PROYECTO FINAL
DESARROLLO DE SOFTWARE

QUITO - ECUADOR

INSTITUTO TECNOLÓGICO SUPERIOR QUITO METROPOLITANO
FORMANDO PROFESIONALES DE ÉLITE
ITSQMET
ITSQMET 2021

EVIDENCIAR EL PROCESO DE APRENDIZAJE POR PARTE DE LOS ESTUDIANTES

WWW.ITSQMET.EDU.EC

INSTITUTO TECNOLÓGICO SUPERIOR

QUITO METROPOLITANO

PROGRAMACIÓN ORIENTADA A OBJETOS II

ING. CRISTIAN PAUL NARANJO B.

Alexander Ramírez Vizuete

ABRIL 2022 - SEPTIEMBRE 2022

CONTENIDO

I NTRODUCCIÓN

1. CAPÍTULO I

1.1. DELIMITACIÓN DEL TEMA DE INVESTIGACIÓN

1.2. TEMA:

Se debe realizar un programa que pueda ejecutar o procesar métodos HTTP, siendo mas específicos el método get,post,delete y put usando un modelo, vista, controlador.

1.3. HIPOTESIS I

Se debe usar programación orientada a objetos la cual conectaremos a un repositorio en la nube para tener un respaldo y volver a algo que se cambio y resulto un error o no debía ser eliminado además de usar programas externos para su verificación.

1.3.1. Objetivo General

Realizar la ejecución de métodos HTTP con la ayuda del framework llamado spring, exportando cada método para hacerlo mucho más fácil y bien estructurado

1.4. MARCO TEÓRICO

Los distintos métodos mencionados anteriormente se usan para realizar una acción especifica, trabajando conjuntamente con el navegador, tratándose del método get el cual muestra un mensaje o retorna un dato, o el post el cual es encargado igual de enviar datos pero tiene mayores funcionalidades que el método get ya que permite enviar formularios, el método delete como la palabra lo dice elimina algo en específico.

Métodos de petición HTTP

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado.

GET, solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

HEAD, pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.

POST, se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

PUT, reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

DELETE, borra un recurso en específico.

CONNECT, establece un túnel hacia el servidor identificado por el recurso.

OPTIONS, es utilizado para describir las opciones de comunicación para el recurso de destino.

TRACE, realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.

PATCH, es utilizado para aplicar modificaciones parciales a un recurso.

Rest Api

Una API de REST, o API de REST ful, es una interfaz de programación de aplicaciones (API o API web) que se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web

2. CAPÍTULO II

2.1. METODOLOGÍA

Ya teniendo creado el proyecto procedemos a crear sus clases, procedemos con la clase Icliente servicio, para poder crear los métodos que se usaran en la otra clase con cada método, después creamos cliente controlador importando los framework de spring, además de definir su estructura para poder identificarlo en una pagina web, lo llamamos /saludar, seguido instanciamos la clase para comunicarse con otra la cual es Icliente, con ello definimos funciones de tipo cadena las cuales solo retornaran el método de la otra clase. Ya culminado procedemos con la clase cliente servicio que implementa a la primera clase para poder obtener los métodos y definir que es lo que se va a retornar cada uno.

2.1.1. Programas Usados

POSTMAN

Postman en sus inicios nace como una extensión que podía ser utilizada en el navegador Chrome de Google y básicamente nos permite realizar peticiones de una manera simple para testear APIs de tipo REST propias o de terceros.

Postman ha evolucionado y ha pasado de ser de una extensión a una aplicación que dispone de herramientas nativas para diversos sistemas operativos como lo son Windows, Mac y Linux.

Postman sirve para múltiples tareas:

Testear colecciones o catálogos de APIs tanto para Frontend como para Backend.

Organizar en carpetas, funcionalidades y módulos los servicios web.

Permite gestionar el ciclo de vida (conceptualización y definición, desarrollo, monitoreo y mantenimiento) de nuestra API.

Generar documentación de nuestras APIs.

SPRING

El uso de inyección de dependencias facilita la programación contra interfaz, permitiendo a los distintos componentes depender únicamente de interfaces y produciendo así un código menos acoplado. No solo eso, también permite implementar el patrón singleton de una forma extremadamente sencilla.

Spring aumenta la productividad y reduce la fricción al ofrecernos abstracciones sobre implementaciones de tecnologías concretas. Un ejemplo claro es el de spring-data, que nos permite definir el acceso a base de datos con interfaces Java. Esto lo consigue parseando el nombre de los métodos y generando la consulta con la sintaxis específica para el driver que utilicemos.

Spring nos permite desactivar estos "comportamientos mágicos" en caso de ser necesario, por lo que podemos tomar el control cuando necesitemos más granularidad. Siguiendo con el

ejemplo de spring-data, este control sería necesario si tenemos que realizar consultas mucho más complejas que un SELECT * BY name. En esos casos, entre otras opciones, podemos anotar nuestro método con @Query y escribir la consulta que deseemos. Ya no hay magia

3. CAPÍTULO III

3.1. DESARROLLO

Client Service

```
package ec.edu.itsqmet.service;


public interface IClientService {
    public String saludar();
    public String saludP();
    public String saludPT();
    public String saludD();
    public String saludPH();
}
package ec.edu.itsqmet.service.impl;


import org.springframework.stereotype.Service;
```

```
import ec.edu.itsqmet.service.IClientService;


@Service
public class ClientService implements IClientService{


    public String saludar() {
            return "Hello World";
    }
    public String saludP() {
            return "Soy un método POST";
    }
    public String saludPT() {
            return "Soy un método PUT";
    }
    public String saludD() {
            return "Soy un método DELETE";
    }
    public String saludPH() {
            return "Soy un método PATCH";
    }
    public String nombre() {
```

```
            return "Alex";
    }

}
package ec.edu.itsqmet.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import ec.edu.itsqmet.service.IClientService;

@RestController
@RequestMapping("/client")
public class ClientController {
```

```
// private ClientService clientService = new ClientService();
    @Autowired
    private IClientService clientService;
    @GetMapping("/saludar")
    public String saludar() {
    return clientService.saludar();
    }
    @PostMapping("/saludP")
    public String saludP() {
            return clientService.saludP();
    }
    @PutMapping("/saludPT")
    public String saludPT() {
            return clientService.saludPT();
    }
    @DeleteMapping("/saludD")
    public String saludD() {
            return clientService.saludD();
    }
    @PatchMapping("/saludPH")
    public String saludPH() {
```

```
            return clientService.saludPH();
        }
    }
```

4. CAPÍTULO IV

4.1. CONCLUSIONES

En conclusión, resulto muy interesante el funcionamiento de los diferentes métodos HTTP, que se puede destacar que el framework nos reduce dolores de cabeza ya que evitamos programar miles de líneas de código para solo dedicarnos a sector importantes.

4.2. RECOMENDACIONES

Debes tener en cuenta que si ocurre algun problema o se produce un error se puede volver a un estado en el que el código funcionaba gracias al repositorio en la nube.

BIBLIOGRAFÍA

(09 de 11 de 2017). *Técnicas de Auditoría Asistidas por Computador - TAAC*. Obtenido de Comunidad Contable: http://www.comunidadcontable.com/BancoConocimiento/Otros/tecnicas-de-auditoria-asistidas-por-computador-

Rodrigo González González y Jorge Jimeno Bernal. (2012). *Check list / Listas de chequeo: ¿Qué es un checklist y cómo usarlo?* Obtenido de Pdcahome: https://www.pdcahome.com/check-list/

Auditool. taac.asp?#:~:text=Las%20técnicas%20de%20auditoría%20asistidas,para%20aplicar%20ciertas%20rutinas%20pre-

Betancourt, D. F. (21 de 05 de 2017). *¿Qué es una salida no conforme?* Obtenido de ingenioempresa: https://ingenioempresa.com/salidas-no-conformes/#:~:text=salida%20no%20conforme-,¿Qué%20es%20una%20salida%20no%20conforme%3F,para%20la%20ejecución%20de%20actividades.

AVAST. (s.f.). *Qué es el phishing.* Obtenido de Avast: https://www.avast.com/es-es/c-phishing

Becerra, V. (2 de 04 de 2013). *Recursos Humanos.* Obtenido de EmprendePyme: https://www.emprendepyme.net/recursos-humanos

Benget. (s.f.). *Benchmarking.*