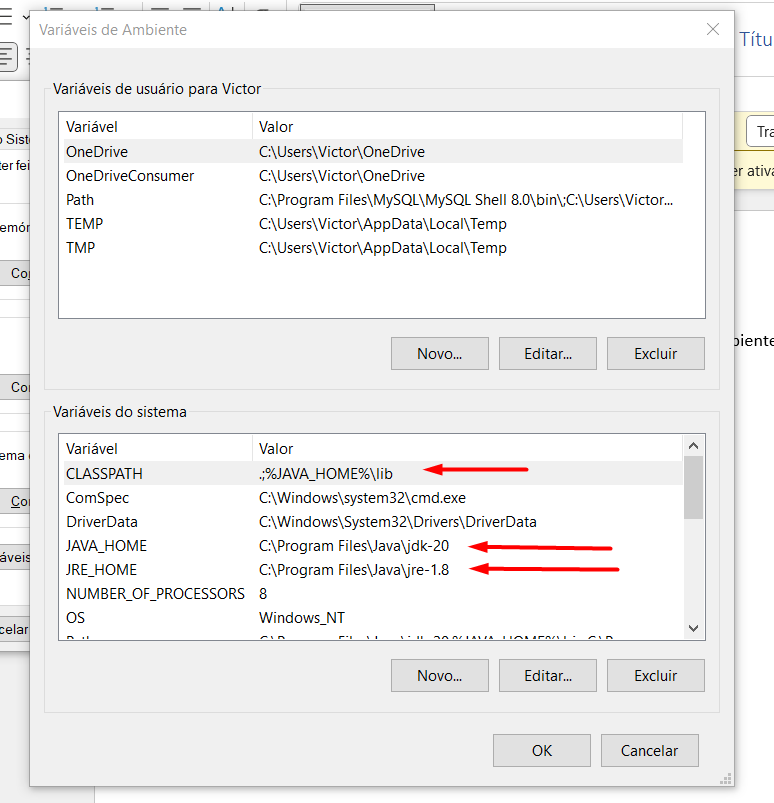
Contraints = restrições

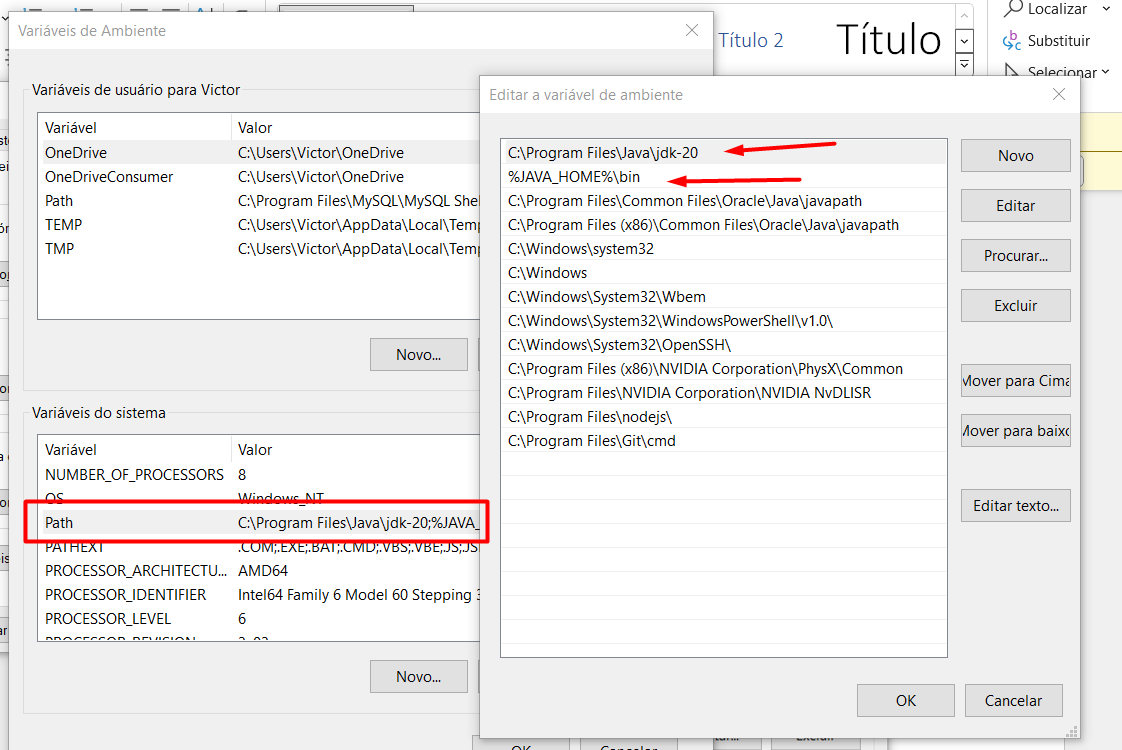
Classe pessoa é uma ABSTRAÇÃO

2.1 - **Criando o projeto Spring Boot Rest API com Maven**

Após instalação do JAVA é necessário configurar as variaveis de ambiente:



ADICIONAR EM PATH 2 LINHAS DE CONFIGURAÇÃO



Configuração do Gmail para envio, é necessário fazer essa configuração para habilitar o servidor de E-mail.

Segue o Vídeo, deu certo de primeira: <https://www.youtube.com/watch?v=gXYpl-9zpqs>

DTO é uma classe onde você pode carregar qualquer tipo de dados, diferente das classes de modelo.

**@Transactional:** Uma situação onde tenho múltiplas transações: um update, um insert e um delete dentro do mesmo serviço e há dependência entre as operações e assim se um deles falhar o **@Transactional** garante a integridade e desfaz todas as operações.

OBS: Utilizar dentro do Repository

**@Bean:** Componentes gerenciados pelo proprio Spring

**@Autowired:** Cria a injeção de depencia (construtor)

OBS: Para uma melhor pratica, Não utilize o @Autowired e sim criar o construtor manualmente ex:

private final CategoriaProdutoRepository categoriaProdutoRepository;  
  
public CategoriaProdutoController(CategoriaProdutoRepository categoriaProdutoRepository) {  
 this.categoriaProdutoRepository = categoriaProdutoRepository;  
}

**@Modifying(flushAutomatically = true, clearAutomatically = true):** force o commit no banco de dados e limpa o cache.

**AWS s3**

s3 – Armazenador de Objeto (PDF’s, imagens, Front-ent Angular, etc..)

Bucket – Pastas dentro do S3

AWS ec2

Ecs – Gerador de instância, um computador na Nuvem

AWS RDS

RDS – serviço de banco de dados

SUBINDO APLICAÇÃO JAVA para o AWS:

Mesmo seguindo passo-a-passo do curso não foi possivel seguir, o ElasticBeansTalk não funcionada e acusava diversos erros de AutoScaling, tive que subir a aplicação .jar direto no Ec2.

Criei um Instancia ec2 igual ao que mostra no Curso, MUITO IMPORTANTE CRIAR A CHAVE SSH E SALVAR (Será muito importante para subir o .jar na máquina), depois é aconselhavel seguir a demonstração da Fernanda kipper de Como subir a aplicação para o EC2. Pontos Importante do Video: Criação VCP, Criação do grupo de segurança e instalação dos recursos.

<https://www.youtube.com/watch?v=bEkCdlrxF54&t=1805s>

OBS: No momento que ela utiliza a chave SSH, você devera utilizar a que foi gerado no momento da criação da instancia EC2.

Comando executados:

Se conectar a maquina AWS via SSH: Copie o comando na aba de conexão SSH e vá na pasta da sua maquina que está o arquivo ssh.pem e execute o comando, ssh -i "chave-ssh-projeto-2.pem" [ec2-user@ec2-18-217-114-253.us-east-2.compute.amazonaws.com](mailto:ec2-user@ec2-18-217-114-253.us-east-2.compute.amazonaws.com)

Subir o .jar para a máquina AWS: scp -i "C:\Users\Victor\Downloads\chave-ssh-projeto-2.pem" C:\Users\Victor\Desktop\Projeto\_SpringBoot\Loja\_virtual\Loja\_virtual\target\loja\_virtual.jar ec2-user@18.217.114.253:/home/ec2-user

Comando para instação do Apache(Executar apenas em caso de não funcionar a configuração padrão que a Fernanda Kipper mostra no video):

sudo su

yum update -y

yum install -y httpd.x86\_64

systemctl start httpd.service

systemctl enable httpd.service

echo "Hello World from $(hostname -f)" > /var/www/html/index.html

**Usando o PID do Processo Java**

1. **Listar o PID do Processo Java**:
   * Primeiro, liste o processo Java em execução para obter o PID (identificador do processo): ps aux | grep java
   * Isso mostrará uma lista de processos Java em execução. Identifique o PID do processo que deseja parar.
2. **Matar o Processo**:
   * Depois de identificar o PID, use o comando kill para parar o processo: kill -9 PID
   * Substitua PID pelo número do processo que deseja parar.

**Nota**: O sinal -9 força o encerramento do processo. Use com cuidado, pois ele encerra imediatamente o processo sem dar tempo para que ele realize uma parada limpa.