CENTRO UNIVERSITÁRIO FANOR WYDEN
UNIFANOR WYDEN

JOÃO BATISTA RAFAEL DA SILVA

Atividade Avaliativa: Desenvolvimento de Software Integrada

Fortaleza/CE
2022

SUMÁRIO

1 ARQUITETURA WEB

A arquitetura de qualquer aplicativo é o esqueleto e a espinha dorsal do aplicativo que suporta fundamentalmente o aplicativo. Ele descreve diferentes interações que ocorrem entre componentes, middleware, bancos de dados e interfaces de usuário. Devido a essas interações, os aplicativos trabalham juntos. A arquitetura lida com o fluxo e a comunicação de todo o aplicativo e lida com os principais aspectos e componentes do produto.

A arquitetura consiste em componentes com uma descrição lógica. Ele determina o design futuro do aplicativo, a experiência do usuário, a infraestrutura e os módulos de software. Portanto, trabalhar na arquitetura é o primeiro passo para o desenvolvimento de um produto.

1.1 Fundamentos

As aplicações da Web têm dois lados: front-end e back-end. Diz-se que o front-end de um aplicativo da Web é do lado do cliente, que interage com os usuários. O front-end é escrito em HTML, CSS e JS. Portanto, quando você abre um aplicativo da Web, tudo o que você vê e interage fica na parte de front-end. Considerando que o back-end está no lado do servidor e inacessível pelos usuários. No lado do servidor, os dados são armazenados que podem ser acessados e manipulados facilmente. As solicitações HTTP do back-end usam os dados recebidos pelos usuários na forma de imagens, texto e arquivos. O back-end pode ser facilmente desenvolvido usando JS, Java, Python, PHP, etc.

Sempre que o usuário visita uma página da web, o servidor web busca os dados e envia a resposta ao navegador com base nas solicitações do usuário. Assim que o navegador recebe a resposta, a interação começa. Todo esse processo inclui vários componentes e modelos, que exploraremos na próxima seção

1.2 Modelos Arquiteturais

1.2.1. Um banco de dados, um servidor web

Um banco de dados, um servidor web é o modelo mais básico e simples usado em modelos de arquitetura de aplicativos Web. Como o nome sugere, os desenvolvedores usam apenas um único banco de dados e servidor web para desenvolver o aplicativo. Este é um dos modelos mais utilizados para arquitetura de aplicativos móveis. Mas, tem uma limitação: se o

servidor estiver inativo, todo o aplicativo ficará inativo. Portanto, o modelo é menos preferido para aplicativos da Web devido à sua limitação.

1.2.2. Um banco de dados, vários servidores web

Um banco de dados, um modelo de vários servidores web, é melhor para aqueles aplicativos em que o armazenamento de dados não é feito usando servidores web. Em vez disso, os dados são processados pelo servidor web e armazenados no banco de dados. Esta é uma arquitetura sem estado que consiste em pelo menos dois servidores web. Ele é usado para evitar problemas improvisados que causam falhas. Por exemplo, sempre que um servidor falha por algum motivo, o segundo servidor toma seu lugar e executa as solicitações.

1.2.3. Vários bancos de dados, vários servidores web

O modelo de vários bancos de dados e vários servidores web é o modelo mais eficiente que garante nenhum ponto de falha, nem para servidores web nem bancos de dados. Os modelos contêm dois tipos de modelos de banco de dados que armazenam dados idênticos e são distribuídos uniformemente.

2 HTML (HYPER TEXT MARKUP LANGUAGE)

HTML é uma linguagem de marcação que é usada para criar páginas da web atraentes com a ajuda de estilo e que parece em um formato agradável em um navegador da web. Um documento HTML é feito de muitas tags HTML e cada tag HTML contém um conteúdo diferente.

Hyper Text (Hipertexto) significa simplesmente "Texto dentro do Texto". Um texto tem um link dentro dele, é um hipertexto. Sempre que você clica em um link que o leva a uma nova página da Web, você clicou em um hipertexto. Hyper Text é uma maneira de vincular duas ou mais páginas da Web (documentos HTML) entre si.

Markup language (Linguagem de marcação) é uma linguagem de computador usada para aplicar convenções de layout e formatação a um documento de texto. A linguagem de marcação torna o texto mais interativo e dinâmico. Ele pode transformar texto em imagens, tabelas, links, etc.

2.2 Exemplo de cadastro de formulário

Na codificação abaixo é um exemplo de formulário utilizando HTML e CSS.

```
<!DOCTYPE html>
<html>
<head>
   <meta charset='utf-8'>
   <meta http-equiv='X-UA-Compatible' content='IE=edge'>
   <title>Exemplo Formulário</title>
   <style>
      body {
         display: flex;
         align-items: center;
         justify-content: center;
         height: 100vh;
         font-family: cursive;
      }
      #name, #email, #enviar{
         width: 100%;
```

```
            padding: 5px;
            outline: none;
        }
        label {
            line-height: 1.9rem;
        }
        input[type="submit"] {
            transform: translate(2.2%);
            padding: 3px;
            margin-top: 0.6rem;
            font-family: cursive;
            font-weight: bold;
        }
        fieldset {
            padding: 20px 40px;
        }
    </style>
</head>
<body>
    <form action="backend">
        <fieldset>
            <div>
                <label>Name:</label>
                <input type="name" id="name" placeholder="Nome Completo" required />
            </div>
            <div>
                <label>Email:</label>
                <input type="email" id="email" placeholder="exemplo@exemplo.com" required />
            </div>
            <div>
                <label for="Sexo">Sexo:</label>
                <input type="radio" name="gender" value="mas" id="mas">
                <label for="male">Masculino</label>
                <input type="radio" name="gender" value="fem" id="fem">
```

```
            <label for="female">Feminino</label>
        </div>
        <div>
            <input type="submit" id="enviar" value="Enviar" />
        </div>
    </fieldset>
  </form>
</body>
</html>
```

3 DIFERENÇAS ENTRE SERVLETS E SCRIPTS CGI

Ambos os servlets Java e CGI são usados para criar aplicativos web dinâmicos que aceitam uma solicitação do usuário, processam-na no lado do servidor e retornam respostas ao usuário. No entanto, os servlets Java oferecem uma série de vantagens sobre o CGI tradicional, que são as seguintes:

1. Eficaz: Ao contrário do CGI tradicional, onde um novo processo é iniciado para cada solicitação do cliente, um servlet processa cada solicitação como uma thread dentro de um processo. Assim, os servlets melhoram o desempenho, pois removem a sobrecarga de criar um novo processo para solicitação a cada vez.
2. Poderoso: Servlets suportam vários recursos que são difíceis ou impossíveis de realizar com CGI tradicional. Esses recursos incluem falar diretamente com o servidor da Web, compartilhar dados entre vários servlets, rastreamento de sessão e armazenamento em cache de cálculos anteriores.
3. Dependência de idioma: Como os servlets podem ser escritos apenas em Java, eles dependem do idioma. Por outro lado, os programas CGI podem ser escritos em qualquer linguagem de programação como C, C++, Perl, Visual Basic ou mesmo Java, portanto, são independentes da linguagem.
4. Seguro: Como o Java foi projetado desde o início como uma linguagem segura, um servlet pode ser executado por um mecanismo de servlet ou contêiner de servlet em uma sandbox restritiva, assim como um applet é executado na JVM de um navegador da Web, o que aumenta a segurança do servidor. Por outro lado, os scripts CGI não estão sujeitos ao mesmo grau de sandboxing de segurança que os servlets Java, eles são significativamente menos seguro

4 INTRODUÇÃO AO JSP

JSP significa Java Server Pages é uma tecnologia para construir aplicações web que suportam conteúdo dinâmico e atua como uma tecnologia de servlet Java.

A tecnologia JSP é usada para criar aplicativos web dinâmicos. As páginas JSP são mais fáceis de manter do que um Servlet. Páginas JSP são o oposto de Servlets, pois um servlet adiciona código HTML dentro de código Java, enquanto JSP adiciona código Java dentro de HTML usando tags JSP. Tudo o que um Servlet pode fazer, uma página JSP também pode fazer.

JSP nos permite escrever páginas HTML contendo tags, dentro das quais podemos incluir poderosos programas Java. Usando JSP, pode-se facilmente separar a lógica de apresentação e de negócios, pois um WEB designer pode projetar e atualizar páginas JSP criando a camada de apresentação e o desenvolvedor JAVA pode escrever código computacional complexo do lado do servidor sem se preocupar com o design da web. E ambas as camadas podem interagir facilmente por meio de solicitações HTTP.

5 INTRODUÇÃO AOS PADRÕES DE PROJETOS

Padrões de projetos são modelos de código que resolvem problemas clássicos. São soluções para problemas de design de software que você pode encontrar em um aplicativo do mundo real. Um padrão de projeto não é um código pronto para ser usado em seu aplicativo, mas é um modelo que você pode usar para resolver um problema.

Os padrões de projetos são neutros em termos de linguagem, portanto, podem ser aplicados a qualquer linguagem que suporte orientação a objetos.

Um padrão bem conhecido é MVC (Model View Controller). O MVC propõe uma divisão da aplicação entre 3 subsistemas: Model, onde os dados são encapsulados; View, em que os dados são recebidos pelo Model e mostrados ao cliente; e Controller, que faz o intermédio entre os outros dois subsistemas em operações de leitura, escrita e modificação de dados. Esta divisão proporciona uma melhor manutenibilidade do sistema, em que alterações podem ser feitas em uma das partes sem alterar as outras, podendo por exemplo alterar a interface de uma aplicação sem precisar alterar o Model.

6 INTRODUÇÃO AO ORIENTAÇÃO A OBJETOS

A programação orientada a objetos (OOP) é um modelo de programação de computador que organiza o design de software em torno de dados ou objetos, em vez de funções e lógica. Um objeto pode ser definido como um campo de dados que possui atributos e comportamento exclusivos. A POO se concentra nos objetos que os desenvolvedores desejam manipular, em vez da lógica necessária para manipulá-los. Essa abordagem de programação é adequada para programas grandes, complexos e atualizados ou mantidos ativamente.

A estrutura, ou blocos de construção, da programação orientada a objetos incluem o seguinte: Classes, são tipos de dados definidos pelo usuário que atuam como o modelo para objetos, atributos e métodos individuais; Objetos, são instâncias de uma classe criada com dados especificamente definidos. Os objetos podem corresponder a objetos do mundo real ou a uma entidade abstrata; Métodos, são funções definidas dentro de uma classe que descrevem os comportamentos de um objeto; Atributos, são definidos no modelo de classe e representam o estado de um objeto. Os objetos terão dados armazenados no campo de atributos;

A programação orientada a objetos é baseada em alguns princípios, são eles: Herança, consiste em uma classe tem a capacidade de derivar propriedades e características de outra classe; Encapsulamento, refere-se à criação de módulos autocontidos (classes) que vinculam funções de processamento a seus membros de dados. Os dados dentro de cada classe são mantidos privados. Cada classe define regras para o que é visível publicamente e quais modificações são permitidas; Polimorfismo, está relacionado aos objetos que são projetados para compartilhar comportamentos e podem assumir mais de uma forma. O programa determinará qual significado ou uso é necessário para cada execução desse objeto de uma classe pai, reduzindo a necessidade de duplicar o código.

7 INTRODUÇÃO A LINGUAGEM UML

UML, abreviação de Unified Modeling Language, é uma linguagem de modelagem padronizada que consiste em um conjunto integrado de diagramas, desenvolvido para ajudar os desenvolvedores de sistemas e software a especificar, visualizar, construir e documentar os artefatos de sistemas de software, bem como para modelagem de negócios e outros sistemas não-software. A UML representa uma coleção das melhores práticas de engenharia que provaram ser bem-sucedidas na modelagem de sistemas grandes e complexos.

A UML contém dois tipos de diagramas, são eles: Diagramas de estrutura, que contempla diagrama de classe, componentes, implantação, objeto, pacote, estrutura composta, perfil; Diagramas de comportamento, que contempla diagrama de casos de uso, atividade, máquina de estado, sequência, comunicação, visão geral de interação, temporização.

8 MODELAGEM DE CONTROLE DE ESTOQUE

9 INTRODUÇÃO A JSF (JAVA SERVER FACES)

JavaServer Faces (JSF) é a tecnologia padrão Java para construir interfaces web baseadas em componentes e orientadas a eventos. Assim como o JavaServer Pages (JSP), o JSF permite acesso a dados e lógica do lado do servidor. Ao contrário do JSP, que é essencialmente uma página HTML imbuída de recursos do lado do servidor, o JSF é um documento XML que representa componentes formais em uma árvore lógica. Os componentes JSF são suportados por objetos Java, que são independentes do HTML e possuem toda a gama de recursos Java, incluindo acesso remoto a APIs e bancos de dados.

O JSF oferece facilidade de uso das seguintes maneiras: Facilita a construção de uma interface do usuário a partir de um conjunto de componentes de interface do usuário reutilizáveis; simplifica a migração de dados do aplicativo de e para a interface do usuário; ajuda a gerenciar o estado da interface do usuário nas solicitações do servidor; fornece um modelo simples para conectar eventos gerados pelo cliente ao código do aplicativo do lado do servidor; permite que componentes de interface do usuário personalizados sejam facilmente construídos e reutilizados.

10 APRESENTAÇÃO DOS CONCEITOS ORM, JPA E HIBERNATE

ORM é uma biblioteca com a qual você pode criar objetos para suas entidades. Por exemplo, as entidades de um aplicativo de comércio eletrônico podem ser usuários, pedidos e produtos. Essas entidades serão convertidas em tabelas por um ORM. Um ORM gerará consultas SQL para você e se comunicará com seu banco de dados.

A especificação JPA permite definir quais objetos devem ser persistidos e como eles são persistidos em seus aplicativos Java. Por si só, o JPA não é uma ferramenta ou estrutura; em vez disso, define um conjunto de conceitos que orientam os implementadores. Embora o modelo de mapeamento relacional de objeto (ORM) do JPA tenha sido originalmente baseado no Hibernate, ele evoluiu desde então. Da mesma forma, embora o JPA tenha sido originalmente destinado ao uso com bancos de dados relacionais, algumas implementações do JPA foram estendidas para uso com datastores NoSQL.

Hibernate é um framework em Java que vem com uma camada de abstração e trata as implementações internamente. As implementações incluem tarefas como escrever uma consulta para operações CRUD ou estabelecer uma conexão com os bancos de dados, entre outros. O Hibernate desenvolve lógica de persistência, que armazena e processa os dados para uso mais longo. É leve e uma ferramenta ORM e, mais importante, de código aberto, o que lhe dá uma vantagem sobre outros frameworks.

11 INTRODUÇÃO A ESTRUTURA DE DADOS

As estruturas de dados são as formas de organizar e armazenar dados em um computador para que possamos realizar várias operações com eficiência nele. É amplamente utilizado em todos os aspectos da ciência da computação. Alguns exemplos de técnicas comumente usadas para organizar dados são: Matriz, Pilha, Fila, Lista encadeada, Árvore, Fila de prioridade, Tabela de hash, Gráfico.

Uma pilha é uma estrutura de dados linear na qual os elementos podem ser inseridos e excluídos apenas de um lado da lista, chamado top. Uma pilha segue o princípio LIFO (Last In First Out), ou seja, o elemento inserido por último é o primeiro elemento a sair.

Uma fila é uma estrutura de dados linear na qual os elementos podem ser inseridos apenas de um lado da lista chamado de back, e os elementos podem ser excluídos apenas do outro lado chamado de front .A estrutura de dados da fila segue o princípio FIFO (First In First Out), ou seja, o elemento inserido primeiro na lista, é o primeiro elemento a ser removido da lista.

11.1 Exemplo Pilha em JAVA

Exemplo de pilha feito em JAVA onde o mesmo tem as funcionalidades de inserir elemento, desempilhar, seleciona o último elemento, exibi o tamanho atual da pilha.

```
public class Main<Array> {

    public Object[] array;
    public int ponteiro;

    public Main() {
        this.ponteiro = -1;
        this.array = new Object[10];
    }

    public boolean isEmpty() {
        if (this.ponteiro == -1) {
            return true;
```

```
            }
            return false;
}

public int tamanho() {
            if (this.isEmpty()) {
                        return 0;
            }
            return this.ponteiro + 1;
}

public Object ultimoValor() {
            if (isEmpty()) {
                        return null;
            }
            return this.array[this.ponteiro];
}

public Object dessempilhar() {
            if (isEmpty()) {
                        return null;
            }
            return this.array[this.ponteiro--];
}

public void adicionar(Object valor) {
            if (this.ponteiro < this.array.length - 1) {
                        this.array[++ponteiro] = valor;
            }
}

public static void main(String[] args) {
            Main oMain = new Main();
            oMain.adicionar("A");
```

```
                oMain.adicionar("B");
                oMain.adicionar("C");
                oMain.adicionar("D");

                int tamanho = oMain.tamanho();
                Object ultimo = oMain.ultimoValor();

                System.out.println("Tamanho da pilha: " + tamanho);
                System.out.println("Ultimo elemento da pilha: " + ultimo);

        }

}
```

12 CONCEITOS DE ESTRUTURA DE DADOS ÁRVORE AVL

A árvore AVL é uma árvore de busca binária na qual a diferença de alturas das sub-árvores esquerda e direita de qualquer nó é menor ou igual a um. A técnica de equilibrar a altura de árvores binárias foi desenvolvida por Adelson, Velskii e Landi e, portanto, recebeu a forma abreviada de árvore AVL ou árvore binária balanceada. Como as árvores AVL são árvores de equilíbrio de altura, operações como inserção e exclusão têm baixa complexidade de tempo.

12.1 Exemplo Implementação JAVA

```
public class Main {

    public int getBalance(Node n) {
        if (n != null) {
            return (getHeight(n.left) - getHeight(n.right));
        }
        return 0;
    }

    public int getHeight(Node n) {
        if (n != null) {
            return n.height;
        }
        return 0;
    }

    public Node rightRotate(Node y) {
        Node x = y.left;
        Node T2 = x.right;

        x.right = y;
        y.left = T2;

        x.height = Math.max(getHeight(x.left), getHeight(x.right)) + 1;
        y.height = Math.max(getHeight(y.left), getHeight(y.right)) + 1;
```

```
        return x;
}

public Node leftRotate(Node x) {
        Node y = x.right;
        Node T2 = y.left;

        y.left = x;
        x.right = T2;

        x.height = Math.max(getHeight(x.left), getHeight(x.right)) + 1;
        y.height = Math.max(getHeight(y.left), getHeight(y.right)) + 1;

        return y;
}

public Node insert(Node node, int data) {
        if (node == null) {
                return (new Node(data));
        }
        if (node.data > data) {
                node.left = insert(node.left, data);
        } else {
                node.right = insert(node.right, data);
        }

        node.height = Math.max(getHeight(node.left), getHeight(node.right)) + 1;

        int balDiff = getBalance(node);

        if (balDiff > 1 && data < node.left.data) {
                return rightRotate(node);
        }
```

```
        if (balDiff < -1 && data > node.right.data) {
                return leftRotate(node);
        }

        if (balDiff > 1 && data > node.left.data) {
                node.left = leftRotate(node.left);
                return rightRotate(node);
        }

        if (balDiff < -1 && data < node.right.data) {
                node.right = rightRotate(node.right);
                return leftRotate(node);
        }

        return node;
}

public void inorder(Node root) {
        if (root != null) {
                inorder(root.left);
                System.out.print(" " + root.data);
                inorder(root.right);
        }
}

public static void main(String args[]) {
        Node oNode = null;
        Main oMain = new Main();

        oNode = oMain.insert(oNode, 4);
        oNode = oMain.insert(oNode, 2);
        oNode = oMain.insert(oNode, 1);
        oNode = oMain.insert(oNode, 5);
```

```
                oNode = oMain.insert(oNode, 6);
                oNode = oMain.insert(oNode, 9);
                oNode = oMain.insert(oNode, 14);

                oMain.inorder(oNode);

                System.out.print("\nA nova raiz da árvore AVL é: " + oNode.data);
        }

}

public class Node {
        int data;
        Node left;
        Node right;
        int height;

        public Node(int data) {
                this.data = data;
                height = 1;
        }
}
```

13 INTRODUÇÃO DE ESTRUTURA DE DADOS GRAFOS

Grafos em estruturas de dados são estruturas de dados não lineares compostas por um número finito de nós ou vértices e as arestas que os conectam. Grafos em estruturas de dados são usados para resolver problemas do mundo real nos quais representa a área do problema como uma rede, como redes telefônicas, redes de circuitos e redes sociais. Por exemplo, ele pode representar um único usuário como nós ou vértices em uma rede telefônica, enquanto o link entre eles via telefone representa arestas. Existem diferentes tipos de grafos, são eles: finito, infinito, trivial, simples, nulo, completo, regular, ponderado, entre outros.

14 DESCRIÇÃO DA PROBLEMÁTICA DO CAMINHO COM CUSTO MÍNIMO

O algoritmo de Dijkstra é usado para determinar o caminho mais curto com base no menor peso de um nó para outro usando o diagrama cartesiano. Para usar o algoritmo de Dijkstra no caso de o Problema do Caixeiro Viajante pode usar um gráfico completo onde cada nó está conectado a todos os outros nós. Neste algoritmo, tem uma desvantagem, não há visita de todos os nós do grafo devido à característica do algoritmo é um algoritmo de caminho mais curto de fonte única. O caixeiro viajante Problema é o problema mais importante, de modo que faz modificações no algoritmo, esta modificação está mudando a forma do sub-roteamento de Dijkstra no algoritmo e adiciona análise de cluster para a parte prioritária dos dados. O algoritmo Dijkstra realizará cálculos em todas as variáveis onde o menor peso de cada nó para que cada local seja calculado a distância percorrida por cada faixa.

15 CARACTERIZAÇÃO DA LINGUAGEM JAVA

Java é uma linguagem de programação criada por James Gosling da Sun Microsystems (Sun) em 1991. O objetivo do Java é escrever um programa uma vez e depois executá-lo em vários sistemas operacionais. A primeira versão publicamente disponível do Java (Java 1.0) foi lançada em 1995. A Sun Microsystems foi adquirida pela Oracle Corporation em 2010. A Oracle agora tem a liderança do Java. Em 2006, a Sun começou a disponibilizar o Java sob a GNU General Public License (GPL). A Oracle continua este projeto chamado OpenJDK.

Java contém alguns tipos primitivos, são elas: byte, é um inteiro de complemento de dois de 8 bits com sinal; short, é um inteiro de complemento de dois de 16 bits com sinal; int é um inteiro de complemento de dois com sinal de 32 bits; long é um inteiro de complemento de dois de 64 bits; float é um ponto flutuante IEEE 754 de 32 bits de precisão simples; double, é um ponto flutuante IEEE 754 de precisão dupla de 64 bits; boolean, tem apenas dois valores possíveis true e false; char é um único caractere Unicode de 16 bits.

Java contém algumas estruturas de seleção, são eles: seleção única, se trata da instrução IF; seleção dupla, se trata da instrução IF ELSE; e pôr fim a seleção múltipla, se trata da instrução SWITCH CASE. O Java contém também as estruturas de repetições, são elas: WHILE, DO WHILE, FOR. No Java contém também o Array, onde este é um objeto que contém elementos de um tipo de dados semelhante. Além disso, os elementos de uma array são armazenados em um local de memória contíguo. É uma estrutura de dados onde armazenamos elementos semelhantes. Podemos armazenar apenas um conjunto fixo de elementos em um array Java.

16 JAVA UTILIZANDO STS SPRING BOOT

Spring Tool Suite (STS) é um IDE java adaptado para o desenvolvimento de aplicativos corporativos baseados em Spring. É mais fácil, rápido e conveniente. E o mais importante, é baseado no Eclipse IDE. Spring Boot é uma estrutura e conjunto de ferramentas para desenvolver e implantar aplicativos baseados em Spring rapidamente e com muito pouca configuração. Ele ainda vem com um servidor Web Tomcat incorporado para executar seus aplicativos como um aplicativo independente.

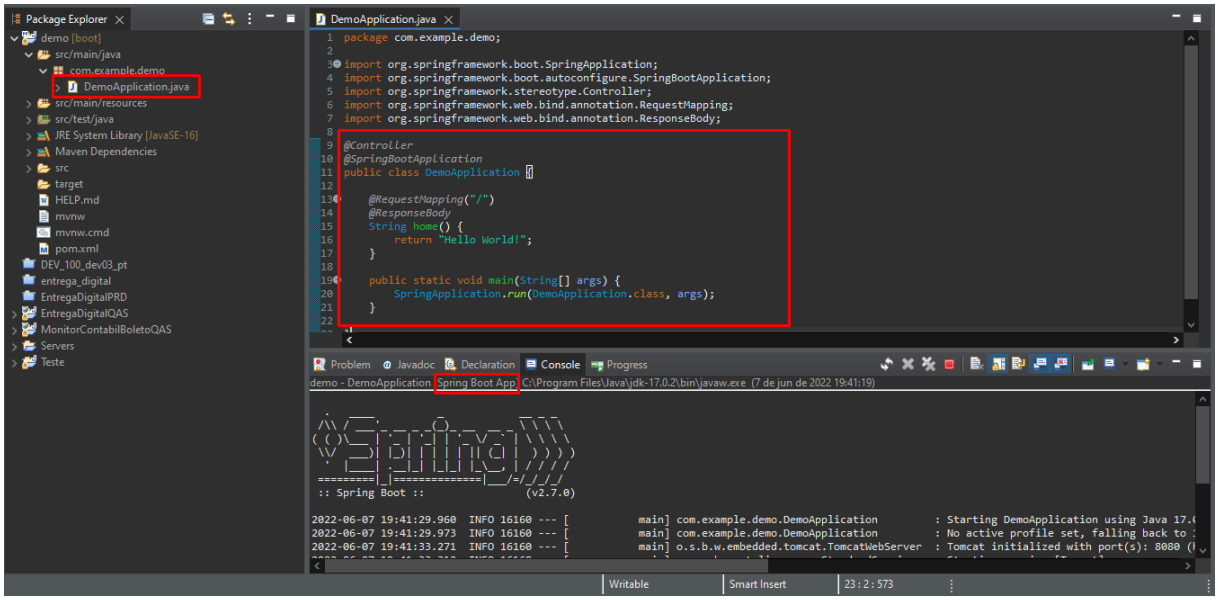
```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@SpringBootApplication
public class DemoApplication {

	@RequestMapping("/")
	@ResponseBody
	String home() {
		return "Hello World!";
	}

	public static void main(String[] args) {
		SpringApplication.run(DemoApplication.class, args);
	}

}
```


Package Explorer
demo [boot]
src/main/java
com.example.demo
DemoApplication.java
src/main/resources
src/test/java
JRE System Library [JavaSE-16]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml
DEV_100_dev03_pt
entrega_digital
EntregaDigitalPRD
EntregaDigitalQAS
MonitorContabilBoletoQAS
Servers
Teste
DemoApplication.java
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
@Controller
@SpringBootApplication
public class DemoApplication {
@RequestMapping("/")
@ResponseBody
String home() {
return "Hello World!";
}
public static void main(String[] args) {
SpringApplication.run(DemoApplication.class, args);
}
Problem
Javadoc
Declaration
Console
Progress
demo - DemoApplication
Spring Boot App
C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (7 de jun de 2022 19:41:19)
:: Spring Boot :: (v2.7.0)
2022-06-07 19:41:29.960 INFO 16160 --- [main] com.example.demo.DemoApplication : Starting DemoApplication using Java 17.
2022-06-07 19:41:29.973 INFO 16160 --- [main] com.example.demo.DemoApplication : No active profile set, falling back to
2022-06-07 19:41:33.271 INFO 16160 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (
Writable
Smart Insert
23 : 2 : 573

17 RESOLUÇÃO DE EXERCÍCIOS

17.1 Questão 1 - UFBA

Escreva um programa que receba uma lista de 10 inteiros via teclado e imprima todo a lista em na mesma linha.

```
import java.util.InputMismatchException;
import java.util.Iterator;
import java.util.Scanner;

public class Main {
	public static void main(String args[]) {
		int notExec  = 0;
		int list[]   = new int[10];
		String texto = "";
		for (int i = 0; i < 10; i++) {

			while ( notExec == 0 ) {
				try {
					Scanner sc = new Scanner(System.in);
					System.out.println("Digite o " + (i+1) + "º
número inteiro:");
					list[i] = sc.nextInt();
					notExec = 1;
			} catch (InputMismatchException e) {
			   System.out.print("O valor informado não é um número!\n");
			}
			    }
			    notExec = 0;
		}

		for (int i = 0; i < list.length; i++) {
			texto = texto +" ( " + list[i] + " ) ";
```

```
                }
                System.out.println("\nOs números digitados foram:" + texto);
        }
}
```

17.2 Questão 2 - UFBA

Escreva um programa que receba uma lista de 10 inteiros via teclado, em seguida o programa deve solicitar um número e informar se o número também está na lista ou não.

```
import java.util.InputMismatchException;
import java.util.Iterator;
import java.util.Scanner;

public class Main {
        public static void main(String args[]) {
                int notExec  = 0,
                   numListVerify = 0;
                int list[]   = new int[10];
                String texto = "";

                for (int i = 0; i < 10; i++) {
                        while ( notExec == 0 ) {
                                try {
                                        Scanner sc = new Scanner(System.in);
                                        System.out.println("Digite o " + (i+1) + "º
número inteiro:");
                                        list[i] = sc.nextInt();
                                        notExec = 1;
                        } catch (InputMismatchException e) {
                           System.out.print("O valor informado não é um número!\n");
                        }
                            }
                            notExec = 0;
```

```
                }
                while ( notExec == 0 ) {
                        try {
                                Scanner sc = new Scanner(System.in);
                                System.out.println("Digite um número:");
                                numListVerify = sc.nextInt();

                                notExec = 1;
            } catch (InputMismatchException e) {
               System.out.print("O valor informado não é um número!\n");
            }
                }
                for (int i = 0; i < list.length; i++) {
                        if (list[i] == numListVerify) {
                                System.out.println("O número " + list[i] + " está na
lista!\n");
                                System.out.println(list.length);
                        }else if( list.length == (i+1) ) {
                                System.out.println("O número " + numListVerify + " não
está na lista!\n");
                        }
                }
        }
}
```

17.3 Questão 7 - UFBA

Escreva um programa que leia e mostre uma lista de 10 elementos inteiros. Em seguida, conte quantos valores pares existem na lista, por fim, exiba a quantidade na tela.

```
import java.util.InputMismatchException;
import java.util.Iterator;
import java.util.Scanner;
```

```
public class Main {
	public static void main(String args[]) {
		int notExec  = 0,
		   qntNumPares = 0;
		int list[]   = new int[10];
		String texto = "";
		for (int i = 0; i < 10; i++) {
			while ( notExec == 0 ) {
				try {
					Scanner sc = new Scanner(System.in);
					System.out.println("Digite o " + (i+1) + "º
número inteiro:");
					list[i] = sc.nextInt();
					texto = texto +" ( " + list[i] + " ) ";
					if ( list[i] % 2 == 0) {
						qntNumPares = qntNumPares + 1;
					}
					notExec = 1;
			} catch (InputMismatchException e) {
			   System.out.print("O valor informado não é um número!\n");
			}
			    }
			    notExec = 0;
		}
		System.out.println("Os números digitados foram: " + texto + "\n");
		System.out.println("A quantidade de números pares da lista é de " +
qntNumPares + " números!");
	}
}
```