**O que é estrutura de dados?**

Estrutura de dados é uma estrutura organiza de dados na memória de um computador ou em qualquer dispositivo de armazenamento, de forma que os dados possam ser utilizados de forma correta.

Essas estruturas encontram muitas aplicações no desenvolvimento de sistemas, sendo que algumas são altamente especilizados e utilizados em tarefas específicas.

Usando as estruturas adequadas através de algorítmos, podemos trabalhar com uma grande quantidade de dados, como aplicações em bancos de dados ou serviços de busca.

Em uma estrutura de dados devemos saber como realizar um determinado conjunto de operações, como por exemplo:

inserir dados

excluir dados

localizar um elemento

percorrer todos os itens constituintes da estrutura para visualização

classificar, que se resume em colocar os itens de dados em uma determinada ordem (numérica, alfabética, etc.)

**Principais Estruturas de Dados:**

Vetores e matrizes

Registro

Lista

Pilha

Fila

Árvore

Tabela hash

Grafos

**Vetores e Matrizes**

Vetores e Matrizes ou Arrays são estruturas de dados simples que podem auxiliar quando há muitas variáveis do mesmo tipo em um algorítmo.

<https://portugol-webstudio.cubos.io/ide>

Vetor ou array uni-dimensional é uma variável que armazena várias variáveis do mesmo tipo.

O vetor é uma estrutura de dados indexada, que pode armazenar uma determinada quantidade de valores do mesmo tipo.

Exemplo:

programa

{

funcao inicio()

{

inteiro numeros[] = {39, 45, 54, 55}

//escreva(numeros[0])

//imprime apenas o índice a que está no colchetes []

para (inteiro posicao = 0; posicao <= 3; posicao++){

escreva(numeros[posicao], " ")

//imprime todos os índices do vetor

//" " imprime um espaço entre os valores

}

}

}

**Matrizes**

Matriz ou array multi-dimensionais é um vetor de vetores

Uma matriz é um vetor que possui duas ou mais dimensões

**Exemplo: matriz de tamanho 5 (5 linhas e 5 colunas)**

programa

{

inclua biblioteca Util --> u

funcao inicio()

{

// Define as dimensões (linhas e colunas) da matriz

const inteiro TAMANHO = 5

// Cria a matriz

inteiro matriz[TAMANHO][TAMANHO]

para (inteiro linha = 0; linha < TAMANHO; linha++)

{

para (inteiro coluna = 0; coluna < TAMANHO; coluna++)

{

matriz[linha][coluna] = u.sorteia(1, 9) // Atribui um valor aleatório à posição da matriz

escreva("[", matriz[linha][coluna], "]") // Exibe o valor contido na posição da matriz

}

escreva ("\n")

}

}

}

**Registro**

Um registro é uma estrutura que fornece um formato especializado para armazenar informações em memória.

Enquanto Arrays nos permitem armazenar vários dados de um único tipo de dados, o recurso de Registro nos permite armazenar mais de um tipo de dado.

Um registro é composto por campos que especificam cada uma das informações que o compõem

Exemplo de campos que constituem o registro de um cliente:

|  |  |
| --- | --- |
| CPF |  |
| Nome |  |
| Endereço |  |
| Contato |  |

Toda estrutura de registro tem um nome (ex.:livro), e seus campos podem ser acessados por meio do uso do operador ponto (.). Por exemplo, para acessar o preço de um livro, poderíamos utilizar a seguinte declaração:

**livro.preco**

Programa que declara registro e armazena informações de três livros

ALGORÍTMO

//declaracap do tipo de dado

tipo

estrutura\_livro = registro

nome : caracter

preco : real

pagina : inteiro

fimregistro

//declaração das variáveis

i inteiro

livro array[1..3] de estrutura\_livro

Escreva(“Entre com os nomes, preços e números de páginas de três livros”)

para i de ate 3 faca //leitura dos dados

Leia(livro[i].nome, livro[i].preco, livro[i].pagina)

fimpara

Escreva(“Esses foram os dados digitados”)

para i de 1 ate 3 faca

Escreva(livro[i].nome, livro[i].preco, livro[i].pagina)

fimpara

FIMALGORITMO

**O que são Listas, Pilhas e Filas**

**Listas**

Estrutura de dados do tipo Lista, armazena dados de um determinado tipo em uma ordem específica.

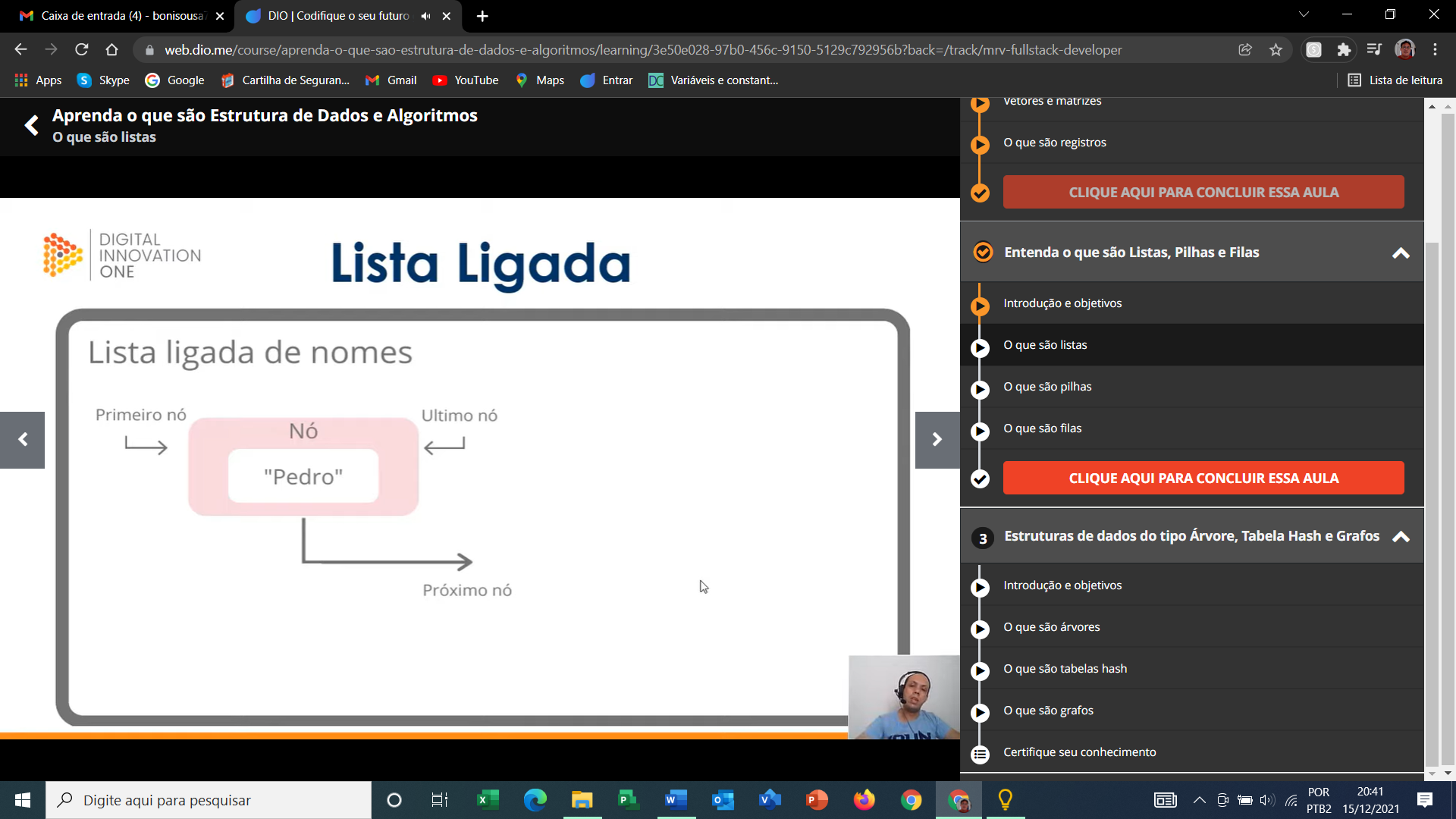
A diferença entre listas e arrays é a de que as listas possuem tamanho ajustável, enquanto arrays possuem tamanho fixo.

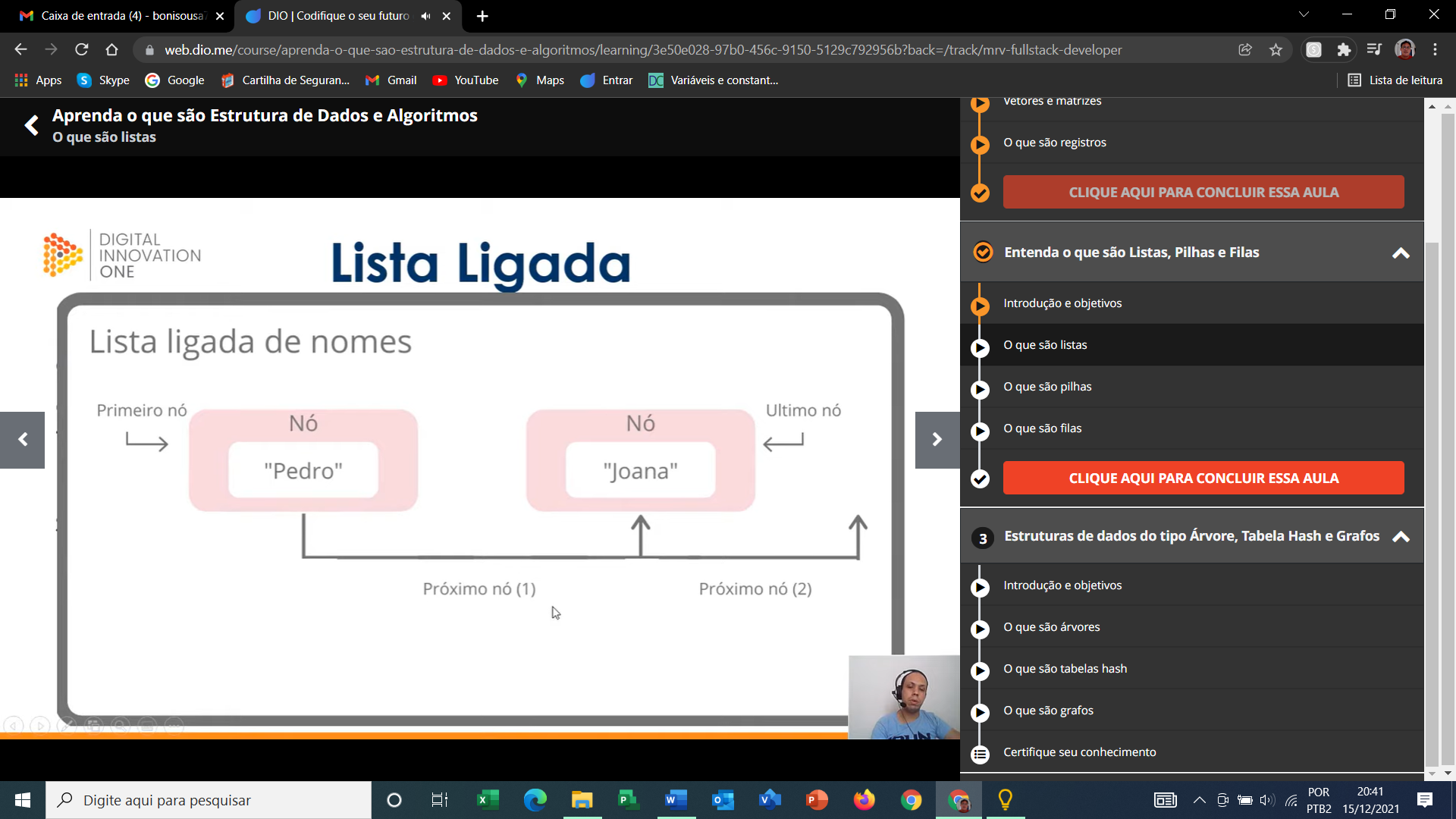
**Existem dois tipos de listas: as Ligadas e as Duplamente Ligadas**

**Lista Ligada**

Na estrutura do tipo lista ligada existem os nós onde cada uma dos nós conhece o valor que está sendo armazenado em seu interior além de conhecer o elemento posteriora a ele: por isso ela á chamada de **“lista ligada”**, pois os nós são amarrados com essa indicação de qual é o próximo nó.

**Lista ligada de nomes**



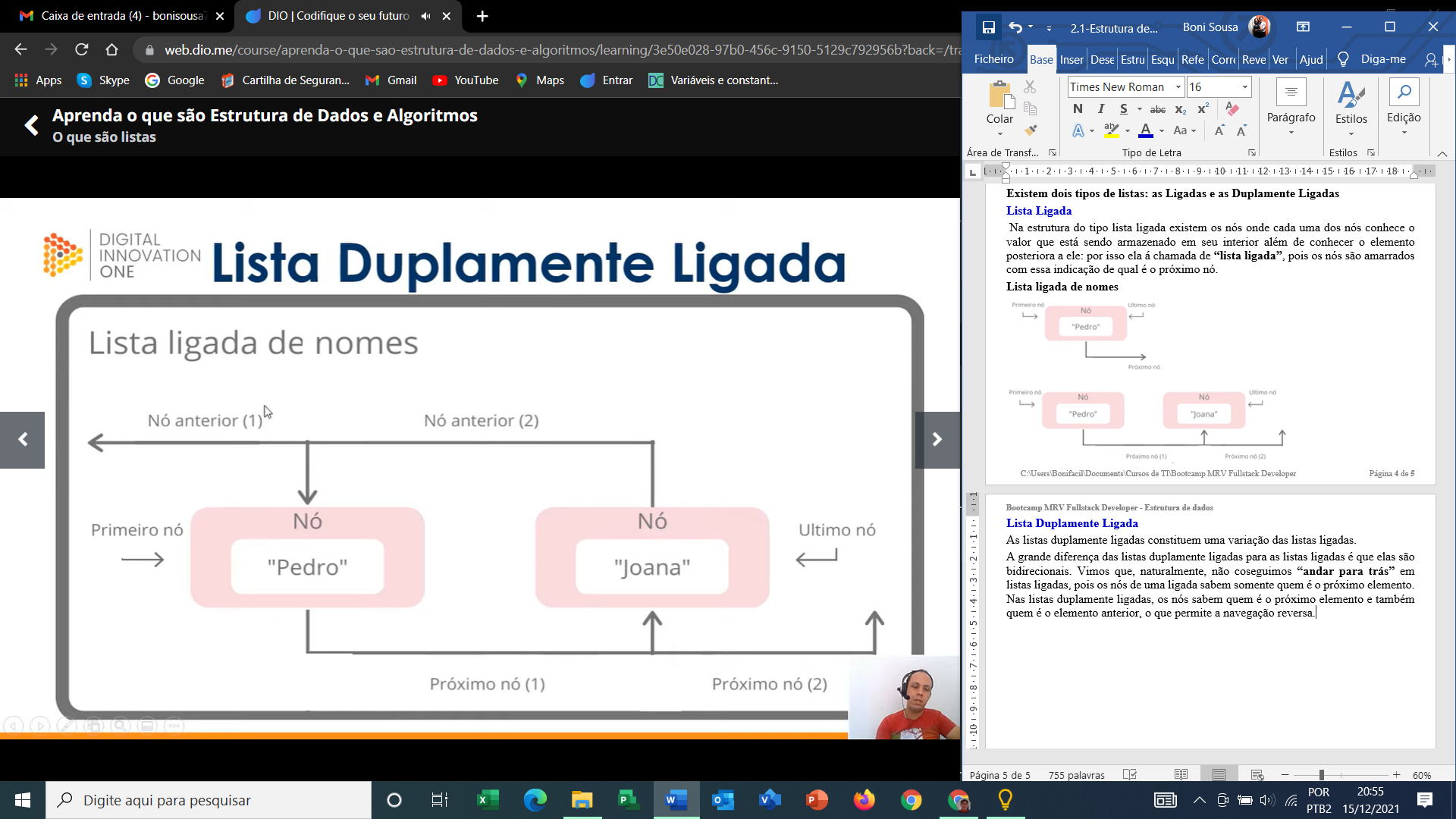


**Lista Duplamente Ligada**

As listas duplamente ligadas constituem uma variação das listas ligadas.

A grande diferença das listas duplamente ligadas para as listas ligadas é que elas são bidirecionais. Vimos que, naturalmente, não coseguimos **“andar para trás”** em listas ligadas, pois os nós de uma ligada sabem somente quem é o próximo elemento. Nas listas duplamente ligadas, os nós sabem quem é o próximo elemento e também quem é o elemento anterior, o que permite a navegação reversa.

**Lista duplamente ligada de nomes**



**Pilhas**

O acesso aos itens de uma pilha é restrito, somente um item pode ser lido ou removido por vez.

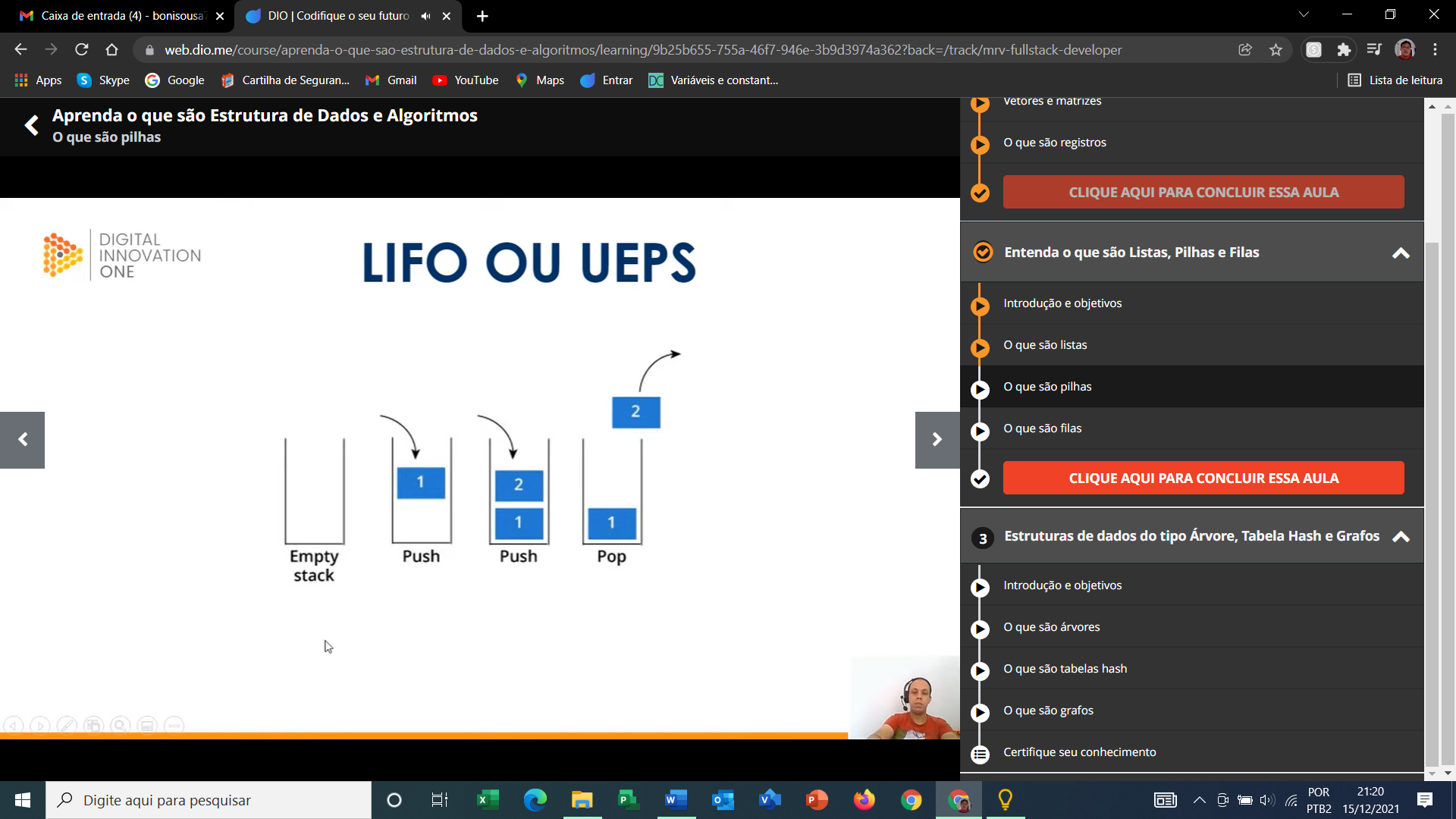
Os tipos de pilhas são:

LIFO OU UEPS - FIFO OU PEPS - também chamados de **stack**

**LIFO OU UEPS**

A estrutura do tipo **PILHA LIFO** (Last in First Out) ou UEPS **(Último que Entra Primeiro que Sai)**, apresenta o seguinte critério: o primeiro elemento a ser retirado é o último que tiver sido inserido

Representação gráfica de uma **LIFO OU UEPS**



Push = entrada >>> Pop = saída

**FIFO OU PEPS**

A estrutura do tipo **PILA FIFO** (First in First Out) ou PEPS **(Primeiro que Entra Primeiro que Sai)**, apresenta o seguinte critério: o primeiro elemento a ser retirado é o primeiro que tiver sido inserido.

Representação gráfica de uma **FIFO OU PEPS**

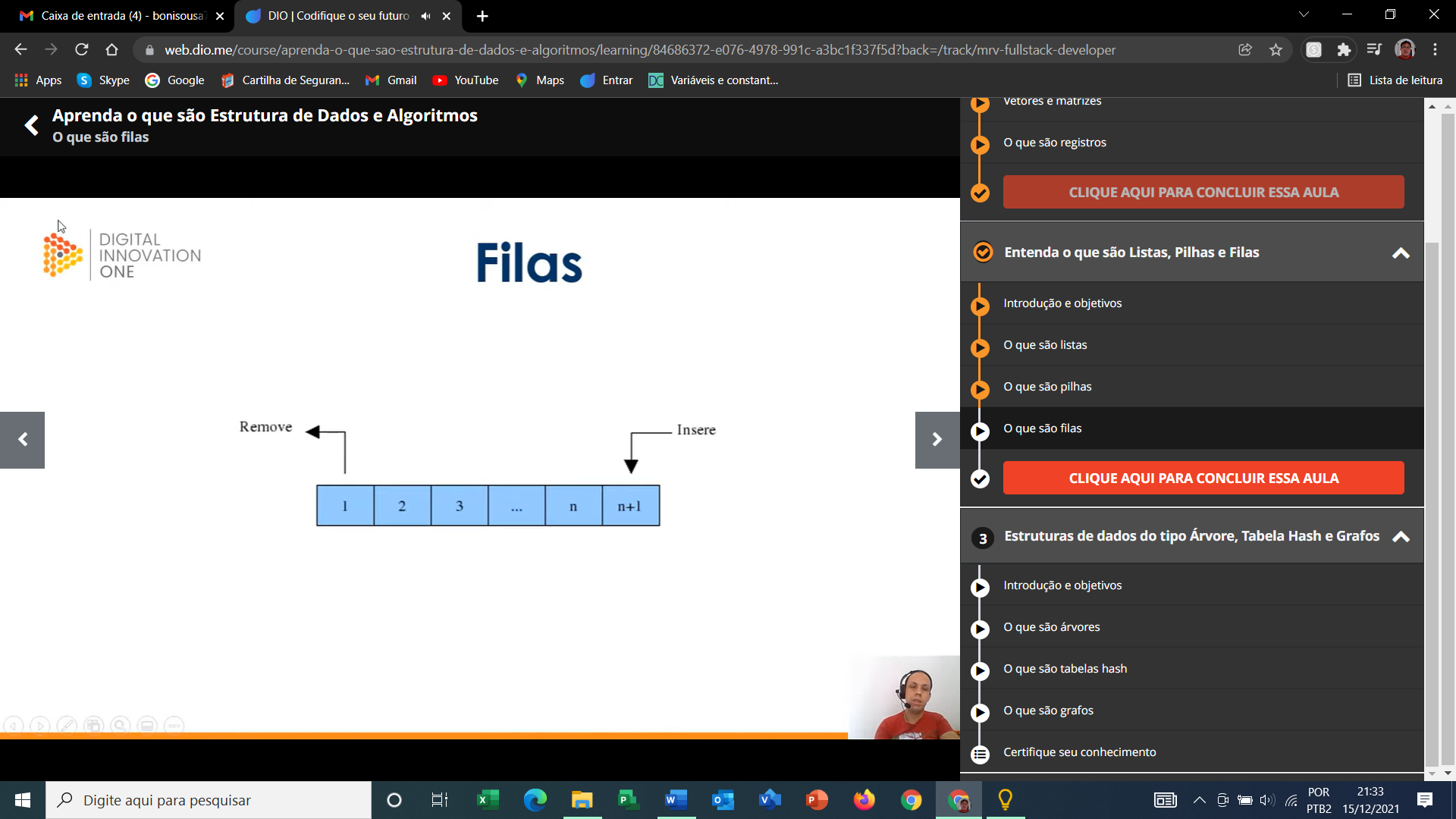


**As Filas**

A estrutura do tipo Fila admite remoção de elementos e inserção de novos sujeitos à seguinte regra de operação:

o elemento removido é o que está na estrutura há mais tempo, ou seja, o primeiro objeto inserido na fila é também o primeiro a ser temovido seguindo o conceito FIFO.

Representação gráfica de uma **Fila**

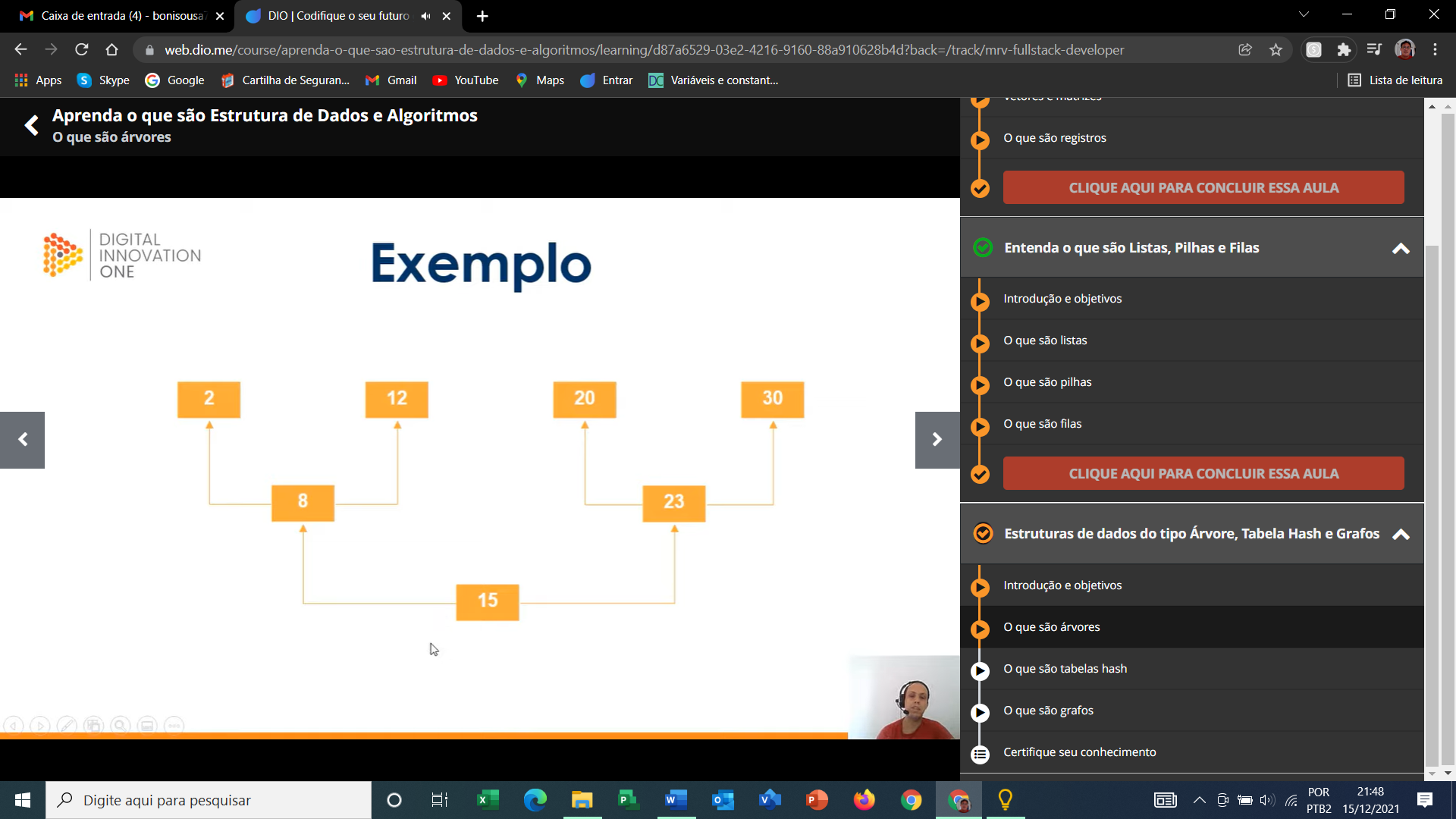


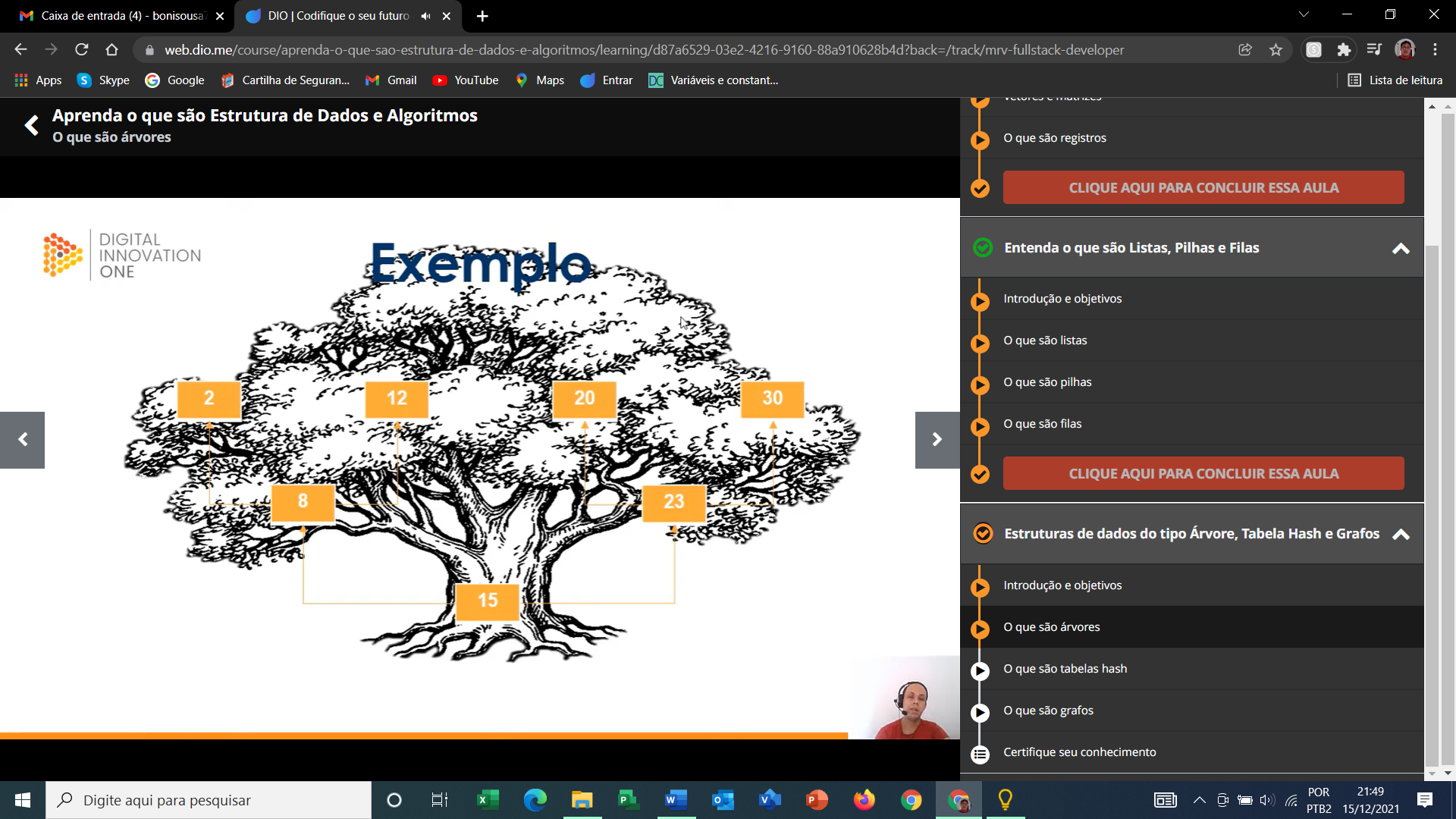
**Estruturas de dados do tipo Árvore, Tabela Hash e Grafos**

**Estrutura de dados do tipo Árvore**

É uma estrutura de dados que organiza seus elementos de forma hierárquica, onde existe um elemento que fica no topo da árvore, chamado de **raiz** e existem os elementos subordinados a ele, que são chamados de nós ou folhas.

Representação gráfica de uma **Estrutura de dados do tipo Árvore**



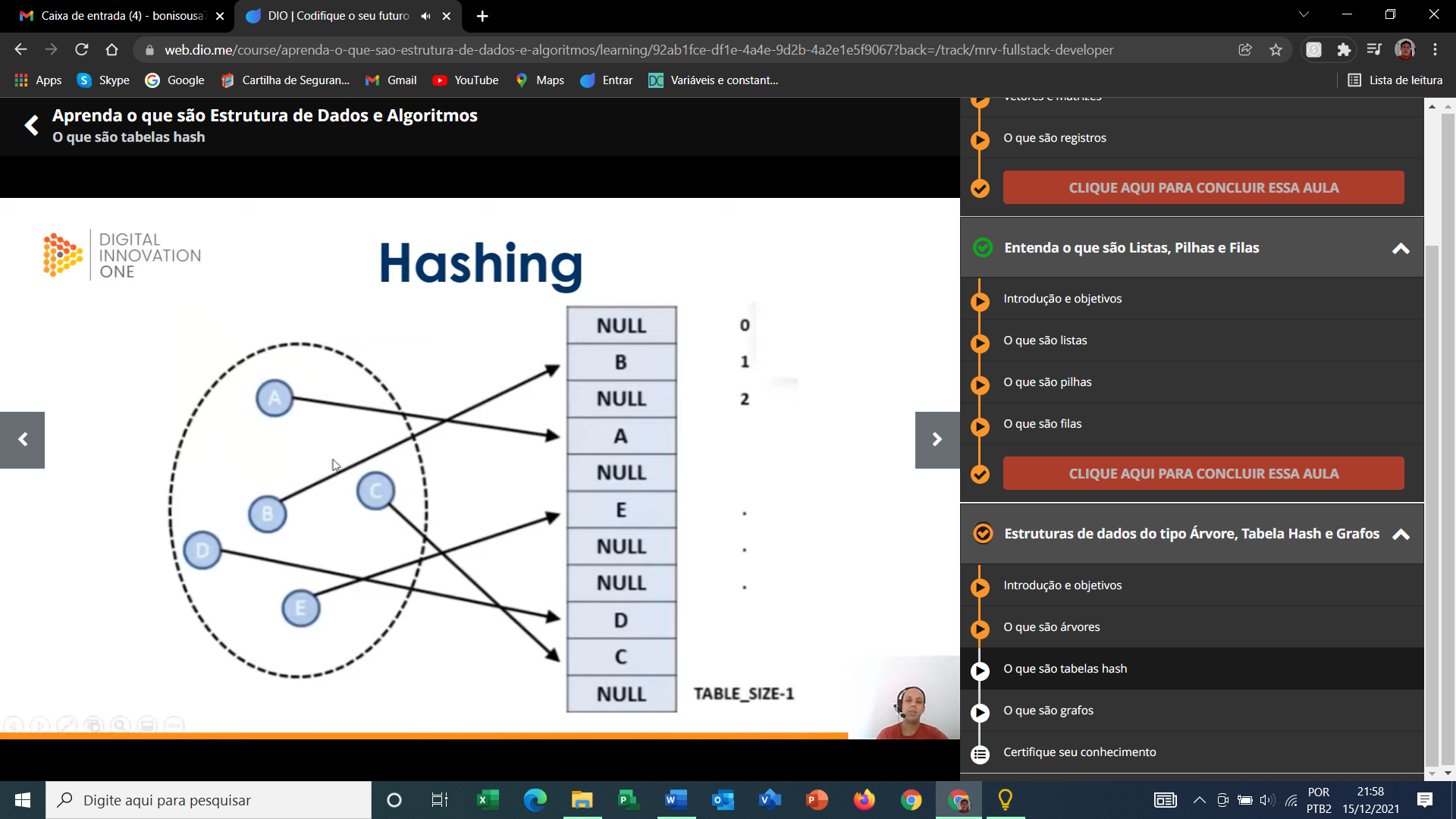


**Estrutura de dados do tipo Tabela Hash**

Uma **tabela hash**, de dispersão ou espalamento é uma estrutura de dados especial, que associa chaves de pesquisa a valores.

Uma tabela hash é uma generalização da ideia de array, porém utiliza uma função denominada **Hashing** para espalhar os elementos, fazendo com que os mesmos fiquem de forma não ordenada dentro do **“array”** que define a tabela.

Representação gráfica de uma **Estrutura de dados do tipo Tabela Hash**



**Porque espalhar?**

A tabela hash permite a associação de “valores” a “chaves”

**Valores:** é a posição ou índice onde o elemento se encontra

**Chaves:** parte da informação que compõe o elemento a ser manipulado

Espalhar facilita a busca na estrutura de dados, pois a partir de uma chave podemos acessar de forma rápida uma posição do **“array”**

**Estruturas de dados do tipo Grafos**

Grafos são estruturas que permitem programar a relação entre objetos

Os **objetos** são vértices ou **“nós”** do grafo

Os **relacionamentos** são arestas

Representação gráfica de uma **Estrutura de dados do tipo Grafos**

