**Relatório BAN2**

**Função para cadastro de um livro (atualizada):**

select \* from livros

select \* from exemplares

insert into exemplares values (11,1,1,'A-101');

select count(cod\_b) from bibliotecario

create or replace function CadastraLivro(F\_isbn varchar, F\_titulo varchar ,F\_autor varchar , F\_editora varchar ,F\_colecao varchar )

    returns void as

$$

declare

    Next\_Id\_Livro int default 0;

    contagem int default 0;

    vLinhas int default 0;

begin

    select count(id) from livros into contagem;

    if(contagem < 1) then

        raise notice 'Não há livros na tabela livros';

        Next\_Id\_Livro := 1;

        else

            select max(id) from livros into Next\_Id\_Livro;

            Next\_Id\_Livro := Next\_Id\_Livro + 1;

    end if;

    insert into livros values

        (Next\_Id\_Livro,F\_isbn,F\_titulo,F\_autor,F\_editora,F\_colecao);

    get diagnostics vLinhas = row\_count;

    if(vLinhas < 1) then

        raise exception 'Erro ao inserir livro na tabela livros';

        return;

        else

            raise notice 'SUCESSSO: Livro inserido com sucesso!';

    end if;

end;

$$

language plpgsql;

select CadastraLivro('252525', 'Jack Sparrow' , 'James Gunn' , 'Delarte rio' ,'Novos 52' )

**create or replace function Reserva\_Livro(id\_usuario int,id\_exemplar int)**

**returns int as**

**$$**

**declare**

**next\_id\_Reserva int default 0;**

**data\_Reserva date := CURRENT\_DATE;**

**Hora\_Reserva time := CURRENT\_TIME;**

**vLinhas int default 0;**

**begin**

**select max(id) from reservas into next\_id\_Reserva;**

**if(next\_id\_Reserva is null ) then**

**next\_id\_Reserva := 1;**

**else**

**next\_id\_Reserva := next\_id\_Reserva + 1;**

**end if;**

**insert into reservas values**

**(next\_id\_Reserva,id\_usuario,id\_exemplar,data\_Reserva,Hora\_Reserva);**

**get diagnostics vLinhas = row\_count;**

**if(vLinhas > 0) then**

**raise notice 'SUCESSO: Reserva feita!';**

**return 1;**

**else**

**raise notice 'ERRO: Reserva não autorizada';**

**return 0;**

**end if;**

**end;**

**$$**

**language plpgsql;**

**create or replace function verificaReserva() returns trigger as**

**$$**

**declare**

**vColecao varchar;**

**begin**

**if exists (**

**select \* from exemplares ex join livros l on ex.id\_livro = l.id**

**where ex.id = new.id\_exemplar and l.colecao = 'Reserva'**

**) then**

**raise exception 'livro da coleção reserva não pode ser emprestado';**

**else**

**return new;**

**end if;**

**end;**

**$$**

**language plpgsql;**

**create or replace function Empresta\_Livro(id\_usuario int,id\_exemplar int)**

**returns void as**

**$$**

**declare**

**next\_id\_emprestimo int default 0;**

**data\_emprestimo date := CURRENT\_DATE;**

**data\_devolucao date := date\_trunc('day', data\_emprestimo + interval**

**'15 day');**

**vLinhas int default 0;**

**categoria\_usuario text;**

**begin**

**select max(id) from emprestimo into next\_id\_emprestimo;**

**if(next\_id\_emprestimo < 1 ) then**

**raise exception 'não há emprestimos na tabela emprestimos';**

**return;**

**end if;**

**next\_id\_emprestimo := next\_id\_emprestimo + 1;**

**select categoria from usuarios where id = id\_usuario into categoria\_usuario;**

**if(categoria\_usuario = 'ALUNO\_GRADUACAO') then**

**data\_devolucao := date\_trunc('day', data\_emprestimo + interval '15 day');**

**elsif(categoria\_usuario = 'ALUNO\_POS\_GRADUACAO') then**

**data\_devolucao := date\_trunc('day', data\_emprestimo + interval '30 day');**

**elsif(categoria\_usuario = 'PROFESSOR') then**

**data\_devolucao := date\_trunc('day', data\_emprestimo + interval '30 day');**

**elsif(categoria\_usuario = 'PROFESSOR\_POS\_GRADUACAO') then**

**data\_devolucao := date\_trunc('day', data\_emprestimo + interval '90 day');**

**else**

**data\_devolucao := date\_trunc('day', data\_emprestimo + interval '15 day');**

**end if;**

**insert into emprestimo values**

**(next\_id\_emprestimo,id\_usuario,id\_exemplar,data\_emprestimo,null,data\_devolucao);**

**get diagnostics vLinhas = row\_count;**

**if(vLinhas > 0) then**

**raise notice 'SUCESSO: Emprestimo feito!';**

**return;**

**else**

**raise notice 'ERRO: Emprestimo não autorizado';**

**return;**

**end if;**

**end;**

**$$**

**language plpgsql;**

**Tarefas:**

~~Atualizar os gatilhos que foram feitos na parte 3 para que impeçam a inserção no banco de dados conforme a String passada com parâmetro.~~

~~A devolução é um delete na tabela de empréstimo. Assim, o exemplar ainda existe na sua tabela de dados, para realizar um empréstimo, basta passar o usuário e o exemplar, mas antes, é necessário verificar se o empréstimo não está reservado, ou se não está emprestado a um outro usuário.~~

~~If(NãoEstaReservado && NaoEstaEmprestado)~~

~~Reservar~~

~~Na devolução de um empréstimo:~~

~~Qual livro do usuário passado como parâmetro ele quer devolver? Listar todos os empréstimos~~

~~O usuário deve digitar o id do empréstimo.~~

~~Perguntar qual assistente está fazendo a operacao devolução do livro. Fazer um update na tabela de usuários~~

~~Verificar automaticamente a data de devolução que está presente na tabela de empréstimo, se a data no momento da devolução do livro for 1 dia a mais da data de devolução que está presente na tabela de empréstimo, cobrar multa do usuário.~~

~~Assim, atualizar o valor da multa no empréstimo, e só será apagada, após o usuário realizar o pagamento da multa (aí sim irá tirar esse dado da tabela de empréstimo).~~

~~Então, para realizar a reserva, será necessário verificar:~~

~~Não reservar se: o id do exemplar estiver presente na tabela de empréstimo e multa for igual a null -> assim, o livro ainda está em posse do usuário.~~

~~Reservar se: o id do exemplar não existir na tabela de empréstimo ou se existir, mas a multa for diferente de null (assim, o usuário devolveu o livro, e agora o livro está disponível, mas o usuário que fez esse empréstimo ainda possui multas em seu nome.)~~

~~If(id\_exemplar não existir na tabela de empréstimo || (id\_exemplar da tabela de reserva for igual a algum exemplar na tabela de empréstimo && multa != null) {~~

~~Reservar()  
}~~

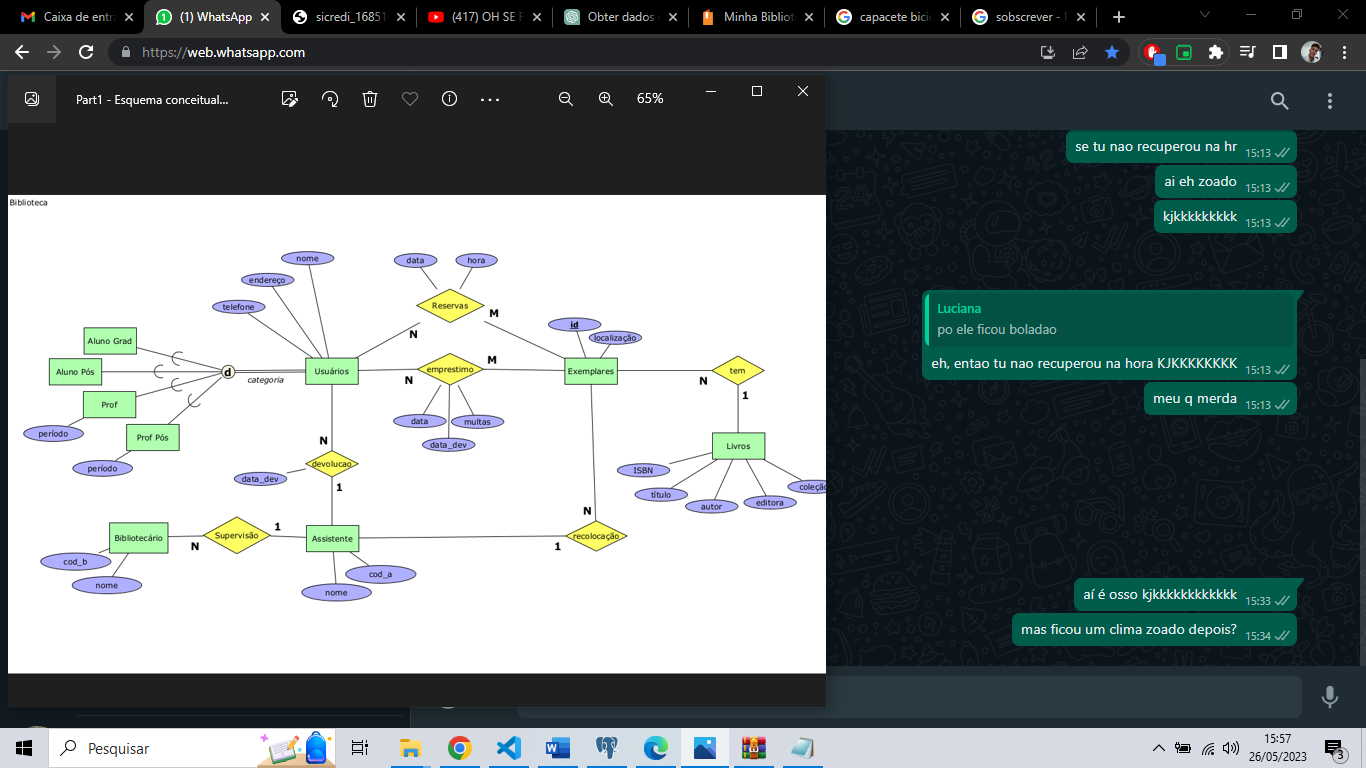
**Se reservar, essa reserva é deletada da tabela de reserva e o usuário que solicitou a reserva agora fica na tabela de empréstimo.**

*~~Após a devolução do livro, a tabela exemplares é atualizada com o id do assistente que recebeu o exemplar. O empréstimo de um livro só poderá ser realizado se o id\_assistente da tabela exemplar for diferente de null, pois assim, quer dizer que o assistente recolocou o livro de volta na estante. -> Não tenho certeza dessa informação e não sei como tratar o relacionamento “recolocação” entre assistente e Exemplar.~~*

*~~O atributo recolocação pode ser atualizado para “Cadastro”, em que é o assistente que faz o cadastro de um exemplar, visto que é ele quem determina a localização que o livro irá ficar.~~*

*~~Mas ainda estou em dúvida quanto a isso, é necessário verificar se o livro já está disponível para empréstimo.~~*

~~Acho que é bom retirar essa parte do banco de dados:~~



**supervisioná-lo logo após a inserção.**

**Periodo dos usuários:**

Aluno Grad -> Turno: Matutino/Vespertino

Aluno Pos-Grad -> Turno: Noturno

Professor -> Turno: Matutino/Vespertino, Noturno ou Integral (perguntar para o usuario)

Professor Pos Grad -> Turno: Matutino/Vespertino, Noturno ou Integral (perguntar para o usuario)

**Objetivo:** O aluno deve implementar um sistema, em linguagem de programação orientada a obejtos, que implemente algumas das funcionalidades definidas na primeira parte do trabalho final (especificação dos requisitos de dados do sistema). Este sistema deve possuir três camadas, onde a camada de apresentação deve permitir a inserção e consulta das informações contidas em algumas tabelas do esquema de dados. A implementação do sistema, incluindo o esquema de banco de dados, deve ser compactada e enviada pelo botão abaixo.

**Funcionalidade a serem implementadas pelo sistema:**

* Cadastrar Bibliotecário
* Cadastrar Assistentes
* Cadastrar Usuário
* Cadastrar Livro
* Cadastrar Exemplar de um Livro
* Realizar Empréstimo de um Exemplar
* Se estiver emprestado, perguntar se o usuário quer reservar. Se sim, Reservar.
* Localizar Usuário que está com um exemplar.

**Funções a serem criadas:**

* Função para Cadastro de um Bibliotecário no PostgreSQL
* Função para Cadastro de um Assistente no PostgreSQL
* Função para cadastro de um usuário no PostgreSQL
* Provavelmente será necessário mudar o cadastro de livros e exemplares, e dividir em duas funções. Uma para cadastro de livro e outra para cadastro de exemplar.
* Função para Reserva de Livros

**Documentação do código Duvidas:**

* Talvez seja necessário mudar o “Date” da classe Usuário para o date do java.util.