

1

# CURSO DE CIÊNCIA DA COMPUTAÇÃO

**Desenvolvimento de um sistema para um prontuário eletrônico.**

D7177A4 – Luan Daniel Nardini D6451D4 – Pedro Otávio Teixeira N313JE4 – Gabriel Andrade Cabral

T6791J6 – Arthur Antunes Da Rosa Augusto D660EA4 – Eduardo Monita Dias D787748 – Murilo Linares Mattioli

RIBEIRÃO PRETO – SP 2019

# ÍNDICE

[Objetivo e motivação 1](#_TOC_250006)

[Introdução 2](#_TOC_250005)

[Referencial teórico 3](#_TOC_250004)

[Plano de desenvolvimento da aplicação 4](#_TOC_250003)

Plano de desenvolvimento da aplicação 5

Plano de desenvolvimento da aplicação 6

[Projeto do programa 7](#_TOC_250002)

Projeto do programa 8

Relatório com as linhas de código 9

Relatório com as linhas de código 10

Relatório com as linhas de código 11

Relatório com as linhas de código 12

Relatório com as linhas de código 13

Relatório com as linhas de código 14

Relatório com as linhas de código 15

Relatório com as linhas de código 16

[Apresentação do programa em funcionamento 17](#_TOC_250001)

Apresentação do programa em funcionamento 18

Apresentação do programa em funcionamento 19

Bibliografia 20

Fichas de atividades práticas supervisionadas. 21

Fichas de atividades práticas supervisionadas. 22

Fichas de atividades práticas supervisionadas. 23

Fichas de atividades práticas supervisionadas. 24

Fichas de atividades práticas supervisionadas. 25

Fichas de atividades práticas supervisionadas. 26

[Modelo conceitual do banco de dados… 27](#_TOC_250000)

Diagrama de Classe 28

Diagrama de Classe 29

# OBJETIVO E MOTIVAÇÃO

O objetivo do trabalho baseia-se em desenvolver um programa na linguagem Java (WEB) que possa inserir, ler e manipular informações referentes a um prontuário eletrônico, que receberá dados, de pacientes e médicos, que serão fornecidos pelo usuário, assim como informações do próprio utilizador do sistema, com o objetivo de facilitar o acesso a informações e também armazenamento de dados.

O prontuário médico é utilizado para cadastrar e consultar informações do paciente, como exames realizados, histórico das doenças, alergias, histórico familiar, entre outras. Assim, com todas as informações reunidas em um mesmo local, o atendimento ao paciente é facilitado, garantindo maior otimização no atendimento do paciente, e aumentando a agilidade e produtividade do médico responsável e facilidade aos atendentes que precisarão acessá-lo.

No desenvolvimento da aplicação, serão criadas as janelas de cadastro, layout do prontuário, transição entre telas, aplicações de consulta, entre outros. Enquanto o banco de dados ficará encarregado de possuir a estrutura de tabelas e objetos utilizados no funcionamento do sistema. Neste ficarão salvos os devidos cadastros de médicos, usuários e pacientes que estarão envolvidos no prontuário, e as demais informações utilizadas para o funcionamento dele.

No cadastro de pessoa, as informações serão divididas em três tipos: médico, usuário e paciente. No cadastro abstrato de pessoa, serão utilizados dados como nome, endereço, cidade, RG e CPF, telefone para contato, naturalidade e sexo. No caso do tipo ‘Médico’, será solicitado ainda o CRM dele, assim como para o tipo ’Usuário’, serão criados dados de login e senha para utilização. No tipo ‘Paciente’, ficarão disponíveis para cadastro as suas possíveis alergias e contatos de emergência.

A motivação do trabalho apresentado foi a do mesmo ser exigência para a avaliação do terceiro bimestre do 3° e 4° semestre, do curso de Ciência da Computação da Universidade Paulista, sob a orientação do professor Fabrício Gustavo Henrique.

# INTRODUÇÃO

Este estudo integra as Atividades Práticas Supervisionadas (APS), que se configura como um trabalho semestral, interdisciplinar e em equipe, do curso de Ciência da Computação da Universidade Paulista – UNIP de Ribeirão Preto – São Paulo.

Este trabalho apresenta as atividades desenvolvidas ao decorrer do semestre, de acordo com os conhecimentos e informações lecionadas pelo docente responsável pela matéria e através de pesquisas extracurriculares. A junção das informações e do conhecimento adquirido pelos membros do grupo possibilitou a realização deste trabalho documentado, assim como o programa solicitado, além de todas as partes exigidas para que a avaliação dele seja realizada.

Tendo como objetivo principal a criação e desenvolvimento de um programa elaborado através das linguagens JAVA (WEB) e SQL. Sendo requerido ao programa as seguintes funções:

* Cadastro de informações de pessoa (médico, usuário e paciente).
* Vínculo dos dados de pessoa ao prontuário do paciente.
* Controle de status do prontuário (Ativo ou Inativo).
* Referência ao local onde o prontuário físico se encontra (localização predial, como: recepção, enfermagem, consultório, entre outros.).

O método de pesquisa utilizado contou com a ampla revisão dos conteúdos ministrados em aula, leitura e análise de livros sobre o tema e técnicas utilizadas, os quais contribuíram com dados importantes para a composição final do trabalho construído pelo grupo.

No conteúdo do trabalho, após apresentados os dados teóricos sobre, foram contemplados códigos utilizados no programa em si, para que não possua em sua construção códigos padrão da IDE utilizada (NetBeans).

Finalizando o documento, estarão disponíveis as fichas da Atividade Prática Supervisionada de cada um dos membros no grupo, com apontamento detalhado das horas utilizadas para o desenvolvimento de cada parte e etapa do projeto.

# REFERENCIAL TEÓRICO

¹ Friedrich Wilhelm Nietzsche – **Bancos de dados e JDBC:**

https://[www.caelum.com.br/apostila-java-web/bancos-de-dados-e-jdbc/](http://www.caelum.com.br/apostila-java-web/bancos-de-dados-e-jdbc/)

² Java Swing: **Conheça os componentes JTextField** https://[www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-](http://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-) jformattedtextfield/30981

³ JDBC – **Banco de dados**

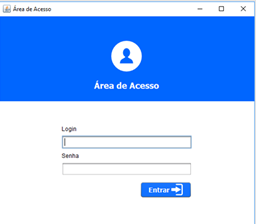
https://[www.devmedia.com.br/jdbc-tutorial/6638](http://www.devmedia.com.br/jdbc-tutorial/6638)

4 Instrução Inherits – **Microsoft**

https://docs.microsoft.com/pt-br/dotnet/visual-basic/language- reference/statements/inherits-statement

# PLANO DE DESENVOLVIMENTO DA APLICAÇÃO

O projeto tem como estruturação inicial, tela inicial para que o funcionário e cliente possa logar no programa com seu usuário e senha, parecido com o esquema abaixo (imagem pega da internet).



Assim que logar, utilizando o usuário e senha previamente cadastrada anteriormente, será carregado à tela referente ao setor que aquele usuário se encontra o método “login ()” é o responsável por validar o usuário e chegar a pagina referente a ele. Existe no total atualmente cinco telas de redirecionamento, onde são, “Enfermagem”, “Medico”, “RH”, “Recepção” e por fim o “Paciente”.

A página “Enfermagem” é construída com os seguintes conceitos, existe a opção referente aos dados de Pessoa, Paciente, Alergia e Alergia referente ao paciente. Todos os fatores que o usuário do setor enfermagem pode fazer são constituídos através de links, permitindo tanto listar ou cadastra os mesmos.

A página “Medico” é construída com os seguintes conceitos, existe a opção referente aos dados de Pessoa, Médico e Prontuário. Todos os fatores que o usuário do setor medico pode fazer é constituído através de links, permitindo tanto listar ou cadastra os mesmos.

A página “Paciente” é construída com os seguintes conceitos, existe a opção referente aos dados de Prontuário. O fator que o usuário do setor pode fazer é constituído através de links, permitindo listar o seu prontuário, percebesse que é

permitido somente à parte de listar para o paciente, pois ele não tem permissão para fazer nenhuma alteração no sistema apenas visualizar a sua ficha.

A página “Recepção” é construída com os seguintes conceitos, existe a opção referente aos dados de Pessoa e Usuário. Todos os fatores que o usuário do setor recepção pode fazer são constituídos através de links, permitindo tanto listar ou cadastra os mesmos. Esse setor é responsável por gerir a parte de pessoas e também de usuários.

A página “RH” é construída com os seguintes conceitos, existe a opção referente aos dados de Pessoa, Medico Usuário, Setor e Local. Todos os fatores que o usuário do setor Rh pode fazer são constituídos através de links, permitindo tanto listar ou cadastra os mesmos. Esse setor ficou responsável pelo os cadastros de parâmetros da empresa juntamente com os dados das pessoas livres para manipulação.

Agora falaremos um pouco mais sobre a parte do código em si e em como foi construído o projeto, primeiramente falaremos sobre o banco de dados, ele possui as seguintes tabelas em sua estrutura, tb\_alergia, tb\_paciente, tb\_alergia\_paciente, tb\_pessoa, tb\_medico, tb\_setores, tb\_local\_fisico, tb\_usuario e pôr fim a tb\_prontuário. A sua estrutura foi feita para suportar toda a nossa lógica de negócio de como queremos armazenar os nossos dados, a linguagem responsável por auxiliar essa questão foi a “plpgsql” no Postgres.

O projeto foi feito inteiramente em Java e xhtml, para a conexão com o banco de dados foi utilizado à unidade de persistência JPA, assim sendo possível gerar as classes entidades a partir desta conexão. Também foi seguido o todo o gerenciamento do projeto utilizado JSF, onde para cada tabela existe um ManagerBean podendo gerenciar os mesmos.

O projeto tem como objetivo para armazenar os dados dos usuários cadastrados foi feita uma conexão com o banco de dados via JDBC, e criadas as entidades necessárias para suprir a aplicação.

“ ³ JDBC é semelhante ao ODBC, e no princípio usava justamente ODBC para conectar-se com o banco de dados. A partir de um código nativo as aplicações Java podiam utilizar qualquer banco de dados que tivesse um driver ODBC disponível. Isso contribuiu bastante para a popularização do JDBC uma vez que existe um driver ODBC para praticamente qualquer banco de dados de mercado.

Assim como ODBC, JDBC também funciona através de drivers que são responsáveis pela conexão com o banco e execução das instruções SQL.”

Foi utilizado também o comando inheritz para que as próximas tabelas criadas recebam também as colunas existentes em uma tabela que serviria de modelo.

“4 Se usado, o Inherits instrução deve ser a primeira linha não está em branco, sem comentário em uma definição de classe ou interface. Você deve seguir imediatamente o Class ou Interfaceinstrução.

Você pode usar Inherits somente em uma classe ou interface. Isso significa que o contexto da declaração para uma herança não pode ser um arquivo de origem, namespace, estrutura, módulo, procedimento ou bloco.

Regras:

Herança de classe. Se uma classe usa o Inherits instrução, você pode especificar apenas uma classe base.

Uma classe não pode herdar de uma classe aninhada dentro dele.

Herança de interface. Se uma interface usa o Inherits instrução, você pode especificar uma ou mais interfaces base. Você pode herdar de duas interfaces, mesmo se cada um deles definir um membro com o mesmo nome. Se você fizer isso, o código de implementação deve usar a qualificação de nome para especificar qual membro que está implementando.

Uma interface não pode herdar de outra interface com um nível de acesso mais restritivo. Por exemplo, uma Public interface não pode herdar de um Friend interface. Uma interface não pode herdar de uma interface aninhada dentro dele.

Um exemplo de herança de classe no .NET Framework é o ArgumentException classe, que herda o SystemException classe. Isso fornece ao ArgumentException todas as propriedades pré-definidas e procedimentos exigidos por exceções de sistema, como o Message propriedade e o ToStringmétodo.”

# PROJETO DO PROGRAMA

A realização das interfaces gráficas foi utilizada JSF para a construção das interfaces, telas e janelas. Nos comandos utilizados dentro da aplicação, foi utilizada a criação de métodos e objetos conforme visto em sala de aula, com os devidos construtores da classe, herança entre as classes e os relacionamentos entre elas.

Para a conexão com o banco de dados, será utilizado à biblioteca JDBC (Java DataBase Connectivity).

¹ “A conexão a um banco de dados é feita de maneira elegante com Java. Para evitar que cada banco tenha a sua própria API e um conjunto de classes e métodos, temos um único conjunto de interfaces muito bem definidas que devem ser implementadas. Esse conjunto de interfaces fica dentro do pacote java.sql e nos referiremos a ele como JDBC. ”

No funcionamento da aplicação, cada tipo de usuário terá permissões diferentes para executar comandos na aplicação.

O usuário definido como atendente (ou recepcionista), por exemplo, possui permissão para cadastrar novos usuários da aplicação, cadastrar um novo médico ou também realizar um cadastro de um novo paciente. O usuário definido como médico, por sua vez, poderá cadastrar o paciente e obter acesso apenas à consulta do prontuário de um paciente, ou então alterar um prontuário já cadastrado de um paciente. O paciente terá uma tela para consulta de seus prontuários,

A tela inicial da aplicação é a de login, sendo necessário um login pré-criado que irá a partir daí realizar o cadastro dos novos colaboradores. Após o login será apresentado uma tela de menu com as ações disponíveis para o usuário, médico ou paciente conectado; como cadastrar novos usuários, médicos, pacientes, e inserir um novo prontuário ou alterá-lo, consultar prontuários já inseridos, entre outros.

No botão de consulta, será possível realizar a pesquisa por prontuários, usuários, pacientes e médicos, de acordo com o acesso de cada login cadastrado no sistema. Para realizar essa consulta, foi utilizado um Rádio Button para que o usuário

conectado possa definir se deseja realizar a consulta de um usuário do tipo médico, paciente ou atendente.

A arquitetura do banco de dados foi idealizada a partir dos seguintes pontos: para cadastro, foi criada a tabela ‘tb\_pessoa’, que armazenará os dados em comum de todos os usuários da aplicação. As tabelas ‘tb\_usuario’, ‘tb\_medico’ e ‘tb\_paciente’ herdam as informações da tabela ‘tb\_pessoa’ e possuem ainda os dados específicos de cada tabela. Além destas 4 tabelas, foi criada também a tabela ‘tb\_prontuario’ para que sejam armazenadas as informações sobre o prontuário de cada paciente. Após definida a arquitetura do banco de dados, a estrutura de cada tabela ficou definida, com o tipo da coluna que armazena cada dado.

Os botões de ‘Salvar’, ‘Alterar’, ‘Excluir’ presentes na aplicação, serão utilizados para passar como parâmetro o texto nos text fields, para assim realizarem os inserts nas devidas tabelas. Enquanto a aplicação de consulta de usuários e prontuários ficarão encarregadas de realizar os selects das informações desejadas em cada tabela.

# RELATÓRIO DE CÓDIGO

**BANCO:**

CREATE DATABASE db\_prontuario ENCODING = 'win1252'

Template template0 CONNECTION LIMIT = -1;

CREATE TABLE tb\_setores( id\_setor INTEGER,

nm\_setor VARCHAR(30) NOT NULL,

CONSTRAINT pk\_tb\_setores\_id\_setor PRIMARY KEY(id\_setor));

CREATE TABLE tb\_local\_fisico( id\_local\_fisico INTEGER,

nm\_local\_fisico VARCHAR(30) NOT NULL,

CONSTRAINT pk\_tb\_local\_fisico\_id\_local\_fisico PRIMARY KEY (id\_local\_fisico));

CREATE TABLE tb\_alergia( id\_alergia INTEGER, descr\_alergia VARCHAR(80),

CONSTRAINT pk\_tb\_alergia\_id\_alergia PRIMARY KEY (id\_alergia));

CREATE TABLE tb\_pessoa( id\_pessoa INTEGER,

nome VARCHAR(80) NOT NULL,

cpf VARCHAR(14) NOT NULL, rg VARCHAR(13) NOT NULL, nome\_mae VARCHAR(80) NOT NULL, endereco VARCHAR(60) NOT NULL,

numero INTEGER,

cep VARCHAR(11),

complemento VARCHAR(100),

cidade VARCHAR(20) NOT NULL,

bairro VARCHAR(20) , dt\_nascimento DATE NOT NULL,

sexo VARCHAR(10) NOT NULL,

naturalidade VARCHAR(20) NOT NULL, telefone VARCHAR(10),

CONSTRAINT pk\_tb\_pessoa\_id\_pessoa PRIMARY KEY(id\_pessoa));

CREATE TABLE tb\_paciente( id\_paciente INTEGER, tel\_emergencia VARCHAR(18),

CONSTRAINT pk\_tb\_paciente\_id\_paciente PRIMARY KEY (id\_paciente));

CREATE TABLE tb\_medico( id\_medico INTEGER,

id\_pessoa INTEGER,

crm VARCHAR(10) NOT NULL,

CONSTRAINT pk\_tb\_medico\_id\_medico PRIMARY KEY (id\_medico), CONSTRAINT fk\_tb\_pessoa\_id\_pessoa FOREIGN KEY (id\_pessoa)

REFERENCES tb\_pessoa(id\_pessoa));

CREATE TABLE tb\_paciente\_alergia( id\_paciente INTEGER,

id\_alergia INTEGER, nivel\_alergia INTEGER,

CONSTRAINT pk\_tb\_paciente\_alergia\_id\_paciente PRIMARY KEY (id\_paciente, id\_alergia),

CONSTRAINT fk\_tb\_paciente\_id\_paciente FOREIGN KEY (id\_paciente) REFERENCES tb\_paciente(id\_paciente),

CONSTRAINT fk\_tb\_alergia\_id\_alergia FOREIGN KEY (id\_alergia) REFERENCES tb\_alergia(id\_alergia));

CREATE TABLE tb\_prontuario( id\_prontuario INTEGER,

id\_medico INTEGER NOT NULL, id\_paciente INTEGER NOT NULL, observacao VARCHAR(250), situacao BOOLEAN NOT NULL, dt\_consulta DATE NOT NULL, id\_local\_fisico INTEGER NOT NULL, receita VARCHAR(200),

CONSTRAINT pk\_tb\_prontuario\_id\_prontuario PRIMARY KEY (id\_prontuario), CONSTRAINT fk\_tb\_medico\_id\_medico FOREIGN KEY (id\_medico)

REFERENCES tb\_medico(id\_medico),

CONSTRAINT fk\_tb\_paciente\_id\_paciente FOREIGN KEY (id\_paciente) REFERENCES tb\_paciente(id\_paciente),

CONSTRAINT fk\_tb\_local\_fisico\_id\_local\_fisico FOREIGN KEY (id\_local\_fisico) REFERENCES tb\_local\_fisico(id\_local\_fisico));

CREATE TABLE tb\_usuario( id\_usuario INTEGER, id\_setor INTEGER NOT NULL,

id\_pessoa INTEGER NOT NULL,

login VARCHAR(10) NOT NULL UNIQUE,

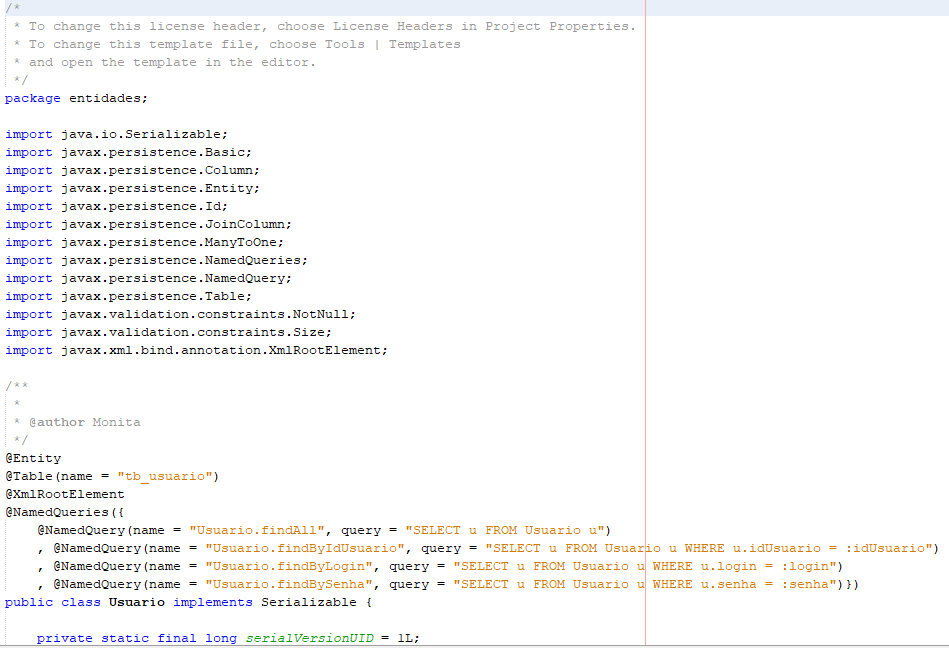
senha VARCHAR(15) NOT NULL,

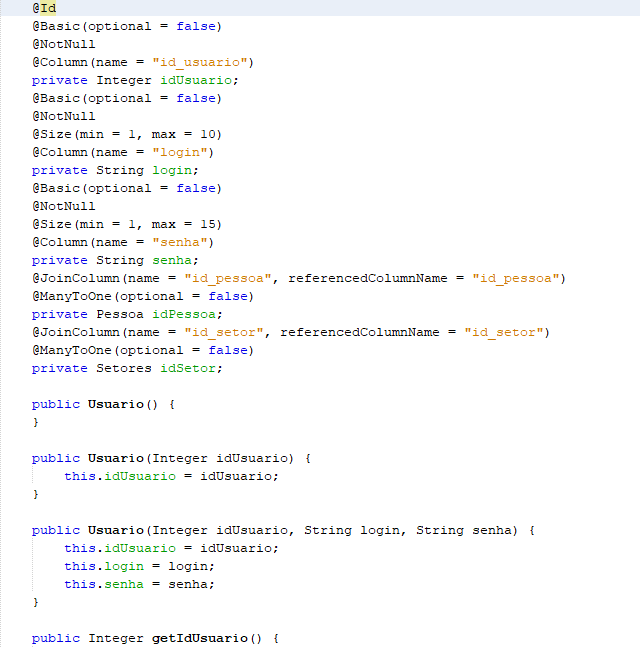
CONSTRAINT pk\_tb\_usuario\_id\_usuario PRIMARY KEY (id\_usuario), CONSTRAINT fk\_tb\_setor\_setor FOREIGN KEY (id\_setor)

REFERENCES tb\_setores(id\_setor),

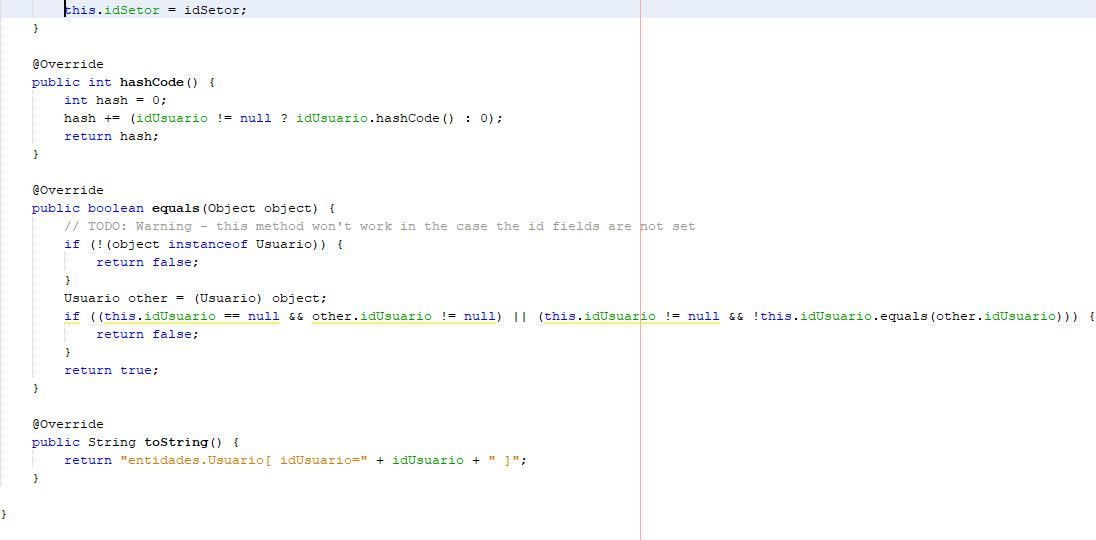
CONSTRAINT fk\_tb\_pessoa\_id\_pessoa FOREIGN KEY (id\_pessoa) REFERENCES tb\_pessoa(id\_pessoa));

# JAVA: Códigos referentes à entidade usuário





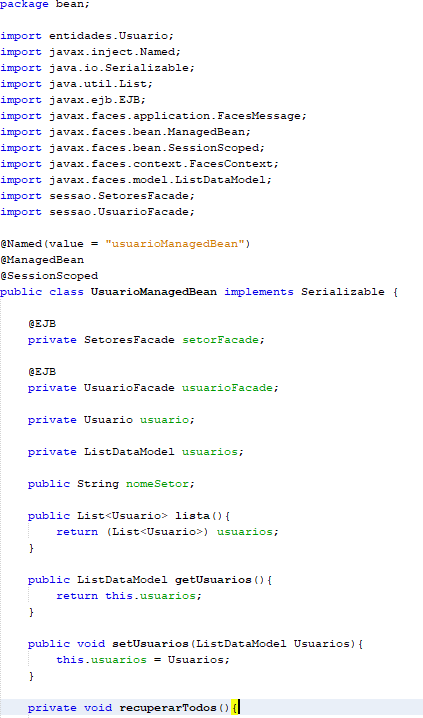


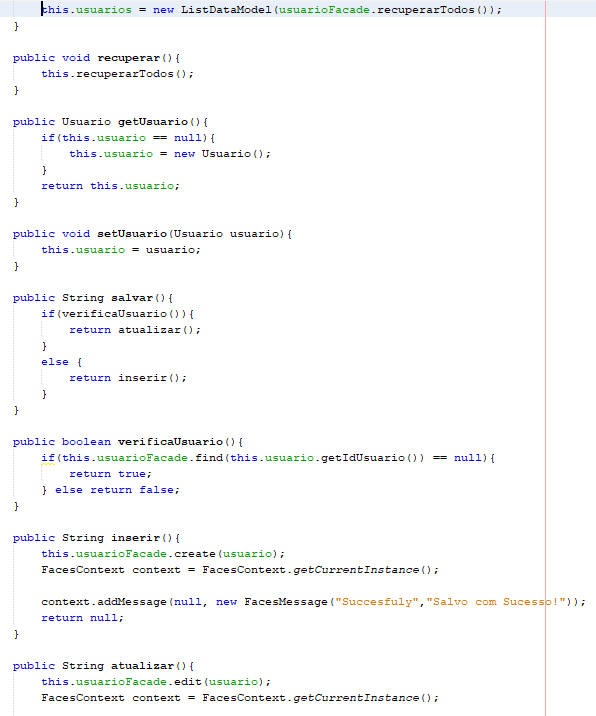


**JSF : LISTAR USUÁRIO**

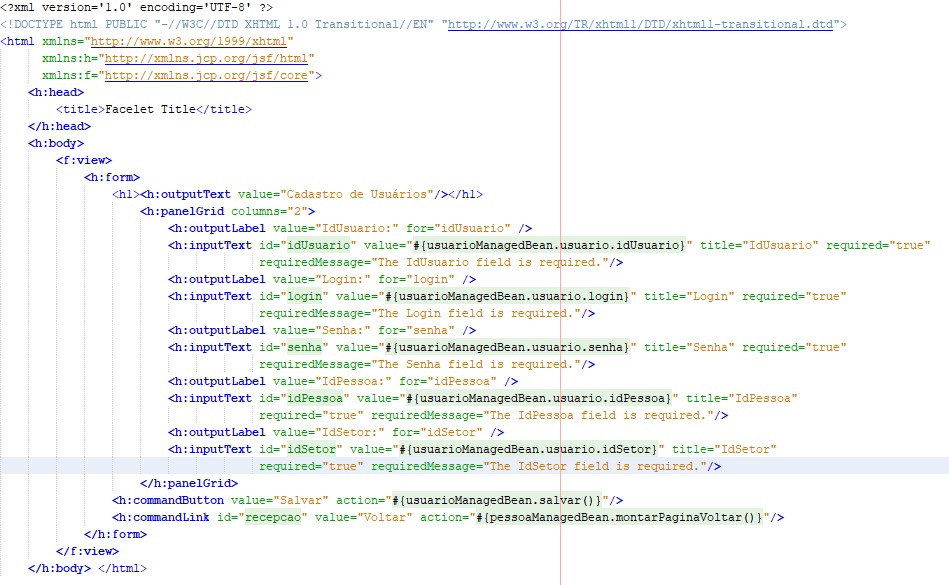


**MANAGED BEAN USUÁRIO**

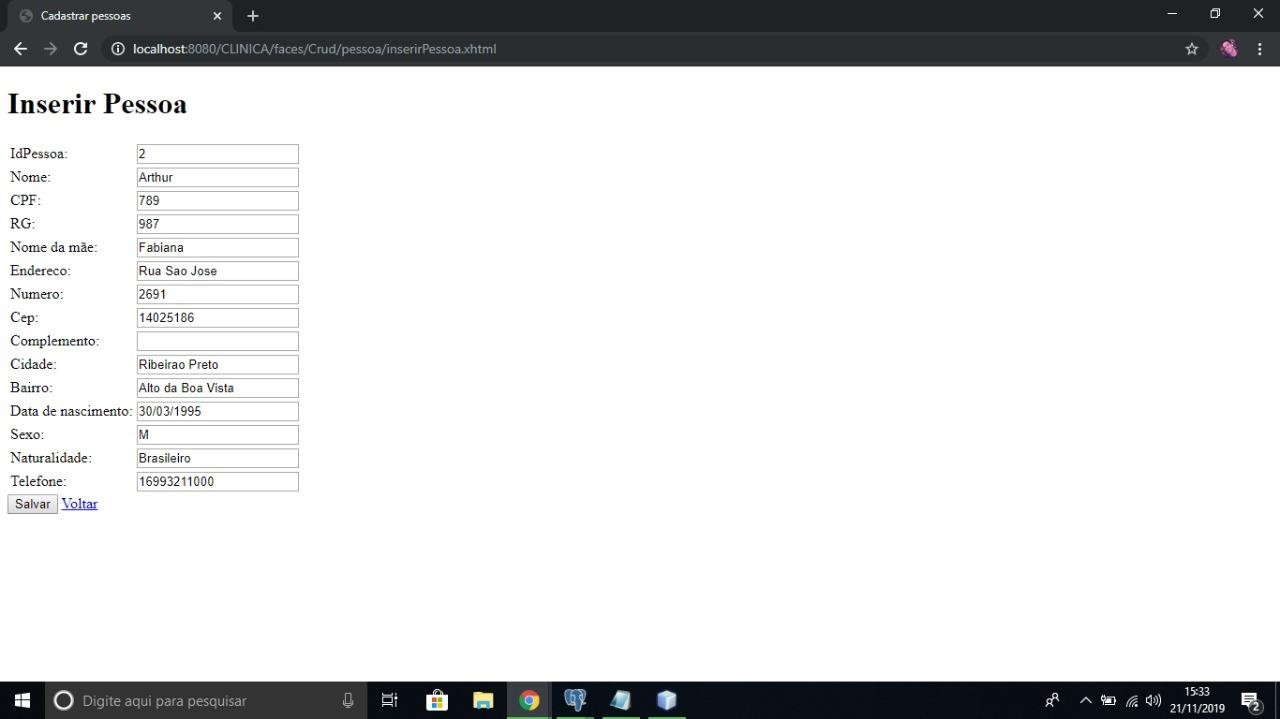


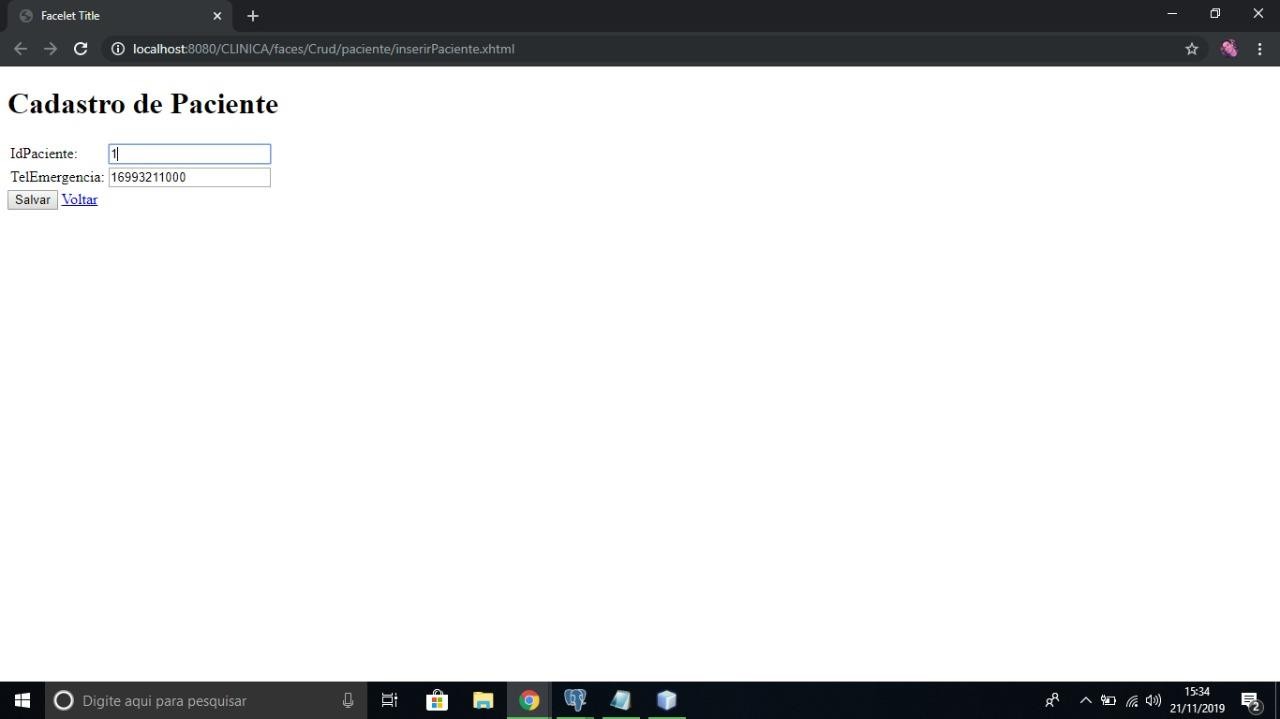


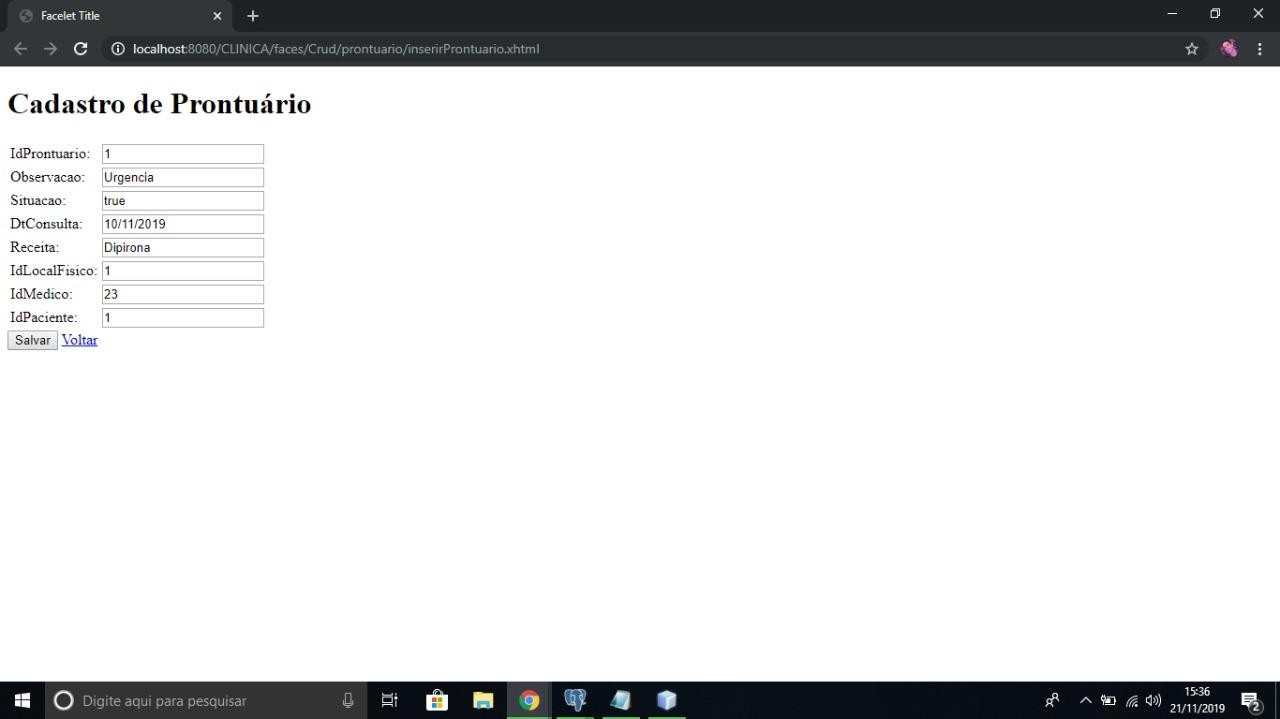


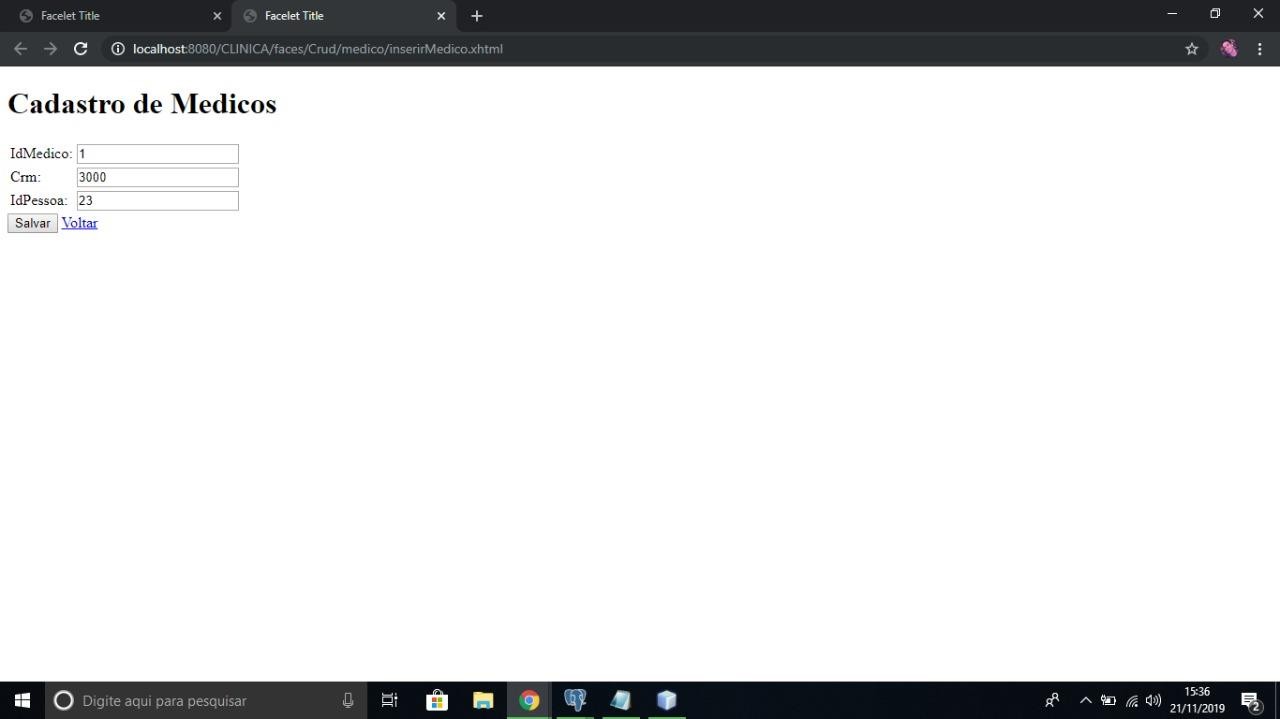
**SALVAR USUÁRIO**

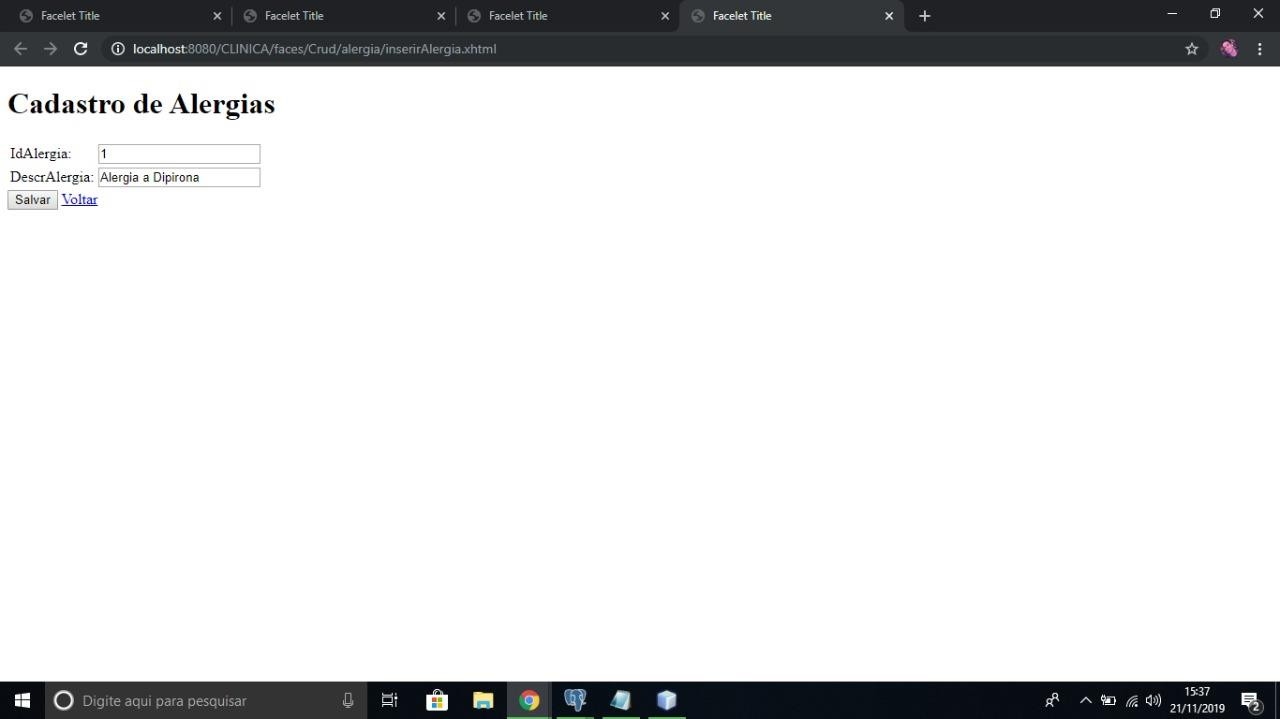
# Apresentação do programa em funcionamento

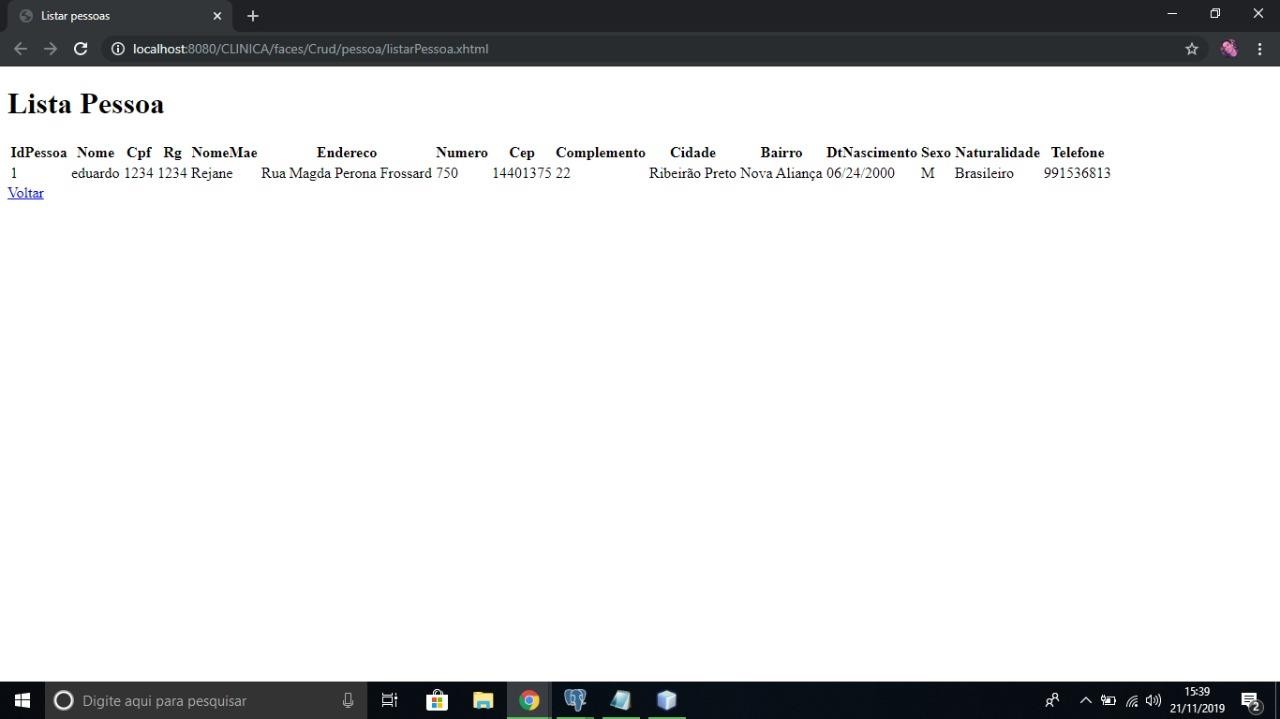












**BLIBLIOGRAFIA**

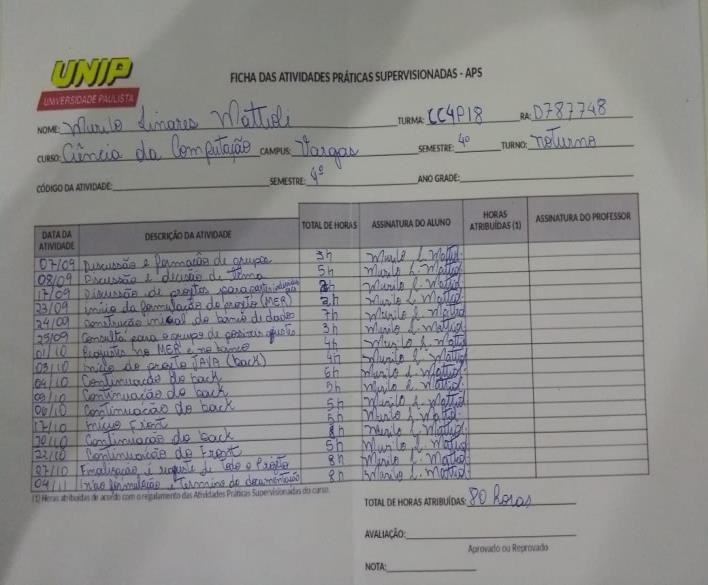
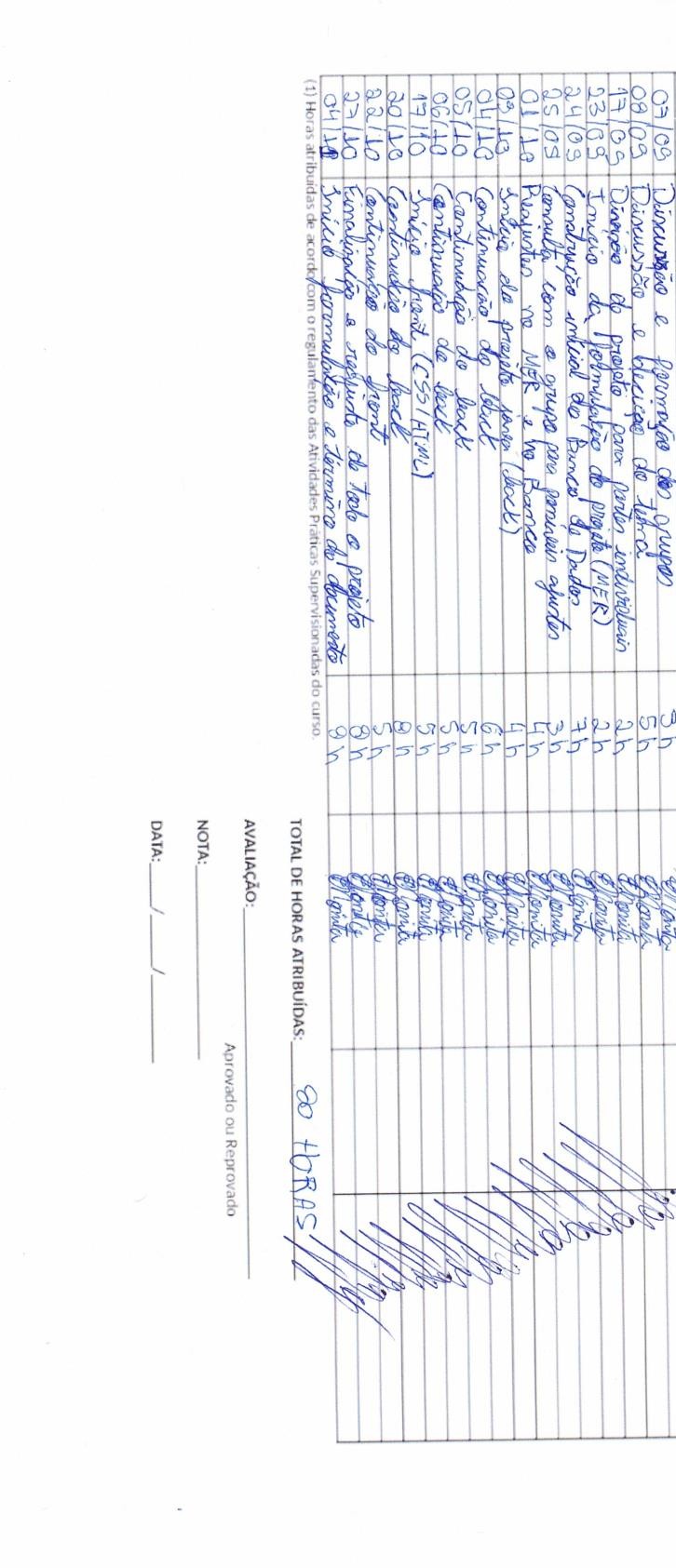
**Interfaces gráficas com jsf :**

[https://www.devmedia.com.br/uma-aplicacao-java-ee-completa-implementando-a-interface-web-](https://www.devmedia.com.br/uma-aplicacao-java-ee-completa-implementando-a-interface-web-com-jsf/8571) [com-jsf/8571](https://www.devmedia.com.br/uma-aplicacao-java-ee-completa-implementando-a-interface-web-com-jsf/8571)

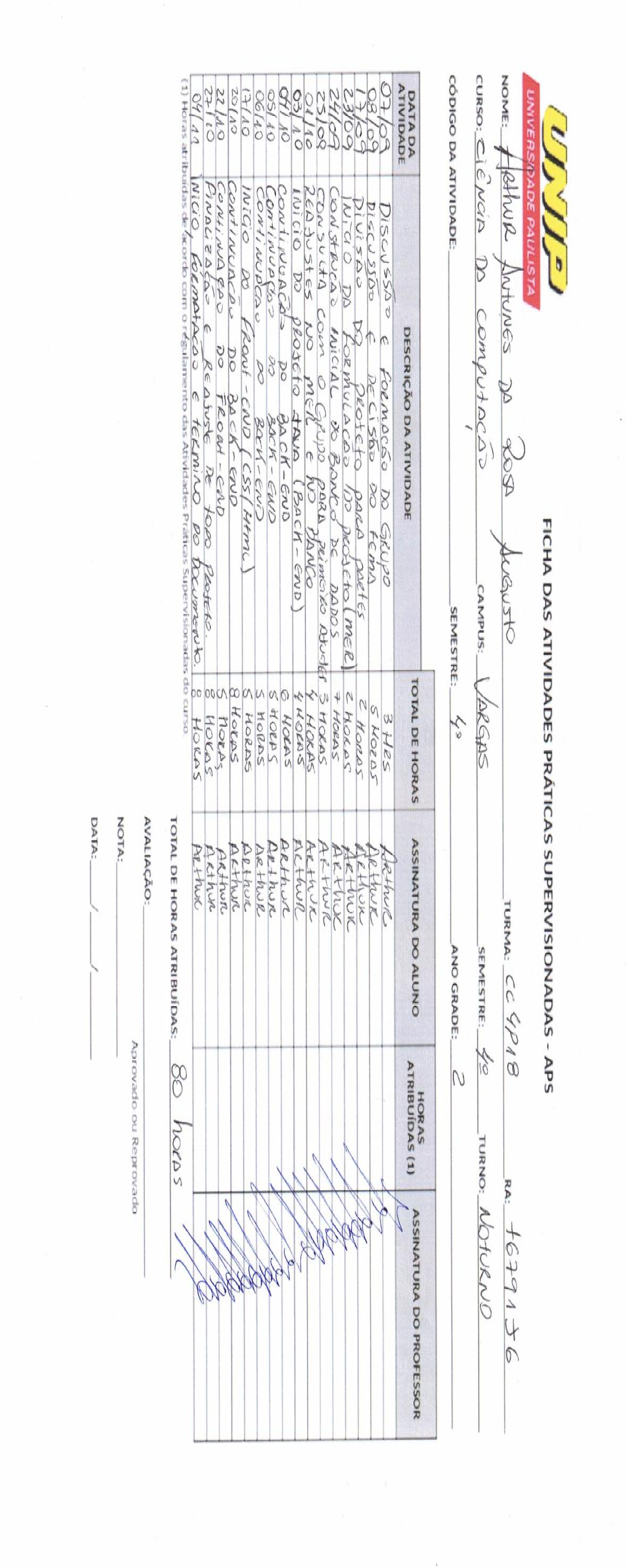
Tércio Costa, **Controlando privilégios herdados com o INHERIT PRIVILEGES:** [https://oraclepress.wordpress.com/2018/04/16/controlando-privilegios-herdados-](https://oraclepress.wordpress.com/2018/04/16/controlando-privilegios-herdados-com-o-inherit-privileges/) [com-o-inherit-privileges/](https://oraclepress.wordpress.com/2018/04/16/controlando-privilegios-herdados-com-o-inherit-privileges/)

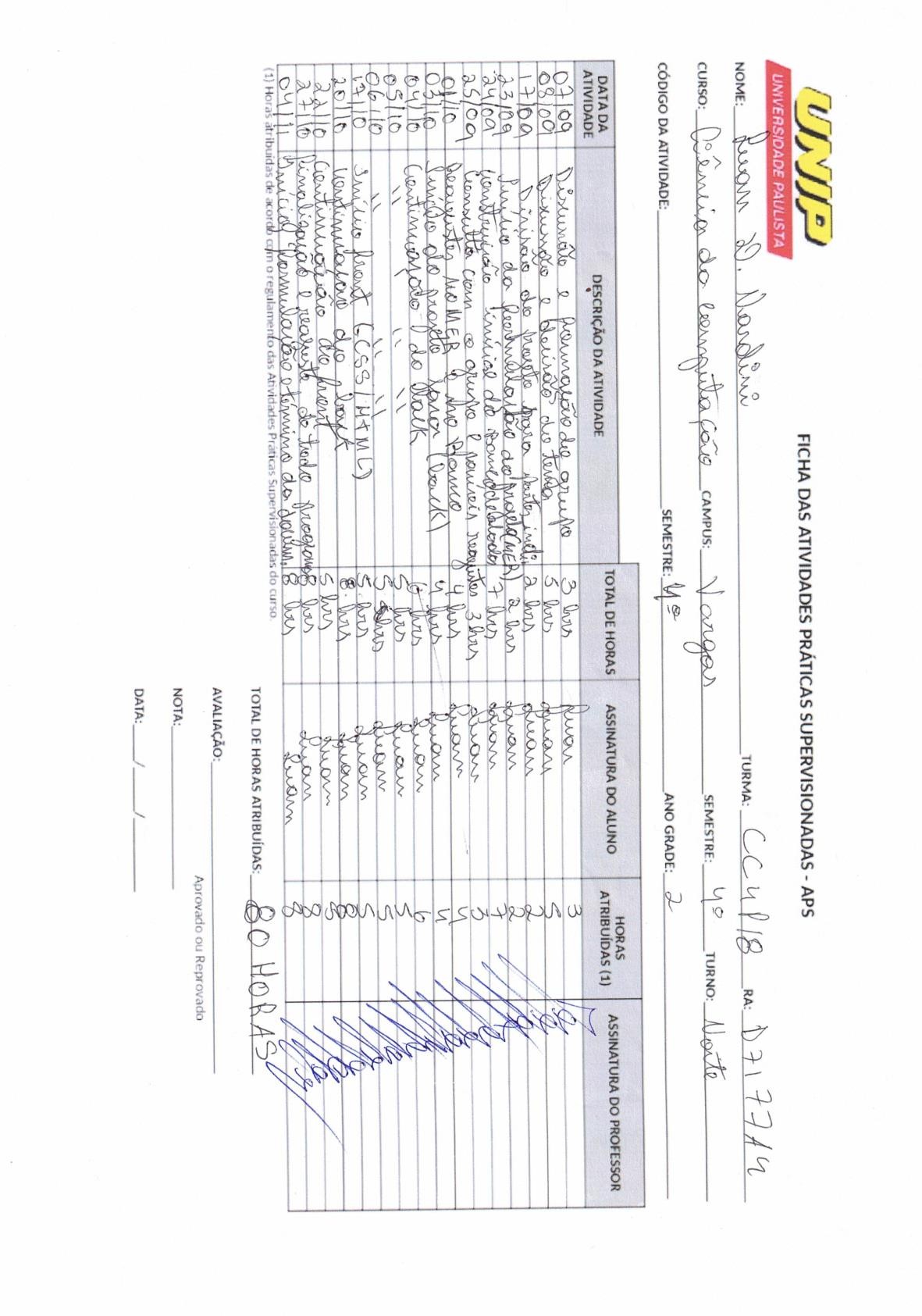
Fórum, **conexão java com postgres:**

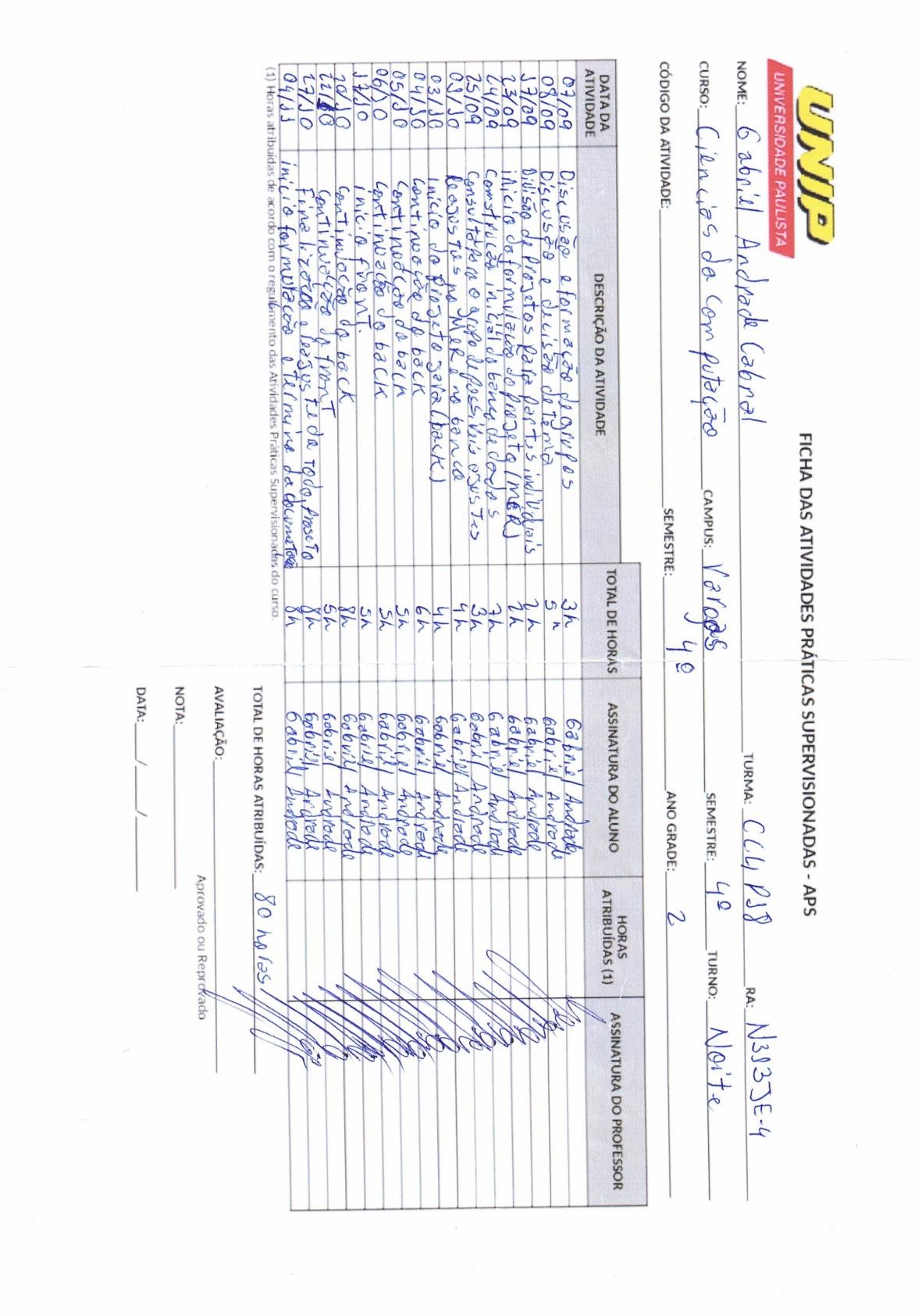
<https://www.guj.com.br/t/conexao-java-com-postgres/42713>

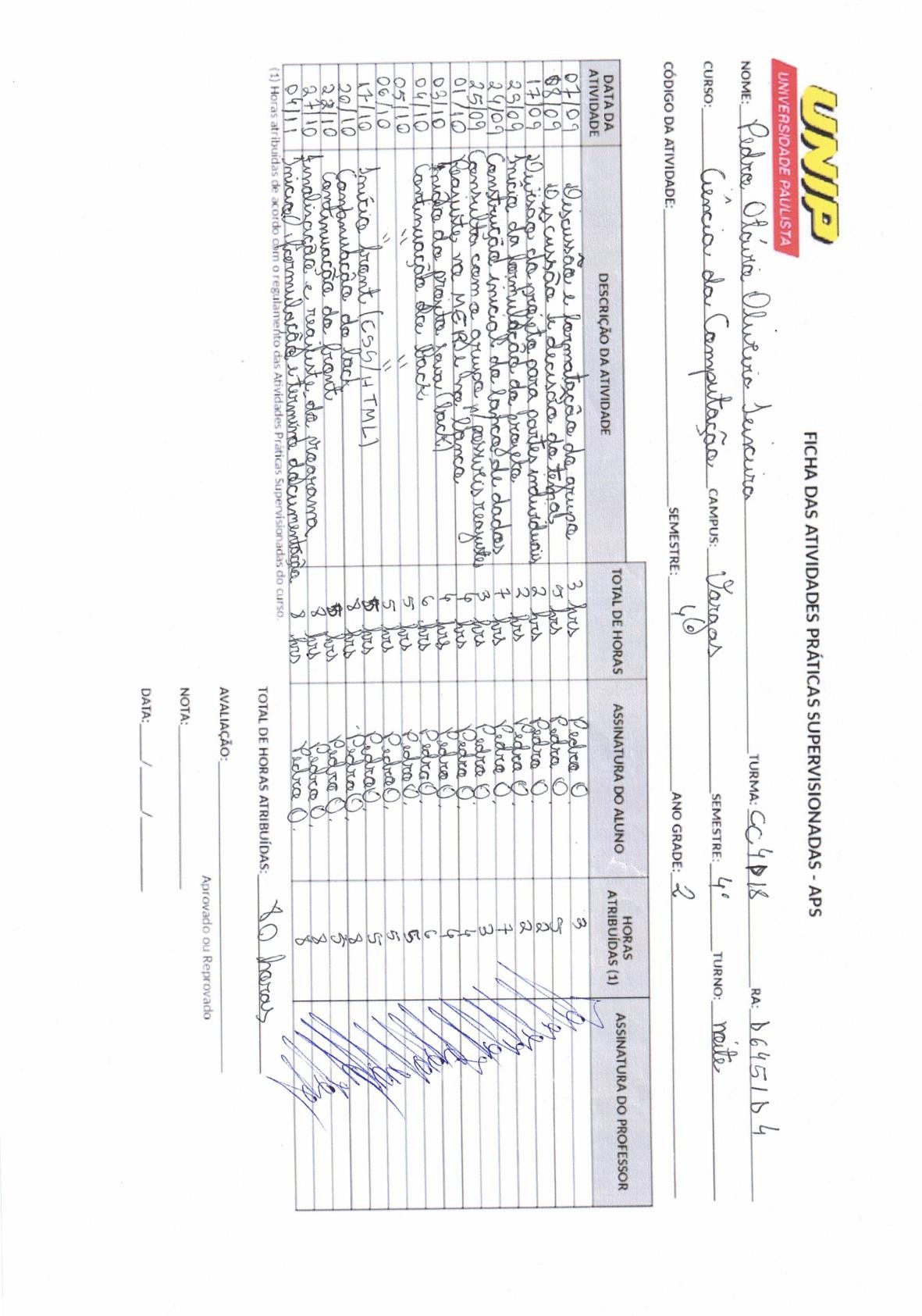


# FICHAS DE ATIVIDADES SUPERVISIONADAS

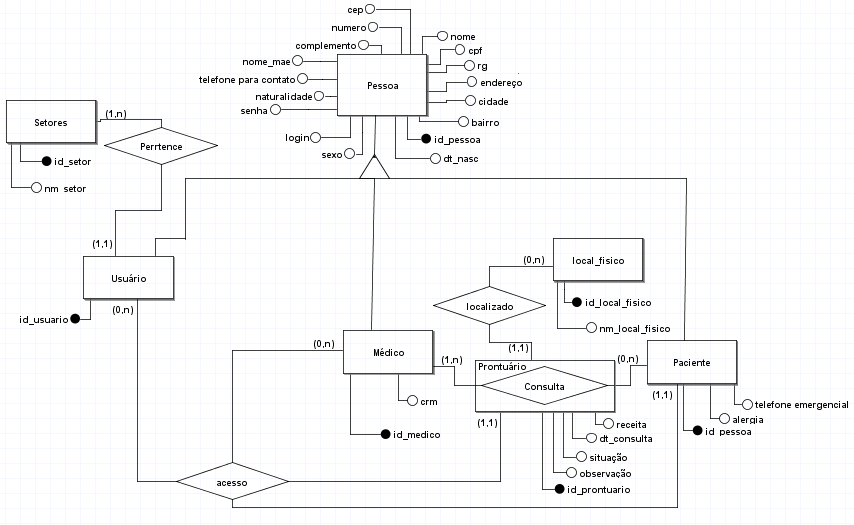


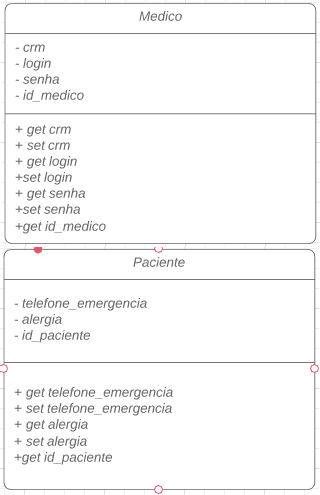






# MODELO CONCEITUAL DO BANCO DE DADOS





**DIAGRAMA DE CLASSES**

