gerenciar recursos dos cursos (para tutores especializados);

- Administrador:
  - Cadastra/desliga tutores da plataforma;
  - Bloqueia ou libera o acesso de outros perfis à plataforma;
  - Verifica aplicações de tutores voluntários ou especializados;

2.3 Possíveis Inconsistências

- Possível inconsistência entre o recurso gerenciado por um tutor e o curso ao qual pertence: A modelagem do banco não garante que o recurso que um tutor está gerenciando pertence a um curso que esse tutor criou. Essa restrição deve ser feita por meio da aplicação para garantir que um tutor só pode gerenciar recursos de um curso que ele criou.
- Alunos devem ter acesso apenas a recursos pagos de cursos dos quais participam
- Agendamento deve ser fornecido apenas para alunos com acesso a recursos pagos (assinantes ou que pagaram preço único). Podem ser oferecidos por qualquer tutor, independente deste ser responsável pela criação ou gerência do curso, ou recurso.
- Aluno não pode avaliar tutor de curso do qual não participa
- Especialistas não podem gerenciar cursos que não criaram

2.4 M.E.R.


Na entidade "Usuário", o atributo "status" indica se ele foi "banido" ou "advertido"
Chave de "Conversa" é composta de "data_hora" e chave dos dois "Alunos"
CPF e nome existem em cartão pois não necessariamente deve ser o do usuário
Em "Administrador", "log_atividades" é uma descrição autogerada pelo sistema da atividade, dentro dela terá a hora
Na entidade "Curso", "código" é gerada a partir da área do conhecimento e um número de série
Um aluno só tem acesso a um recurso pago se ele for assinante ou se pagar o preço unico
Aluno só tem acesso a um recurso se ele cursar o curso que o contém
Um especialista só pode gerenciar recursos de cursos que ele criou
Especialista deve gerenciar recurso para poder fornecer tutoria personalizada
Chave de "Agendamento" é composta de "data_hora" e chave das entidades "Aluno" e "Especialista" envolvidas
Usuário
cpf
nome
email
senha
tipo
data_nasc
endereço
telefone
idiomas
url_foto_perfil
cartão_cred
status
data_último_acesso
data_última_edição
'tipo'
D
número
data_validade
cvv
cpf
nome_titular
conteudo
data_hora
Mensagem
Conversa
Administrador
log_atividades
nível_acesso
data_hora
conteúdo
1
Cadastra
N
Tutor
é avaliado
tipo
habilidades
'tipo'
interesses
tipo
N
avalia
Aluno
Cursa
avaliação
nota
data_hora
código
titulo
categoria
descrição
nivel_dificuldade
média_aval
Curso
Tutoria
motivação
Voluntário
D
Acessa
Assinante
Possui
nome
descrição
tipo
Recurso
Cria
Especialista
taxa
currículo_acadêmico
dados_bancários
nro_banco
agência
conta
Responde
alternativa
preço_único
tipo
Pago
'tipo'
Gerencia
número
pergunta
gabarito
alternativas
Questão
compõe
duração
assunto
Atividade Prática
Comum
tipo
Tutoria Personalizada
assunto
Agendamento
Fornece
data_hora
duração
url
Videotutorial
Guia
formato
url
data_hora
nota
avaliação
Avalia

3 Alterações na segunda parte do projeto

3.1 Levantamento

A seção de Levantamento de Requisitos teve mudanças significativas para condizer com o MER apresentado e as funcionalidades alteradas.

3.2 M.E.R

- Relacionamento Usuário - Conversa - Usuário gera uma agregação Mensagem com novos atributos: data_hora como uma chave primária
- Atributo id de usuário removido e chave primária movida para email
- Atributo id_curso de curso removido e foi adicionado codigo_curso como chave primária
- Atributo id_recurso removido e chave parcial de recurso passa a ser nome
- Adicionada notação de participação total de recurso no CR com curso
- Entidade Perfil foi desmantelada, e os atributos de suas respectivas especializações foram atribuídos às entidades às quais tinham relação.
- Entidade Usuário é agora Aluno e a nova entidade Usuario é a entidade geral de Aluno, Tutor e Administrador.
- Removidos todos os id sintéticos.
- Tutor/Pago agora é Especialista
- Tutor/Voluntario agora é Voluntario
- A relação Tutor/pago (Especialista) **Fornece** Tutoria Personalizada que tinha atributo Agendamento Agora resulta em uma agregação Agendamento com atributos Horário e Data
- Usuario(Aluno) agora se relaciona com Recurso. A restrição de recursos pagos é tratada na implementação.
- A especialização disjunta de recurso agora requer participação total
- A chave primária de Usuario é CPF, sendo que email é chave fraca

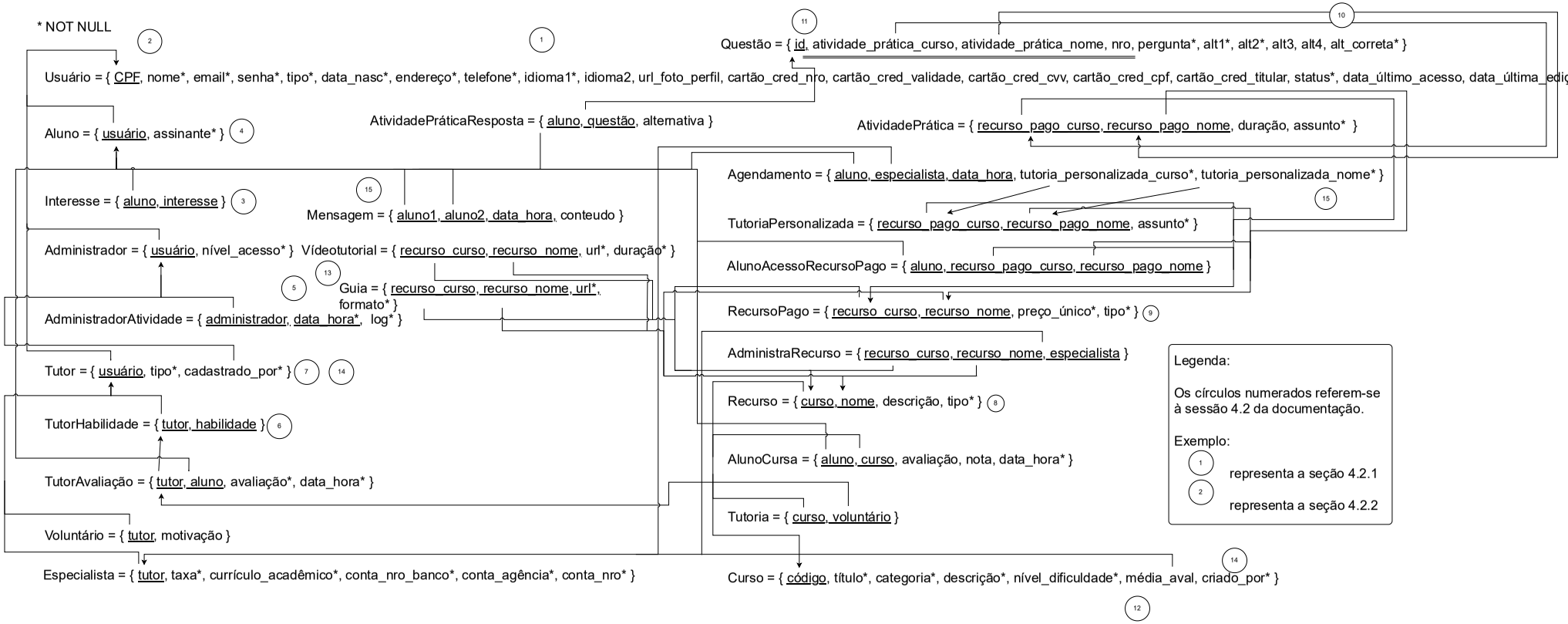
Outras possíveis alterações (não realizadas):

- Identificador sintético em **mensagem** removido.
- Atributo de avaliação da entidade aluno passado para o relacionamento Cursa entre Aluno e Curso.
- Atributo cursos ativos de Aluno removido.
- Relacionamento ternário gerencia entre administrador curso e aluno removido
- Relacionamento aluno acessa recurso removido. E para garantir que aluno pago tenha acesso a recurso pago, foi adicionado novo relacionamento.
- **E**specializações de recurso pago e comum passaram a ser disjuntos
- Participação total em tutor na relação Admin Cadsatra Tutor
- Nova agregação criada: Agendamento, ao invés de uma relação apenas.
- Log_atividades agora é multivalorado e derivado.
- Adicionado relacionamento Aluno Avalia Tutor
- Adicionados atributos de especialização nas entidades genéricas
- Participação em Curso criado por especialista.
- Como um recurso só está dentro de um curso, foi removido o atributo autor de recurso
- atributo duração movido de Recurso para Tutorial em vídeo
- Removidas especializações Certificações e Suporte Técnico de Recurso Pago
- Especialização Atividade prática movida da especialização Comum de Recurso para Pago.
- Adicionada questão como entidade fraca de atividade prática
- adicionados relacionamentos: contem (atividade pratica contêm questões) e responde (aluno responde questões)
- adicionada participação total de especialista em agendamento
- removido especialização fórum de recurso comum
- adicionada participação total em recurso pago e recurso comum na especialização

- adicionados novos atributos às especializações de recurso comum Tutorial em Vídeo e Guia
- entidade Tutoria em Vídeo renomeada para Videotutorial
- notação de especialização consertada, especificando o atributo de especialização

4 Modelo Relacional

4.1 Esquema Relacional


* NOT NULL
Usuário = { CPF, nome*, email*, senha*, tipo*, data_nasc*, endereço*, telefone*, idioma1*, idioma2, url_foto_perfil, cartão_cred_nro, cartão_cred_validade, cartão_cred_cvv, cartão_cred_cpf, cartão_cred_titular, status*, data_último_acesso, data_última_edição }
Aluno = { usuário, assinante* }
Interesse = { aluno, interesse }
Administrador = { usuário, nível_acesso* }
AdministradorAtividade = { administrador, data_hora*, log* }
Tutor = { usuário, tipo*, cadastrado_por* }
TutorHabilidade = { tutor, habilidade }
TutorAvaliação = { tutor, aluno, avaliação*, data_hora* }
Voluntário = { tutor, motivação }
Especialista = { tutor, taxa*, currículo_acadêmico*, conta_nro_banco*, conta_agência*, conta_nro* }
AtividadePráticaResposta = { aluno, questão, alternativa }
Mensagem = { aluno1, aluno2, data_hora, conteudo }
Vídeotutorial = { recurso_curso, recurso_nome, url*, duração* }
Guia = { recurso_curso, recurso_nome, url*, formato* }
Questão = { id, atividade_prática_curso, atividade_prática_nome, nro, pergunta*, alt1*, alt2*, alt3, alt4, alt_correta* }
AtividadePrática = { recurso_pago_curso, recurso_pago_nome, duração, assunto* }
Agendamento = { aluno, especialista, data_hora, tutoria_personalizada_curso*, tutoria_personalizada_nome* }
TutoriaPersonalizada = { recurso_pago_curso, recurso_pago_nome, assunto* }
AlunoAcessoRecursoPago = { aluno, recurso_pago_curso, recurso_pago_nome }
RecursoPago = { recurso_curso, recurso_nome, preço_único*, tipo* }
AdministraRecurso = { recurso_curso, recurso_nome, especialista }
Recurso = { curso, nome, descrição, tipo* }
AlunoCursa = { aluno, curso, avaliação, nota, data_hora* }
Tutoria = { curso, voluntário }
Curso = { código, título*, categoria*, descrição*, nível_dificuldade*, média_aval, criado_por* }
Legenda:
Os círculos numerados referem-se à sessão 4.2 da documentação.
Exemplo:
1 representa a seção 4.2.1
2 representa a seção 4.2.2

4.2 Justificativas do mapeamento

4.2.1 Atributo multivalorado idioma da entidade Usuário

Solução adotada: Foram adicionadas duas colunas na tabela da entidade Usuário para armazenar o atributo idiomas.

Vantagens: A vantagem desse mapeamento é uma menor complexidade na entidade usuário. Maior eficiência nas consultas por não necessitar de junções adicionais.

Desvantagens: A desvantagem desse mapeamento é a limitação para os usuários no número de idiomas cadastrados no sistema.

Solução alternativa: Criação de uma tabela de relacionamento para os usuários e os idiomas.

4.2.2 Especialização da entidade Usuário

Solução adotada: Para mapear a especialização da entidade Usuário foram criadas outras três tabelas: Aluno, Tutor e Administrador, tendo como chave primaria a chave estrangeira do Usuário.

Vantagens: Garante a separação das funções dos alunos, tutores e administradores, que possuem papéis consideravelmente distintos, facilitando a criação dos relacionamentos e garantindo uma maior consistência.

Desvantagens: Adição de redundância de dados em disco ao armazenar duas vezes o CPF dos usuários. Outra desvantagem é a necessidade de criar um usuário antes de criar um aluno, tutor ou administrador no banco. Não garante a disjunção e especialização total.

Solução alternativa: Armazenar as informações de aluno, tutor e administrador na tabela usuário e fazer verificações na aplicação para a utilização de funcionalidades como a criação de um curso.

4.2.3 Atributo multivalorado interesses da entidade Aluno

Solução adotada: Criação de uma tabela separada com os interesses dos alunos e a utilização de uma chave composta pelo interesse e a chave estrangeira do aluno.

Vantagens: A possibilidade de um aluno possuir uma quantidade indeterminada de interesses.

Desvantagens: Aumento na complexidade da entidade aluno, criando mais uma tabela para o relacionamento e diminuindo a eficiência do banco caso seja necessário saber todos os

interesses de determinado aluno.

Solução alternativa: Limitar a quantidade de interesses de cada aluno, colocando cada um de seus interesses como um atributo da entidade aluno.

4.2.4 Especialização da entidade Aluno

Solução adotada: Para mapear a especialização da entidade Aluno foi adicionado um atributo assinante na sua tabela.

Vantagens: Menor complexidade no banco e maior eficiência nas consultas dos alunos.

Desvantagens: Nesse caso, como o assinante será um atributo booleano, e o CEE não possui atributos/relacionamentos específicos, não há desvantagens para essa abordagem em relação as alternativas.

Solução alternativa: Criação de tabelas separadas para cada tipo de aluno e separar as suas funcionalidades.

4.2.5 Atributo derivado e multivalorado log_atividades da entidade Administrador

Solução adotada: Para o atributo log_atividades da entidade Administrador foi criada uma tabela separada utilizando como chave estrangeira a chave primaria do administrador e uma coluna separada para o armazenamento do log.

Vantagens: A possibilidade de guardar quantos logs de atividades forem necessários no sistema.

Desvantagens: Aumento na complexidade na consulta dos logs guardados por um administrador.

Solução alternativa: Armazenar o log como atributo de administrador limitando a quantidade de logs de atividades a serem armazenados no sistema.

4.2.6 Atributo multivalorado habilidades da entidade Tutor

Solução adotada: Criação de uma tabela separada com as habilidades dos tutores e a utilização de uma chave composta pela habilidade e a chave estrangeira do tutor.

Vantagens: A possibilidade de um tutor possuir quantas habilidades ele quiser.

Desvantagens: Aumento na complexidade da entidade tutor, criando mais uma tabela para o relacionamento e diminuindo a eficiência do banco caso seja necessário saber todas as

habilidades de um dado tutor.

Solução alternativa: Limitar a quantidade de habilidades de cada tutor, colocando cada uma de suas habilidades como um atributo da entidade tutor.

4.2.7 Especialização da entidade Tutor

Solução adotada: Criação de tabelas separadas para as especializações dos tutores e separar as suas funcionalidades, mantendo a tabela genérica que terá um atributo tipo para indicar a especialização.

Vantagens: Menor complexidade no banco e maior eficiência nas consultas dos tutores.

Desvantagens: Necessidade de realizar verificações dos tipos de tutores na aplicação para garantir que cada tutor possua apenas as funcionalidades de sua especialização. Não há como garantir a disjunção e especialização total (deve ser feito a nível de implementação/aplicação).

Solução alternativa: Criação das tabelas de especialização apenas ou manter apenas a tabela genérica adicionando como colunas todos os atributos das especializações.

4.2.8 Especialização da entidade Recurso

Solução adotada: Para modelar a especialização da entidade recurso foi mantida a tabela referente a entidade genérica Recurso (com um atributo tipo) e feita a criação de uma nova tabela para a especialização.

Vantagens: Garante a separação do acesso aos recursos pagos e recursos gratuitos.

Desvantagens: Redundância de armazenamento nas informações das chaves na tabela recursos pagos e diminuição na eficiência das consultas no banco pelos recursos pagos. Não há como garantir a disjunção e especialização total (deve ser feito ao nível de implementação/aplicação).

Solução alternativa: Armazenar todos os dados da entidade recurso pago na tabela recurso e criação de um atributo tipo para diferenciar os recursos pagos dos recursos gratuitos.

4.2.9 Especialização da entidade Recurso Pago

Solução adotada: Para modelar a especialização da entidade recurso pago foi mantida a tabela de Recurso Pago (com um atributo tipo) que é a entidade genérica e foram criadas duas novas tabelas, uma para a entidade atividade pratica e outra para a tutoria personalizada, separando as informações referentes a cada uma das entidades.

Vantagens: Melhor eficiência no armazenamento dos dados, evitando muitas colunas vazias quando armazenar dados de uma das duas especializações.

Desvantagens: Maior custo de armazenamento dos recursos pagos e menor eficiência nas consultas de cada recurso. Não garante a disjunção e especialização total.

Solução alternativa: Armazenar todas as especializações em uma tabela e fazer a verificação do tipo de recurso na aplicação.

4.2.10 Atributo multivalorado alternativas da entidade Questão

Solução adotada: Para o atributo alternativas da entidade questão foram criados quatro colunas: alt1, alt2, alt3, alt4 na tabela questão, limitando em quatro o número de alternativas em cada questão.

Vantagens: Menor complexidade na entidade e maior eficiência nas consultas das questões.

Desvantagens: Limitação no número de alternativas para as questões.

Solução alternativa: Criação de uma tabela separada para armazenar cada uma das alternativas de uma dada questão

4.2.11 Chave primária da entidade Questão

Solução adotada: Criação de um id sintético como chave primária.

Vantagens: Aumento na eficiência das buscas da entidade questão e diminuição no número de atributos trazidos ao utilizar uma chave estrangeira.

Desvantagens: Perda da semântica com uso do identificador sintético. Necessidade de junções adicionais quando houver a necessidade de recuperar informações referentes aos atributos da 'antiga' chave primaria em questão.

Solução alternativa: Utilizar o número da questão e a chave estrangeira da atividade pratica como chave primária.

4.2.12 Atributo derivado media_aval (Curso)

Solução adotada: Criação do atributo na tabela Curso.

Vantagens: Eficiência na busca.

Desvantagens: Custo na inserção ao ter que computar a média a cada inserção na tabela Curso.

Solução alternativa: Não mapear esse atributo em nenhuma tabela, e calcular a média por meio de consultas sempre que necessário.

4.2.13 Especialização da Entidade Recurso Comum

Solução adotada: Criação de duas tabelas adicionais referentes as especializações.

Vantagens: Maior eficiência nas buscas ao descartar a necessidade de junções para obter dados das especializações. Melhor desempenho em consultas específicas, pois é possível otimizar consultas específicas que envolvam apenas uma das especializações. Flexibilidade no tratamento das especializações, pois é possível adicionar novas especializações no futuro sem afetar a estrutura da entidade genérica.

Desvantagens: Aumenta a replicação de dados relacionados a entidade Genérica. Aumenta a complexidade de compreensão do modelo de dados. Possibilidade de inconsistências nos dados, se as atualizações não forem realizadas corretamente em todas as tabelas envolvidas, pode ocorrer a falta de sincronia entre os dados.

Solução alternativa: Manter a tabela da entidade genérica com um atributo tipo e criação das tabelas de especialização referenciando pela chave primaria a tabela genérica.

4.2.14 Relacionamentos 1:N

Solução adotada: Mapeamento com chave estrangeira na entidade com cardinalidade 1, referenciando a entidade com cardinalidade N.

Vantagens: Garantia de integridade referencial entre as entidades e simplicidade na implementação, facilitando o entendimento da estrutura. Eficiência nas consultas, pois há menos junções e tabelas necessárias.

Desvantagens: Nesse caso, não há desvantagens em relação à solução alternativa.

Solução alternativa: Uso de tabelas intermediárias e chave composta para relacionar as entidades.

4.2.15 Agregações Mensagem e Agendamento

Solução adotada: Criação de uma única tabela para a agregação, associando as entidades envolvidas no relacionamento através do uso de suas chaves primárias para composição da chave da nova tabela, juntamente com um campo adicional para especificar a hora.

Vantagens: O mapeamento em uma única tabela é consideravelmente mais simples, trazendo benefícios tanto na manutenção e consistência dos dados quanto na complexidade de consulta, diminuindo a quantidade de junções necessárias para recuperar os dados armazenados.

Desvantagens: Nesse caso, não há desvantagens em relação à solução alternativa, visto que esta apenas traz mais complexidade, sem benefícios significativos.

Solução alternativa: Além da tabela da agregação, criação de uma tabela adicional para o relacionamento.

5 Alterações na terceira parte do projeto

5.1 Correções no MER

- Adicionado atributo conteúdo na agregação Mensagem
- Adicionado atributo alternativa no relacionamento "aluno responde questão"
- Consertada notação no agendamento
- Conforme sugerido, transformado campo log_atividade em composto com conteúdo e data_hora
- Adicionado campo assunto na entidade TutoriaPersonalizada
- Adicionado campo data_hora no relacionamento "aluno avalia tutor"

5.2 Correções no Relacional

- Consertada formatação das chaves e adicionados line jumps para melhor visualização
- Entidade AdminitradorAtividade ajustada de acordo com novas alterações no MER
- Entidade Mensagem ajustada de acordo com novas alterações no MER
- Adicionado campo data_hora no mapeamento da entidade TutorAvaliação
- Adicionada notação de chave estrangeira na entidade Questão
- Campo tipo da entidade Recurso como NOT NULL
- Adicionado campo nota na entidade AlunoCursa
- Adicionada notação de chave estrangeira no campo aluno da entidade AlunoAcessoRecursoPago
- Adicionado atributo url em videotutorial e guia

5.3 Correções nas Justificativas

- Correção das justificativas:
  - 4.2.1
  - 4.2.2
  - 4.2.4
  - 4.2.5
  - 4.2.7
  - 4.2.8
  - 4.2.9
  - 4.2.11
- Novas justificativas:
  - 4.2.12
  - 4.2.13
  - 4.2.14
  - 4.2.15

6 Aplicação

De acordo com a proposta, a aplicação desenvolvida trata-se de um *shell* interativo no qual o usuário consegue gerenciar a base de dados da plataforma, realizando consultas e inserções simples através de uma interface de linha de comando.

6.1 Instruções de uso

Para o desenvolvimento da aplicação, foi utilizada a linguagem de programação **Python** e o SGBD **PostgreSQL**, assim como o adaptador **Psycopg2**. Para executar a aplicação, é necessário instalar os pacotes listados no arquivo "requirements.txt"através do gerenciador de pacotes **pip** utilizando o comando:

```
pip install --user -r requirements.txt
```

Após instalar os pacotes e criar uma base de dados no PostgresSQL, basta configurar as variáveis de ambiente no arquivo ".env.example", renomeá-lo para ".env", e em seguida executar* o script **main.py** a partir da raiz da estrutura de pastas do projeto:

```
python app/main.py
```

Nota: apenas versões de Python superiores a 3.10 são suportadas.

Para facilitar o uso e configuração do PostgreSQL e da aplicação, também está disponível no repositório do projeto os arquivos de configuração de um container **Docker Compose** e de um **Dockerfile** para a imagem da aplicação, com todos os pacotes necessários. Para utilizá-lo, basta executar, na pasta raiz do projeto:

```
docker-compose up -d
docker attach
```

É válido ressaltar que a aplicação foi testada em ambientes GNU/Linux, porém ela também deve funcionar em ambientes Windows e macOS.

6.2 Funcionalidades

Como é possível observar no menu inicial da aplicação, as seguintes funcionalidades foram implementadas:

6.2.1 Listar todos os cursos

(courses.py)

```
    def list(connection):
    '''
    Retrieves and prints all courses from the database.

    Parameters
    ----------
    connection : psycopg2.extensions.connection
        The database connection.
    '''

    try:
        query = '''
            SELECT
```

```
            codigo AS "C'odigo",
            titulo AS "T'itulo",
            categoria AS "Categoria",
            nivel_dificuldade AS "N'ivel de dificuldade",
            media_aval AS "M'edia de avaliaçoes",
            SUBSTRING(descricao, 1, 20) AS "Descriçao",
            criado_por AS "Tutor respons'avel"
            FROM curso
            '''

        with connection.cursor() as cursor:
            cursor.execute(query)

            if cursor.rowcount > 0:
                print('Cursos:')
                print(from_db_cursor(cursor))
            else:
                print('Ainda nao h'a cursos cadastrados.')
    except Exception as e:
        connection.rollback()
        print_error(e)

    press_enter_message()
```

6.2.2 Listar todos os alunos

(students.py)

```
    def list(connection):
        '''
        Retrieves and prints all students from the database.

        Parameters
        ----------
        connection : psycopg2.extensions.connection
            The database connection.
        '''

```

```
        try:
            query = '''
                SELECT
                U.cpf AS "CPF",
                U.nome AS "Nome",
                U.email AS "E-mail",
                CASE WHEN A.assinante IS TRUE THEN 'Sim'
                   ELSE 'Nao' END AS "Assinante"
                FROM aluno AS A JOIN usuario AS U ON A.
                   usuario = U.cpf
                '''

            with connection.cursor() as cursor:
                cursor.execute(query)

                if cursor.rowcount > 0:
                    print('Alunos:')
                    print(from_db_cursor(cursor))
                else:
                    print('Ainda nao h'a alunos cadastrados.'
                       )
        except Exception as e:
            connection.rollback()
            print_error(e)

        press_enter_message()
```

6.2.3 Listar alunos de um curso

(courses.py)

```
    def list_students(connection, course):
    '''
    Retrieves and prints all students enrolled in a specific
        course.

    Parameters
    ----------
```

```
    connection : psycopg2.extensions.connection
        The database connection.
    course : str
        The course code.
    '''

    try:
        query = '''
            SELECT
            U.cpf AS "CPF",
            U.nome AS "Nome",
            U.email AS "E-mail",
            CASE WHEN A.assinante IS TRUE THEN 'Sim' ELSE 'N
               ao' END AS "Assinante",
            AC.nota AS "Nota",
            AC.data_hora AS "Data de in'icio"
            FROM aluno_cursa AS AC
            JOIN aluno AS A ON AC.aluno = A.usuario
            JOIN usuario AS U ON A.usuario = U.cpf
            WHERE AC.curso = %s
            '''

        with connection.cursor() as cursor:
            cursor.execute(query, (course,))

            if cursor.rowcount > 0:
                print(f'Alunos do curso {course}:')
                print(from_db_cursor(cursor))
            else:
                print('Ainda nao h'a alunos cadastrados neste
                    curso. Nao se esqueça de verificar se o
                   curso existe.')
    except Exception as e:
        connection.rollback()
        print_error(e)

    press_enter_message()
```

6.2.4 Buscar curso

(courses.py)

```
1   def search(connection, key):
2   '''
3   Searches for a course in the database based on a key.
4
5   Parameters
6   ----------
7   connection : psycopg2.extensions.connection
8       The database connection.
9   key : str
10      The key to search for (course code or title).
11  '''
12
13  try:
14      query = '''
15          SELECT
16          codigo AS "C'odigo",
17          titulo AS "T'itulo",
18          categoria AS "Categoria",
19          nivel_dificuldade AS "N'ivel de dificuldade",
20          media_aval AS "M'edia de avaliaçoes",
21          SUBSTRING(descricao, 1, 20) AS "Descriçao",
22          criado_por AS "Tutor respons'avel"
23          FROM curso WHERE codigo = %s OR titulo = %s
24          '''
25
26      with connection.cursor() as cursor:
27          cursor.execute(query, (key, key))
28
29          if cursor.rowcount > 0:
30              print(f'{cursor.rowcount} resultado(s) para
                    a busca pelo curso "{key}":')
31              print(from_db_cursor(cursor))
32          else:
```

```
33                print(f'Nao foi poss'ivel encontrar nenhum
                      curso "{key}".')
34    except Exception as e:
35        connection.rollback()
36        print_error(e)
37
38    press_enter_message()
```

6.2.5 Inserir aluno em um curso

(courses.py)

```
1     def insert_student(connection, course, student):
2         '''
3         Inserts a student into a course.
4
5         Parameters
6         ----------
7         connection : psycopg2.extensions.connection
8             The database connection.
9         course : str
10            The course code.
11        student : str
12            The student's identifier.
13        '''
14
15        query = 'INSERT INTO aluno_cursa(aluno, curso,
              data_hora) VALUES (%s, %s, %s)'
16
17        try:
18            with connection.cursor() as cursor:
19                cursor.execute(query, (student, course,
                      datetime.now()))
20            print(f'\nAluno {student} inserido no curso {
                  course} com sucesso.\n')
21        except psycopg2.errors.UniqueViolation as e:
22            connection.rollback()
```

```
23              print_error('o aluno {student} j'a est'a cursando
                    {course}.')
24          except psycopg2.errors.ForeignKeyViolation as e:
25              connection.rollback()
26              constraint_name = e.diag.constraint_name
27
28              print_error(f'violaçao de chave estrangeira ({
                    constraint_name})')
29
30              error_messages = {
31                  'fk_aluno_cursa_aluno': 'o aluno
                        especificado nao existe.',
32                  'fk_aluno_cursa_curso': 'o curso
                        especificado nao existe.'
33              }
34
35              print_error(f'{error_messages[constraint_name]}
                    Verifique os dados informados.')
36          except Exception as e:
37              connection.rollback()
38              print_error(e)
39
40          press_enter_message()
```

6.2.6 Inserir aluno

(students.py)

```
1       def create(connection, student):
2       '''
3       Creates a new student in the database.
4
5       Parameters
6       ----------
7       connection : psycopg2.extensions.connection
8           The database connection.
9       student : dict
10          The student information.
```

```
    Returns
    -------
    bool
        True if the student was created successfully, False otherwise.
    '''

    insert_user_query = '''
        INSERT INTO usuario (cpf, nome, email, senha, tipo, data_nasc,
            endereco, telefone, idioma1, status)
        VALUES (%(cpf)s, %(nome)s, %(email)s, %(senha)s, %(tipo)s,
            %(data_nasc)s, %(endereco)s, %(telefone)s, %(idioma1)s, %(status)s)
        '''

    insert_student_query = 'INSERT INTO aluno(usuario, assinante) VALUES (%s, %s)'

    try:
        with connection.cursor() as cursor:
            cursor.execute(insert_user_query, student)
            cursor.execute(insert_student_query, (student['cpf'], False))

        print('\nAluno inserido com sucesso.\n')
        press_enter_message()

        return True
    except psycopg2.errors.UniqueViolation as e:
        connection.rollback()
        print_error('j'a existe um usu'ario com esse CPF.')
    except psycopg2.errors.CheckViolation as e:
        connection.rollback()

        constraint_name = e.diag.constraint_name
```

```
42          print_error(f'violaçao de check ({constraint_name})'
                )
43
44          error_messages = {
45              'cpf_formato': 'formato de CPF inv'alido.',
46              'cartao_cred_cpf_formato': 'formato do n'umero de
                    cartao de cr'edito inv'alido.',
47              'cartao_cred_val_formato': 'formato de validade
                    do cartao de cr'edito inv'alido.',
48              'telefone_formato': 'formato de n'umero de
                    telefone inv'alido.',
49              'email_formato': 'formato de e-mail inv'alido.'
50          }
51
52          print_error(f'{error_messages[constraint_name]}
                Verifique os dados informados.')
53      except psycopg2.errors.InvalidDatetimeFormat as e:
54          connection.rollback()
55          print_error('formato de data inv'alido. A data deve
                estar no formato DD/MM/YYYY.')
56      except psycopg2.errors.StringDataRightTruncation as e:
57          connection.rollback()
58          print(e)
59          print_error('tamanho inv'alido. Verifique os dados
                informados.')
60      except psycopg2.Error as e:
61          connection.rollback()
62          print_error(e.pgcode)
63          print_error(e)
64      except Exception as e:
65          connection.rollback()
66          print_error(e)
67
68      press_enter_message()
69      return False
```

6.3 Consultas

6.3.1 C1. Número de agendamentos de tutores por período de tempo e o número de alunos atendidos no período

Essa consulta retorna o número de agendamentos de tutores em um determinado período de tempo, agrupados por mês, ano e tutor. Além disso, também retorna o número de alunos atendidos no período.

```
SELECT
  TO_CHAR(A.data_hora, 'Month') AS "Mes",
  TO_CHAR(A.data_hora, 'YYYY') AS "Ano",
  COUNT(*) AS "Agendamentos",
  COUNT(DISTINCT A.aluno) AS "Alunos Atendidos"
FROM
  agendamento AS A
  JOIN especialista AS E ON A.especialista = E.tutor
GROUP BY
  TO_CHAR(A.data_hora, 'Month'),
  TO_CHAR(A.data_hora, 'YYYY'),
  A.especialista;
```

Essa consulta pode ser utilizada para analisar a demanda de agendamentos de tutores ao longo do tempo e verificar quantos alunos estão sendo atendidos. Isso pode ajudar a identificar padrões de procura por tutores, avaliar a eficiência do sistema de agendamento e tomar decisões relacionadas à alocação de recursos e tutores.

6.3.2 C2. Média de avaliação de tutores de cursos que contenham recurso pago tutoria personalizada e sejam cursados por mais de 3 alunos. Seleciona também a média de avaliação dos cursos

Essa consulta calcula a média de avaliação dos tutores que ministram cursos específicos. Apenas são considerados os cursos que possuem o recurso pago de tutoria personalizada e são cursados por mais de 3 alunos. Além disso, também é calculada a média de avaliação dos cursos em si.

```
SELECT
  AVG(TA.avaliacao) AS "Media de Avaliaçao dos Tutores",
```

```
3       AVG(C.media_aval) AS "Media de Avaliaçao dos Cursos",
4       C.codigo AS "Codigo do Curso"
5     FROM
6       tutor_avaliacao AS TA
7       JOIN tutor AS T ON TA.tutor = t.usuario
8       JOIN voluntario AS V ON T.usuario = V.tutor
9       JOIN tutoria AS TR ON TR.voluntario = V.tutor
10      JOIN curso AS C ON C.codigo = TR.curso
11      JOIN aluno_cursa AS AC ON C.codigo = AC.curso
12      JOIN recurso_pago AS RP ON C.codigo = RP.recurso_curso
13      AND RP.tipo = 'tutoria_personalizada'
14    GROUP BY
15      C.codigo
16    HAVING
17      COUNT(AC.aluno) > 3;
```

Essa consulta pode ser útil para identificar os tutores e cursos que têm as melhores avaliações por parte dos alunos. Isso pode auxiliar na tomada de decisões sobre quais tutores ou cursos promover, melhorar ou investir mais recursos. Também pode ser utilizado como critério para a seleção de tutores em determinados cursos.

6.3.3 C3. Dificuldade média de todos os cursos, e se houver, média das avaliações dos cursos, por categoria, assim como a idade média dos alunos que cursam esses cursos

Essa consulta calcula a dificuldade média de todos os cursos, bem como a média das avaliações dos cursos por categoria. Também é calculada a idade média dos alunos que estão cursando esses cursos.

```
1     SELECT
2       C.categoria AS "Categoria",
3       TO_CHAR(AVG(C.nivel_dificuldade), 'FM999999999.00') AS
            "Dificuldade Media",
4       TO_CHAR(AVG(AGE(CURRENT_DATE, U.data_nasc)), 'YY') AS
          "Idade Media",
5       TO_CHAR(AVG(C.media_aval), 'FM999999999.00') AS "Media
            de Avaliacoes"
6     FROM
```

```
      usuario AS U
      JOIN aluno AS A ON U.cpf = A.usuario
      LEFT JOIN aluno_cursa AS AC ON A.usuario = AC.aluno
      RIGHT JOIN curso AS C ON AC.curso = C.codigo
    GROUP BY
      C.categoria
    HAVING
      AVG(C.nivel_dificuldade) < 5
    ORDER BY
      C.categoria;
```

Essa consulta pode ser útil para analisar a dificuldade e as avaliações dos cursos oferecidos, tanto de forma geral quanto por categoria específica. Além disso, a idade média dos alunos pode fornecer insights sobre o público-alvo dos cursos e auxiliar na criação de estratégias de marketing ou no desenvolvimento de cursos direcionados a determinadas faixas etárias.

6.3.4 C4. Média de avaliação dos cursos avaliados pelo usuário, considerando apenas avaliações com nota entre 3 e 8, e a média de avaliação dos cursos cursados pelo usuário

Essa consulta calcula a média de avaliação dos cursos avaliados por cada usuário, considerando apenas as avaliações com nota entre 3 e 8. Também é calculada a média de avaliação dos cursos que o usuário cursou.

```
    SELECT
      U.nome AS "Aluno",
      AVG(
        CASE
          WHEN AC.avaliacao BETWEEN 3
          AND 8 THEN AC.avaliacao
        END
      ) AS "Media de Avaliaçao de Cursos Pelo Usuario",
      AVG(C.media_aval) AS "Media de Avaliaçao dos Cursos
         Cursados Pelo Usuario"
    FROM
      usuario AS U
      JOIN aluno AS A ON U.cpf = A.usuario
      LEFT JOIN aluno_cursa AS AC ON A.usuario = AC.aluno
```

```
14        JOIN curso AS C ON AC.curso = C.codigo
15      GROUP BY
16        U.nome
17      ORDER BY
18        U.nome;
```

Essa consulta pode ser utilizada para analisar as avaliações dos cursos feitas por cada usuário, levando em consideração apenas as notas relevantes (entre 3 e 8). Isso pode ajudar a identificar o nível de satisfação dos alunos e também comparar as avaliações dos cursos com as avaliações feitas pelos alunos que realmente cursaram esses cursos.

6.3.5 C5. Alunos que cursam todos os cursos tutorados pelo tutor voluntário Tutor 9

Essa consulta seleciona os alunos que estão cursando todos os cursos tutorados pelo tutor voluntário de CPF "123.456.789-18", configurando uma divisão relacional.

```
1   SELECT A.usuario
2   FROM aluno AS A
3   WHERE NOT EXISTS (
4     SELECT T.curso
5     FROM tutoria AS T
6     WHERE T.voluntario = '123.456.789-18'
7       AND T.curso NOT IN (
8         SELECT AC.curso
9         FROM aluno_cursa AS AC
10        WHERE AC.aluno = A.usuario
11      )
12  );
```

Essa consulta pode ser útil para identificar os alunos que estão engajados nos cursos tutorados por um tutor específico. Isso pode ser utilizado para reconhecer o desempenho dos alunos, monitorar seu progresso e oferecer suporte adicional, se necessário. Além disso, também pode ser útil para avaliar a eficácia do tutor em ajudar os alunos a completarem os cursos.

7 Conclusão

Durante o desenvolvimento do projeto, o grupo teve a oportunidade de compreender a diferença entre a parte lógica do design e a implementação real de um banco de dados. Ao final do projeto, percebemos que muitos conceitos foram reconsiderados e adaptados à medida que o sistema era desenvolvido, mostrando como o processo de design lógico pode evoluir mesmo durante a implementação.

Apesar disso, o Modelo Entidade-Relacionamento (MER) e o projeto lógico não se afastaram drasticamente dos conceitos iniciais. Isso nos destacou a importância de um planejamento sólido e de um entendimento inicial claro do escopo do sistema.

Um dos principais desafios enfrentados pelo grupo foi o projeto lógico, que depende de um bom MER. Nas primeiras entregas, tivemos dificuldades para elaborá-lo, e somente no meio da implementação da última entrega consideramos o Projeto Lógico como concluído.

Outro obstáculo que enfrentamos foi a perda de membros do grupo, passando de 4 para 2 integrantes. Fomos obrigados a nos adaptar a essa redução súbita de membros e à mudança de perspectiva em relação ao tempo necessário para implementar o sistema e o protótipo.

Apesar das dificuldades, todos concordamos que o projeto foi uma excelente oportunidade para aplicar os conhecimentos de Banco de Dados aprendidos em sala de aula. Além disso, os feedbacks recebidos ao longo das entregas nos ajudaram a aprimorar nossos estudos e repensar nosso projeto, especialmente na parte de Mapeamento Lógico.

O sistema final mostrou-se simples, mas é uma aplicação com potencial para novos recursos e regras de negócio. Portanto, um dos desafios enfrentados pelo grupo foi manter o projeto dentro do escopo inicial, equilibrando o esforço dedicado à implementação, planejamento e documentação.