Workshop JAVA 24/08/19

JAVA é:

* Aplicações
* Bibliotecas
* JDK
* Linguagem e JVM

Possui dois tipos de classes:

Negocio – descreve as características(atributos – variável)

Class Pessoa {

String Nome;

String Endereço;

Void fazercompra(double valor) {

Fazer compra (valor);

}

}

Que executa os métodos – main

**VARIÁVEL** – guardar os dados na memoria do computador; precisa ser definida que tipo de variável é (caractere, inteiro, etc)

**CLASSE**

* São agrupadores
* Tudo aquilo que tem características(**atributos**) e comportamento em comum pertencem a mesma casse
* Composta por: Atriibutos; métodos

Tudo executa dentro de um bloco

**MÉTODO** – diz o que vai acontecer dentro do bloco; para executar um método, precisa estar dentro de uma classe; descreve as ações que serão executadas

Class {

Método ()

{

Comando

}

}

**ATRIBUTO** – classes possuem atributos (características), as agrupam.

**OBJETO** – Produto da classe, o objeto é a representação real daquilo que a classe descreve; aplica todas as características da classe.

**CONSTRUTOR** – método responsável pela alocação de recursos necessários ao funcionamento do objeto. Pode ou não receber argumentos;

ENCAPSULAMENTO – sem o encapsulamento, qualquer pessoa pode entrar no código e modificar os atributos, pois está exposto. O encapsulamento esconde partes do código, usando os modificadores de acesso e métodos que fazem a leitura e escrita dessas informações.

Class Pessoa {

**Private** String Nome; //modificadores de acessos ‘private’

**Private** String Endereço;

Void fazercompra(double valor) {

Fazer compra (valor);

}

Public void setname (stringName){

This.name=nome;

}

Public String getNome (){

Return nome;

}

<http://dontpad.com/workshop-devsjavagirl>

**CRIANDO UM NOVO PROJETO NO ECLIPSE**

New > Java Project

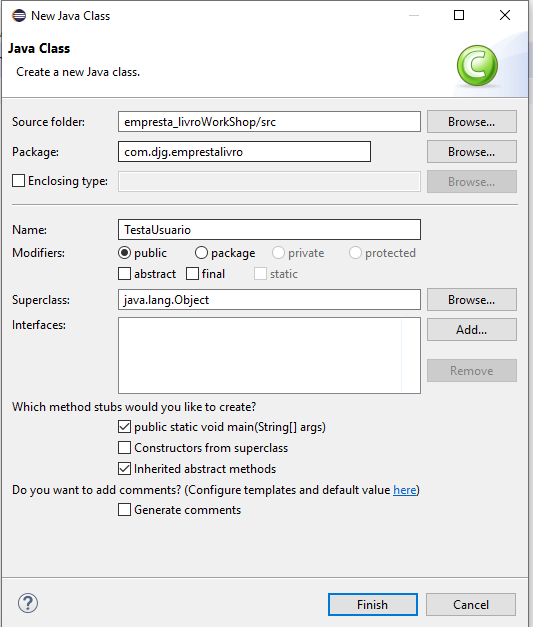
Criar um pacote(package) dentro de source (src) – organização das classes

A convenção indica que tem que criar os pacotes com o domínio da empresa com.br.inmetrics

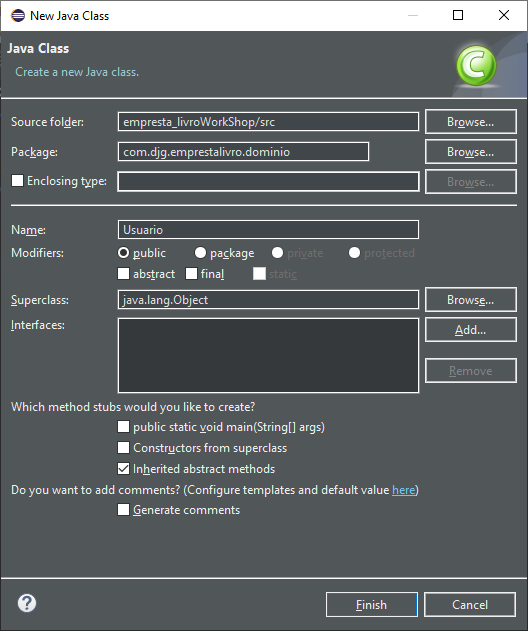
Classe tem que iniciar com letra minúscula – convenção

Variável – nomeVariavel - convenção

Para criar o método principal > new>class>frag public static void



Criar o pacote domínio > pacote de regras de negócio; característica do objeto

Criar classe de usuário dentro do domínio. Não marca o main

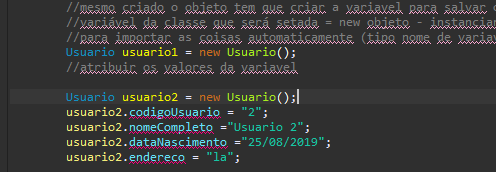
Colocar características para criar o usuário

**COMENTÁRIOS NO CÓDIGO**

**W!ReC@RD{2018}**

**ENCAPSULAMENTO**

**ANTES DO ENCAPSULAMENTO – VARIAVEIS TRIBUIDAS**

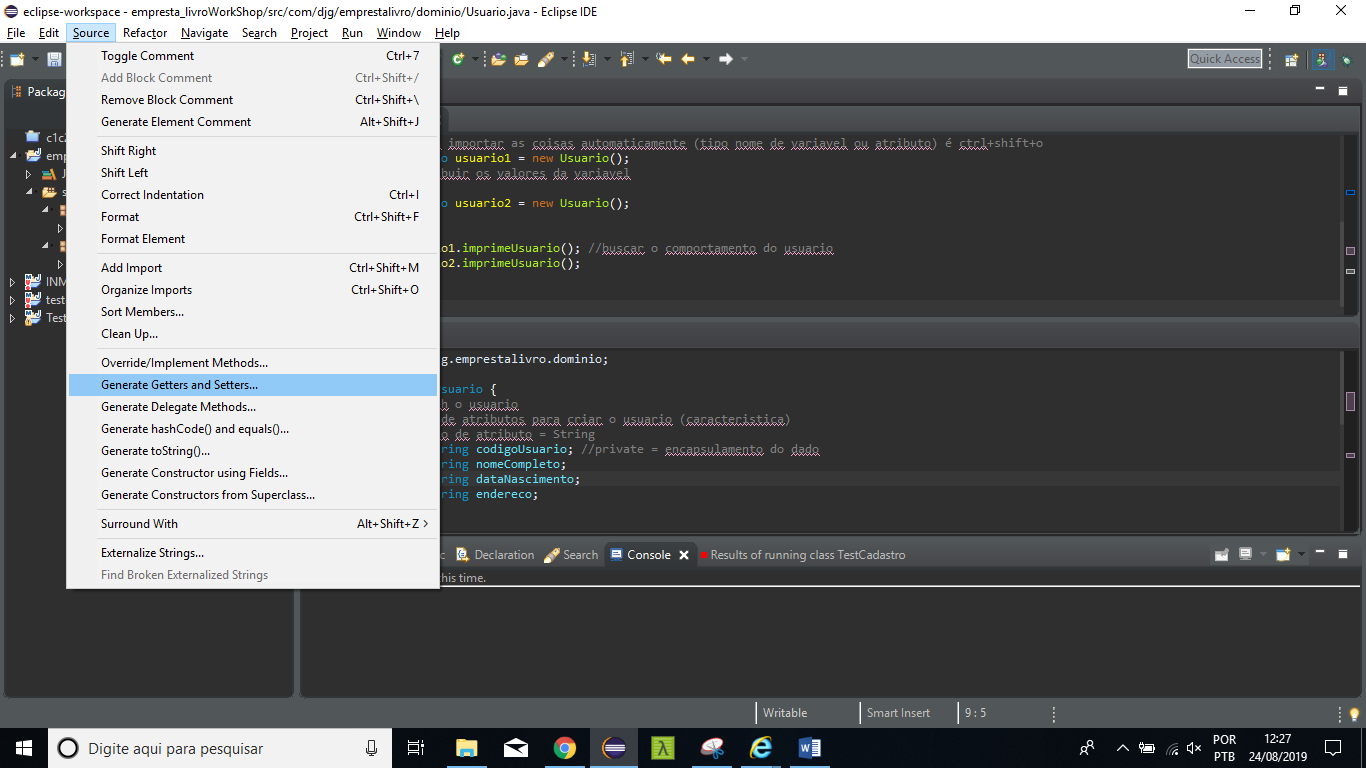


Colocar private no atributo no lugar do public

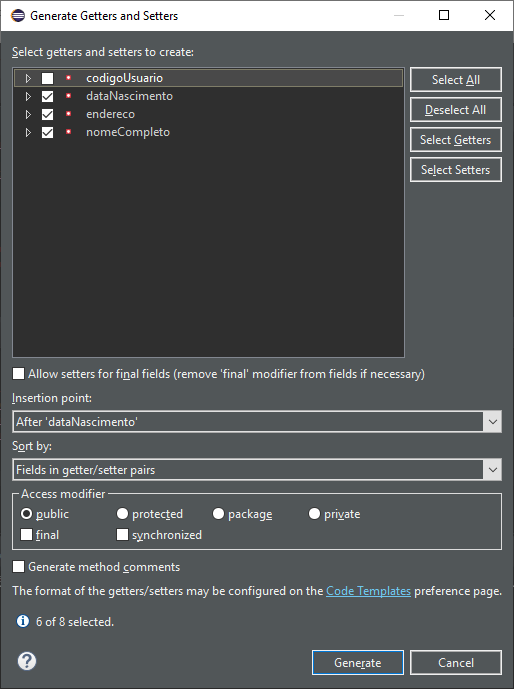
Depois pode apagar os dados da variável

Para criar os limites de visualização tem que criar um Getter

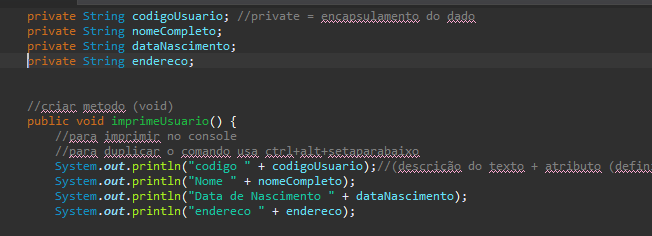
Source> generate getters and setters



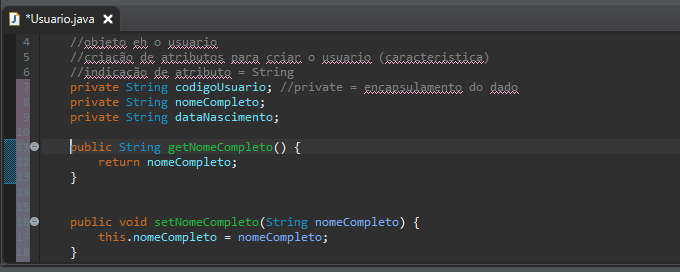
Selecionar o que eu quero que não seja visível



ANTES DO COMANDO

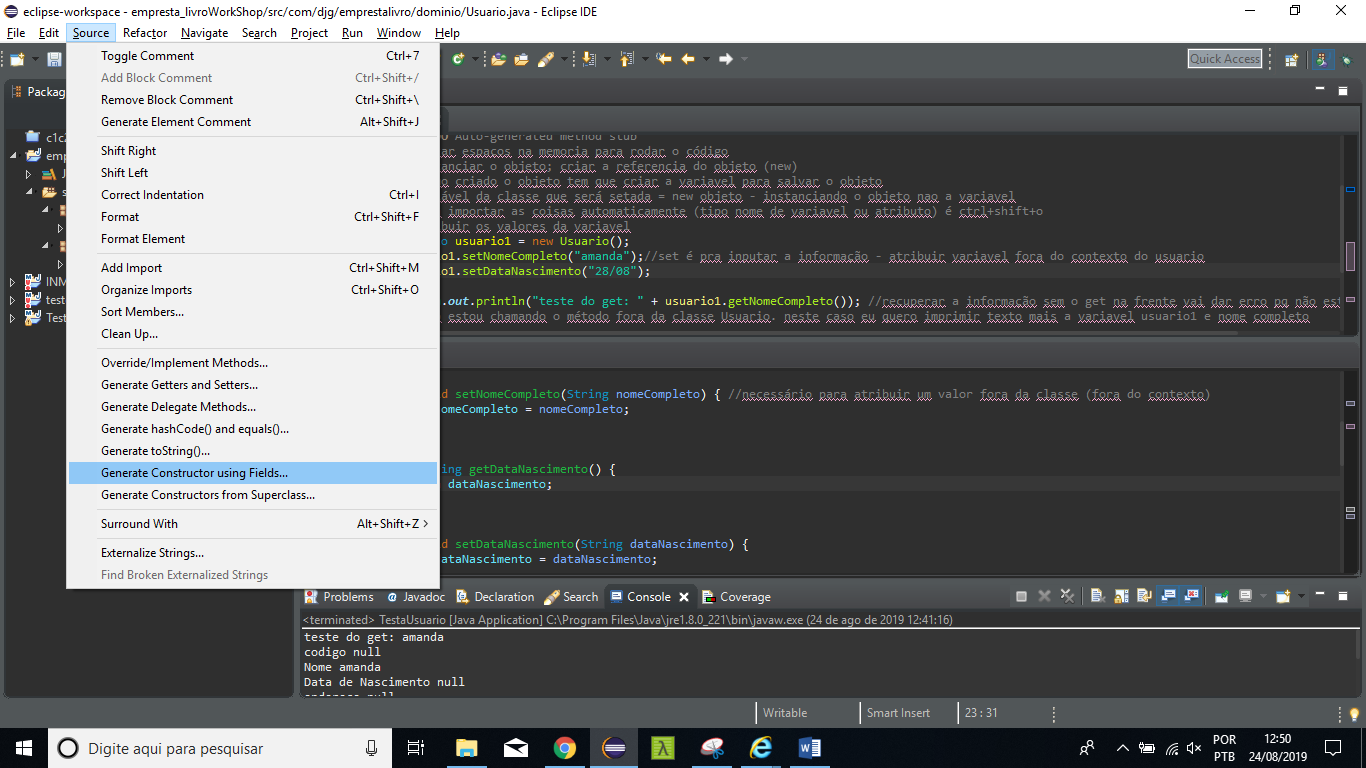


DEPOIS

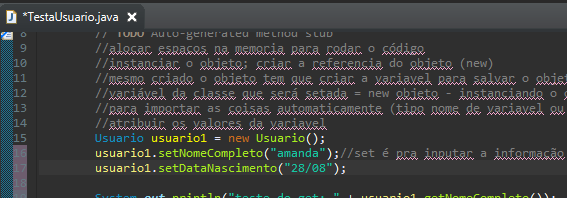


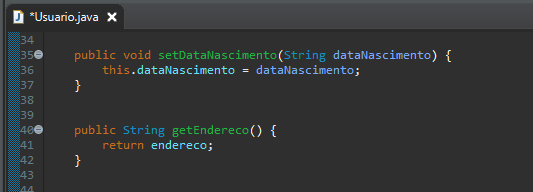
Para usar um construtor – para não escrever todas as linhas de set para todos os atributos

Source>Generate Constructor using Fields...

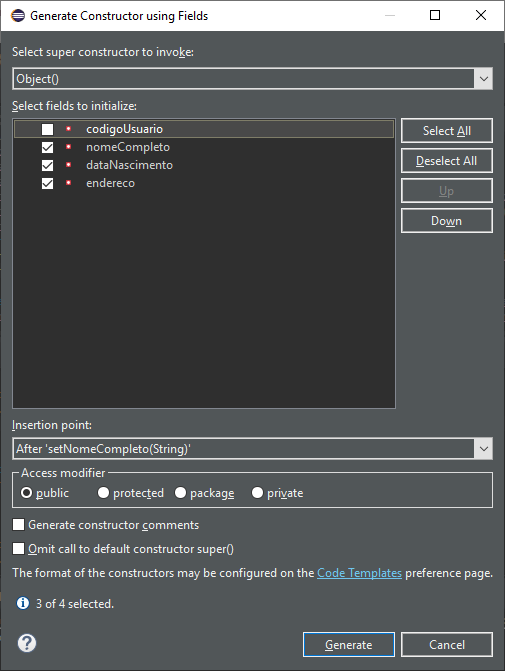


ANTES DO CONSTRUTOR

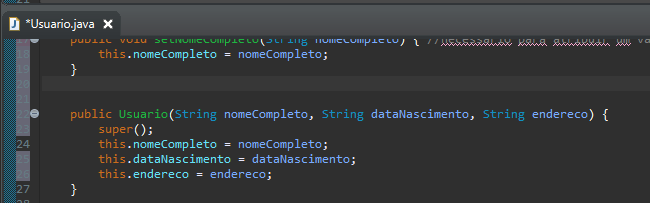




Seleciona o que quero aplicar



DEPOIS DE CRIADO



O construtor sempre terá o nome da classe

PASSO A PASSO CRIAÇÃO DE BIBLIOTECA

1. Criar classe – dentro do pacote domínio
2. Cria os atributos
3. Cria construtor
4. Testa a classe usando

25/08/2019

API REST

Ferramentas: GitHub; Postman; SprinRestaLisy; Maven

Postman – faz requisições http

Githup – versionador de código, compartilhamento de código

FRONTEND – site propriamente dito

BACKEND – cuida da logica da programação e cuida de toda a parte que o frontend irá exibir

REST – forma simples de integrar sistema; comunicação entre sistemas distintos – os dados são feitas de forma que qlqr aplicação consegue ler/consumir.

API – application, Programming, Interface – como é apresentado para o cliente

HTTP – expor os dados – protocolo de comunicação entre sistemas que permite a transferência de dados via web; tudo que transfere entre um servidor e a aplicação é feito por uma requisição(request) de texto, o servidor retorna um response em html

HyperText, Transfer, Protocol

REST - REpresentational State Transfer – utilize métodos padrão – regras

Comandos REST:

Get – listar ordens

Post – inserir

Put – editar

Delete – excluir

Para expor na internet para o frontend exibir tem que usar o Jason.

Linguagem Jason

“Pesso” {

“id”: 01,

“nome”: “amanda”,

“endereço”: “rua1”

}

BANCO DE DADOS

H2 console – banco de dados relacional

PARA INSERIR DADOS NA TABELA:

1. Selecionar a tabela
2. Insert into nomedatabela values (1, ‘Dan’, ‘o conto’) – infos que vao entrar na tabela

FAZER REQUISIÇÃO NO POSTMAN

Serve para fazer requisições e receber as requests

Bloco jason

{

“nome”: “nome do livro”

“autor”: “autor do livro”

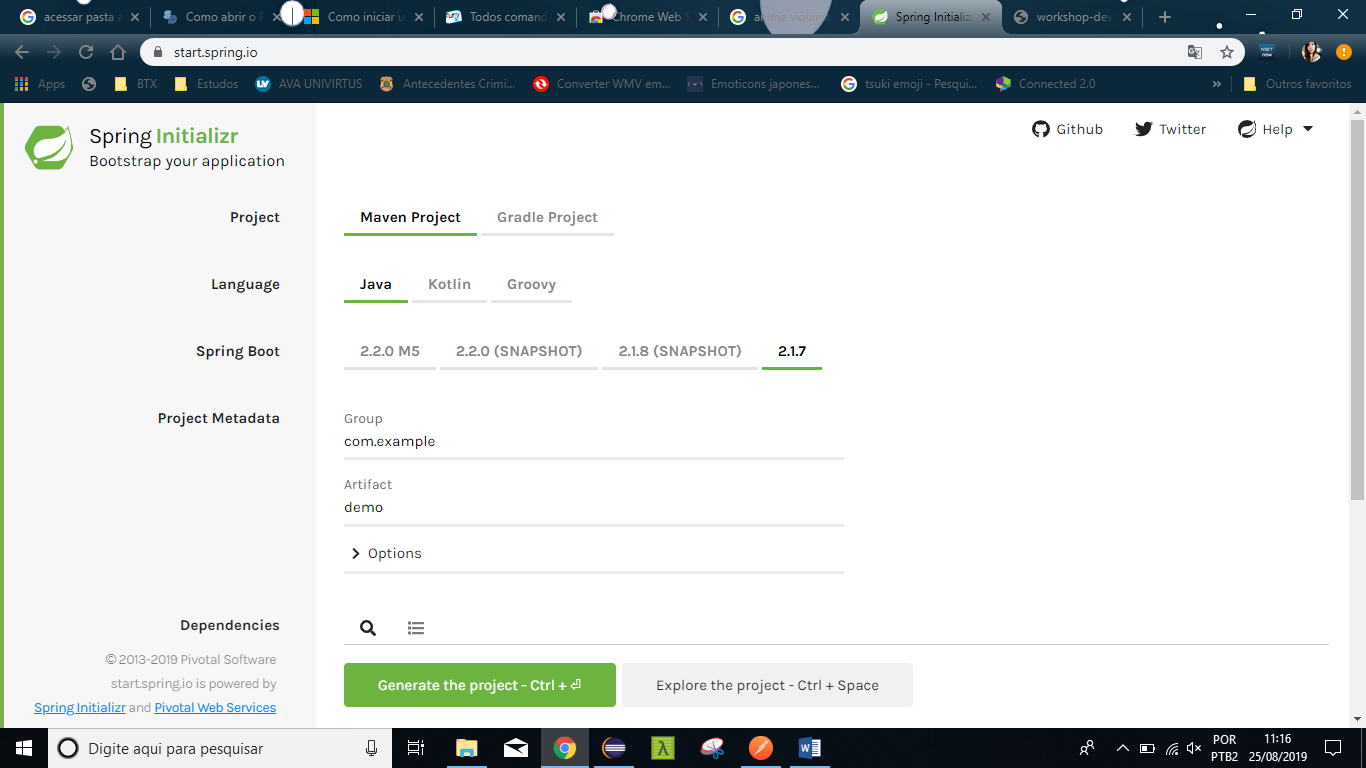
}

CRIAR UM post para enviar os dados para o endpoit (url)

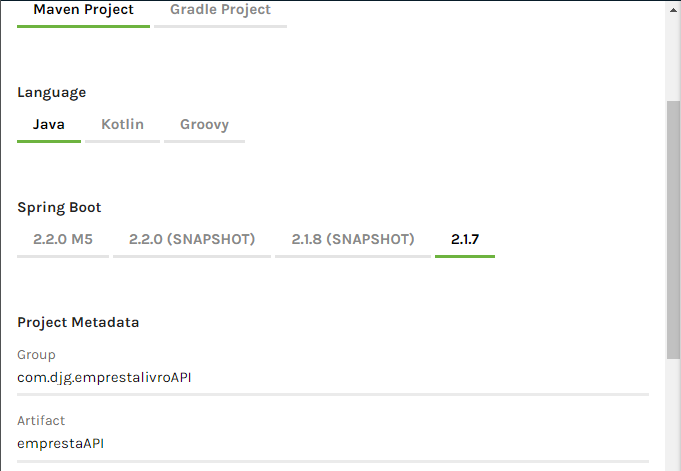
PROGRAMAR O POSTMAN

<https://start.spring.io/>

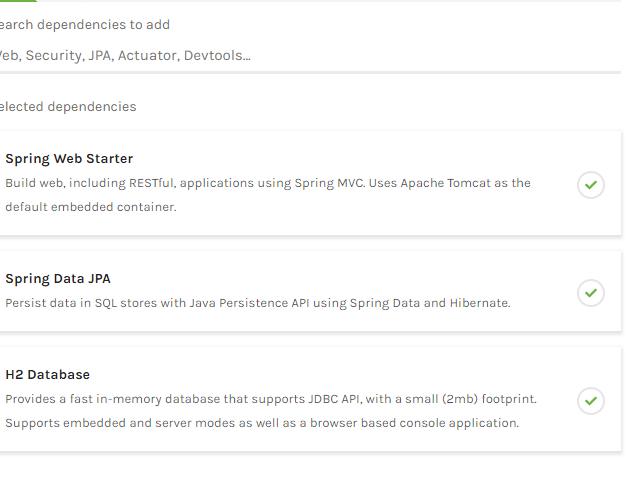
baixar



Colocar maven Project>java>2,17>colocar nomes



Colocar as dependências: especificação do java - JPA faz os inserts(conexão) no banco de dados, build web e o tipo de banco de dados



Generate the Project para fazer download > extrair para o workspace

Importar maven Project no eclipse > a pasta que foi feito o download no spring

1 – criar classes domino dentro da pasta de emprestaAPI

2 – criar get e setters

3 – colocar @Entity antes de public classe

Toda tabela de banco de dados precisa de um identificador iniciado com @Id em cima do atributo que eu quero que transforme em ID

*@Entity* //cria uma tabela chamada Usuario - indica que essa classe será uma tabela

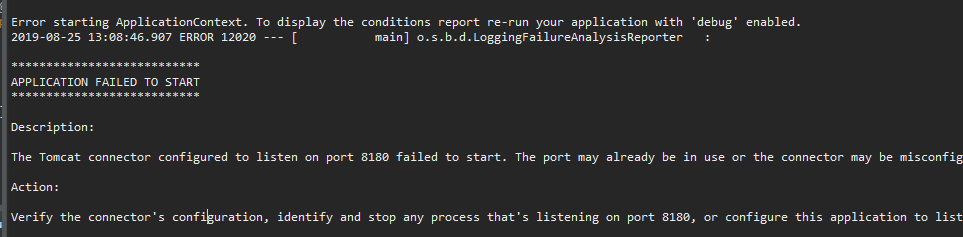
public class UsuarioAPI {

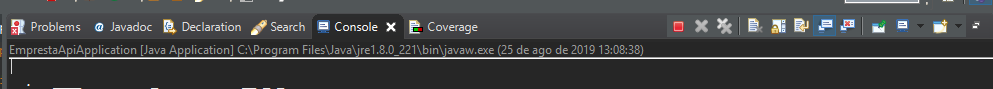
*@Id* //esse codigo usuario será um ID - está dando um Id para a tabela - sempre por em cima do atributo que quero transformar em ID

*@GeneratedValue* //e esse ID será gerado automaticamente

private Long codigoUsuario;//variavel tipo numero

No console, o botão quadrado vermelho serve para baixar a aplicação, o botão Xx serve para limpar. Quando da o erro abaixo, tem que apertar o xx e o quadrado vermelho





@many to one – mtos livros para vários emrpestimos

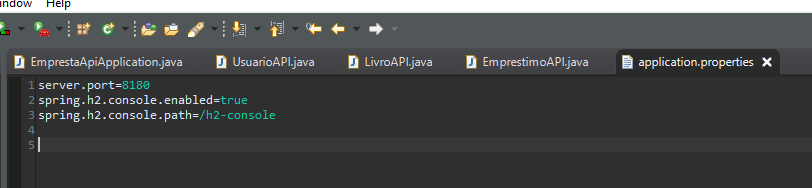
@one to many – 1 empretimo para muitos livros

Falar para o spring que queremos ver o banco de dados no application.properties

server.port=8180

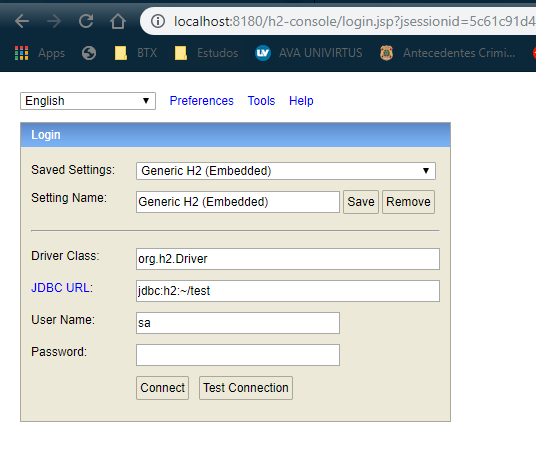
spring.h2.console.enabled=true

spring.h2.console.path=/h2-console



Roda novamente o console

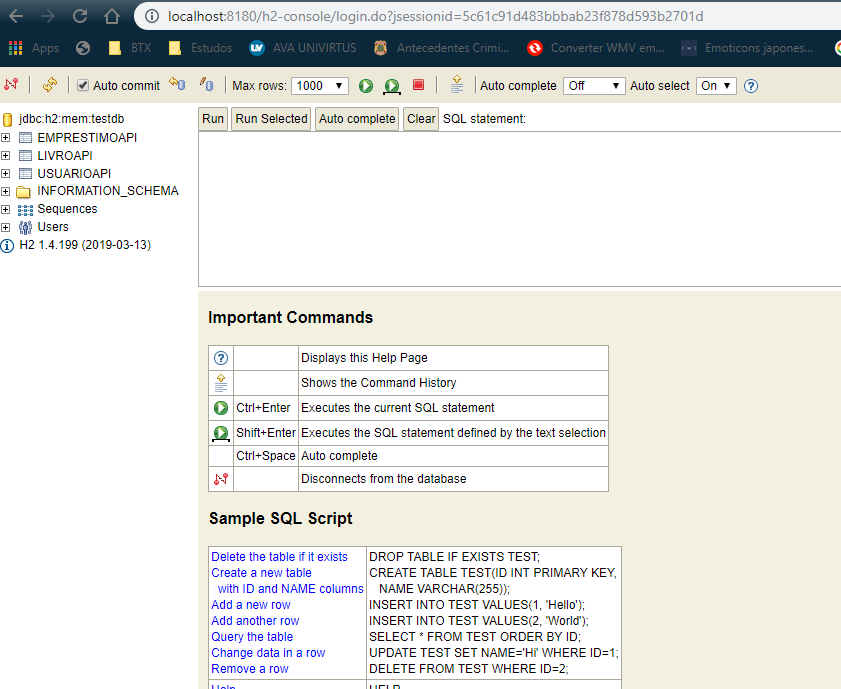
Abrir o navegador para visualizar o banco <http://localhost:8180/h2-console>

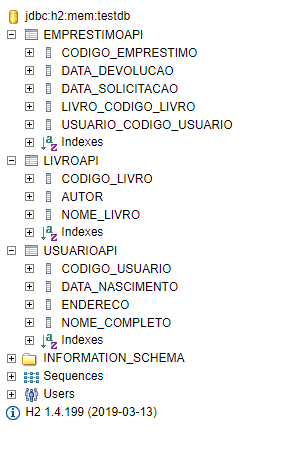


Mudar a url do jdbc para jdbc:h2:mem:testdb

Clicar em test connection e depois em connect

O que foi criado no eclipse passou para o banco

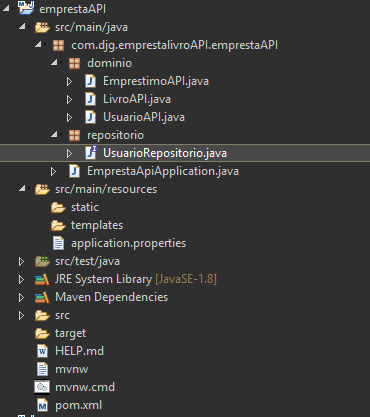




No caso do livro será dado o código automaticamente, pois concatena o nome do livro e o código.

Criar repositório

Dentro da aplicattion cria um package e depois cria uma interface



1. Repositor > Interface
2. Service>serviço
3. Controller>Controler

**Fluxo de acesso ao banco de dados**

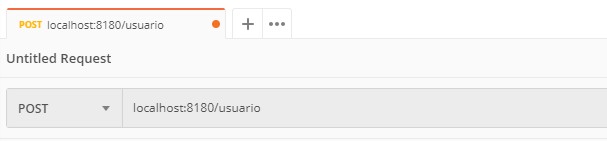
Front end > Controller (determina gets e posts)> Service(contem regras de negocio) > Repository(acesso o banco de dados) > Banco de dados

Para fazer o código tem que fazer da direita pra esquerda

Roda a aplicação.

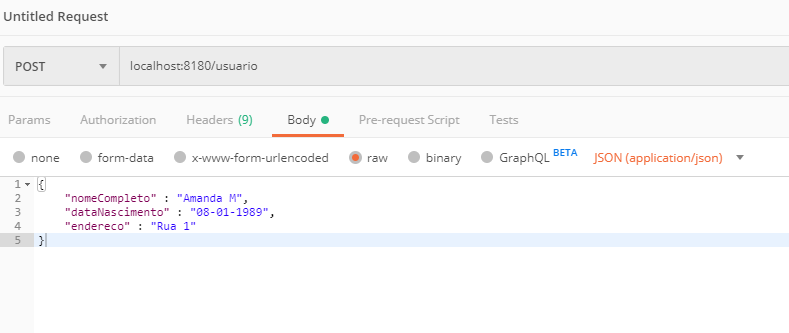
**No postman**

Criar um local host como post



**Para colocar o Jason vai em body>raw]**

**Tem que ser igual que está na classe usuário do domínio . selecionar o tipo de variável e deixa JSon aplication**



**Para consultar no banco, abre o h2, clica em usuário e aperta play**

