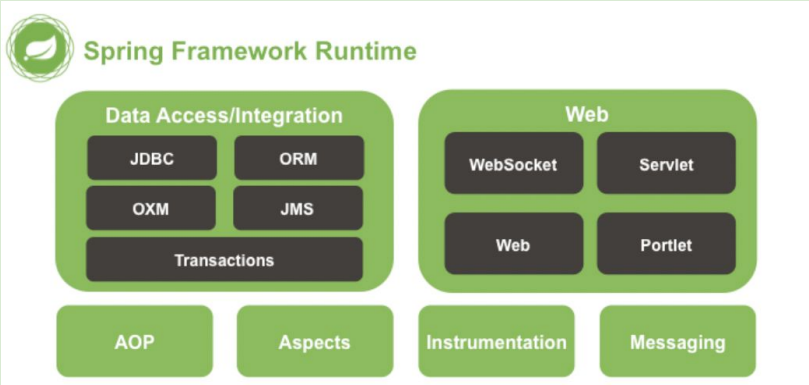

spring

1. ¿Qué es Spring Framework?

- ***Spring*** es un ***framework*** de código abierto para la creación de aplicaciones empresariales con soporte
- Spring framework es un ***conjunto de bibliotecas*** y ***herramientas*** que facilitan y agilizan el desarrollo de aplicaciones Java
- Proporciona una infraestructura ***sólida*** y ***modular***



1. ¿Qué es Spring Framework?

- Los primeros componentes de lo que se ha convertido en Spring Framework fueron escritos por ***Rod Johnson*** en el año 2000, mientras trabajaba como consultor independiente para sus clientes en la industria financiera en Londres.
- El MVC de Spring presenta una ***arquitectura Tipo 2***
- ***Spring Framework*** puede hacer diferentes tipos de aplicaciones:

    - **Aplicaciones que acceden a base de datos vía SQL**
    - **Aplicaciones de escritorio**

2. Ventajas Vs Inconvenientes

Podemos encontrar diferentes ***ventajas*** , entre las que destacan:

- ***Modularidad***: Permite a los desarrolladores usar solo los módulos necesarios
- ***Inyección de dependencias***:Ayuda a reducir la dependencia de clases
- ***Simplificación del acceso a datos***: Gracias a los ORM (Object Relational Mapping), Spring simplifica el acceso a datos.
- ***Seguridad:*** Spring tiene un enfoque especial en la seguridad, lo que es crucial para las aplicaciones empresariales.
- ***Integración con otros frameworks***:Spring proporciona buen soporte para los principales frameworks como Hibernate, Struts,JSF … ……..

2. Ventajas Vs Inconvenientes

Podemos encontrar diferentes ***inconvenientes*** , entre los que destacan:

- ***Complejidad:*** Spring es un framework complejo que requiere tener claros muchos conceptos y tiene miles de clases que aportan muchas funcionalidades.

- ***Falta soporte nativo para GraalVM:*** Si la aplicación usa la API Java Reflection, que permite descubrir código en tiempo de ejecución, Spring Framework aún no ofrece soporte nativo para las imágenes con GraalVM1

- ***Abstracción:*** Esto puede llevar a una falta de comprensión de lo que realmente está sucediendo debajo del capó, lo que puede complicar la depuración y el mantenimiento.

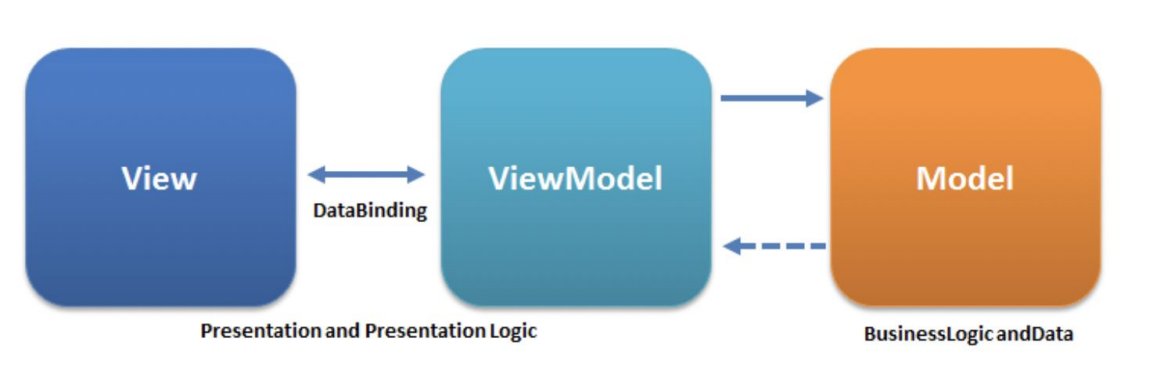
3. Desarrollo De La Aplicación

En el desarrollo de la aplicación hemos usado el ***Modelo- Vista-Controlador***

- ***Modelo:***Es la representación de la información con la cual el sistema opera.Encargado de instanciar las entidades de la base de datos .

- ***Vista:***Presenta el 'modelo'.Encargada de definir visualización que el usuario observa en la interfaz

- ***Controlador:***Responde a eventos e invoca peticiones al 'modelo'



3. Desarrollo De La Aplicación

Modelo(Roles y tMuestra)

```java
package GI.proyecto.model;

public enum Roles {
    Invitado,
    Usuario,
    Administrador
}
```

```java
package GI.proyecto.model;

import java.util.Objects;

@Entity
public class tMuestra {
    @Id
    @GeneratedValue
    private Integer IDMuestra;
    @Nullable
    private String NIF_Paciente;
    @Nullable
    private String Cultivo;
    @ManyToOne
    private tSolucion Solucion;

    public tMuestra() {}

    public Integer getID() {
        return IDMuestra;
    }
    public void setID(Integer ID) {
        IDMuestra = ID;
    }
    public String getNIF_Paciente() {
        return NIF_Paciente;
    }
    public void setNIF_Paciente(String Nif_Paciente) {
        NIF_Paciente = Nif_Paciente;
    }
    public String getCultivo() {
        return Cultivo;
    }
    public void setCultivo(String cultivo) {
        Cultivo = cultivo;
    }
    public tSolucion getSolucion() { return Solucion; }
    public void setSolucion(tSolucion solucion) { Solucion = solucion; }

    @Override
    public int hashCode() {
        return Objects.hash(IDMuestra);
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        tMuestra other = (tMuestra) obj;
        return IDMuestra == other.IDMuestra;
    }

}
```

3. Desarrollo De La Aplicación

Modelo(tPermiso)

```java
package Gi.proyecto.model;

import jakarta.persistence.Entity;

@Entity
public class tPermiso {
    @Id
    String pantalla;
    @ManyToOne
    tRol rolName;
    Boolean acceso;
    Boolean insertar;
    Boolean modificar;
    Boolean borrar;

    public tPermiso(){

    }

    public tPermiso(tRol rolName) {
        this.rolName = rolName;
        String rol = rolName.getRolName();
        this.acceso = true;
        this.insertar = false;
        this.modificar = false;
        this.borrar = false;
        if(rol != "Invitado") {
            this.modificar = true;
            if(rol == "Administrador") {
                this.insertar = true;
                this.borrar = true;
            }
        }
    }

    public String getPantalla() {
        return pantalla;
    }

    public void setPantalla(String pantalla) {
        this.pantalla = pantalla;
    }

    public tRol getRolName() {
        return rolName;
    }

    public void setRolName(tRol rolName) {
        this.rolName = rolName;
    }

    public Boolean getAcceso() {
        return acceso;
    }
```

```java
public Boolean getInsertar() {
    return insertar;
}

public void setInsertar(Boolean insertar) {
    this.insertar = insertar;
}

public Boolean getModificar() {
    return modificar;
}

public void setModificar(Boolean modificar) {
    this.modificar = modificar;
}

public Boolean getBorrar() {
    return borrar;
}

public void setBorrar(Boolean borrar) {
    this.borrar = borrar;
}
```

3. Desarrollo De La Aplicación

Modelo(tRol y tSolución)

```java
package Gi.proyecto.model;

import java.util.List;

@Entity
public class tRol {
    @Id
    private String rolName;
    @Nullable
    private String rolDes;
    private boolean admin;
    @OneToMany (mappedBy = "nif")
    private List<tUsuario> usuario;
    @OneToMany (mappedBy ="pantalla")
    private List<tPermiso> permiso;


    public List<tUsuario> getUsuario() {
        return usuario;
    }

    public void setUsuario(List<tUsuario> usuario) {
        this.usuario = usuario;
    }

    public tRol() {
    }

    public String getRolName() {
        return rolName;
    }
    public void setRolName(String rolName) {
        this.rolName = rolName;
    }
    public String getRolDes() {
        return rolDes;
    }
    public void setRolDes(String rolDes) {
        this.rolDes = rolDes;
    }
    public boolean isAdmin() {
        return admin;
    }
    public void setAdmin(boolean admin) {
        this.admin = admin;
    }

}
```

```java
package Gi.proyecto.model;
import jakarta.annotation.Nullable;

@Entity
public class tSolucion {
    @Id
    @GeneratedValue
    private Integer IDSolucion;
    @Nullable
    private String Solucion;
    @Nullable
    private String Uso;
    @OneToMany (mappedBy = "Solucion")
    private List<tMuestra> Muestras;

    public tSolucion() {
    }
    public int getId() {
        return IDSolucion;
    }
    public void setId(int id) {
        this.IDSolucion = id;
    }
    public String getSolucion() {
        return Solucion;
    }
    public void setSolucion(String solucion) {
        Solucion = solucion;
    }
    public String getUso() {
        return Uso;
    }
    public void setUso(String uso) {
        Uso = uso;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof tSolucion tSolucion)) return false;
        return IDSolucion == tSolucion.IDSolucion && Objects.equals(Solucion, tSolucion.Solucion) && Objects.equals(Uso, tSolucion.Uso);
    }

    @Override
    public int hashCode() {
        return IDSolucion*Solucion.hashCode()*Uso.hashCode();
    }
}
```

3. Desarrollo De La Aplicación

Modelo(tUsuario)

```java
package GI.proyecto.model;

import jakarta.persistence.Entity;

@Entity
public class tUsuario {

    @Id
    String nif;
    String password;
    @ManyToOne
    tRol rolName;
    public tUsuario() {

    }

    public String getNif() {
        return nif;
    }

    public void setNif(String nif) {
        this.nif = nif;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public tRol getRol() {
        return rolName;
    }

    public String getRolName() {
        return rolName.getRolName();
    }
    public void setRolName(tRol rolName) {
        this.rolName = rolName;
    }
}
```

3. Desarrollo De La Aplicación

Vista

```java
package GI.proyecto.repository;

import GI.proyecto.model.tMuestra;

public interface MuestraRepository extends JpaRepository<tMuestra, Integer> {

    @Transactional
    @Modifying
    @Query("UPDATE tMuestra m SET m.Cultivo = ?3, m.NIF_Paciente = ?2, m.Solucion = ?4 WHERE m.ID = ?1")
    void updateMuestra(Integer id, String nif, String cultivo, tSolucion solucion);

}
```

```java
package GI.proyecto.repository;

import GI.proyecto.model.tSolucion;

public interface SolucionRepository extends JpaRepository<tSolucion, Integer> {

}
```

```java
package GI.proyecto.repository;

import org.springframework.data.jpa.repository.JpaRepository;

public interface RolRepository extends JpaRepository<tRol, String> {


}
```

```java
package GI.proyecto.repository;

import java.util.List;

public interface UsuarioRepository extends JpaRepository<tUsuario, String> {

    @Query(value = "select * from t_Usuario where NIF = :nif", nativeQuery = true)
    List<tUsuario> findByNif(String nif);
}
```

3. Desarrollo De La Aplicación

Controlador

```java
package G1.proyecto.controller;

import java.util.List;

@Controller
public class webController {
    @Autowired
    tMuestraService muestraService;

    @Autowired
    tSolucionService solucionService;

    @Autowired
    tUsuarioService usuarioService;

    @GetMapping("/muestra")
    public String listarMuestras(Model model, HttpSession sesion) {
        tMuestra muestra = new tMuestra();
        tUsuario usuario = usuarioService.findById((String) sesion.getAttribute("nif")).get(0); //sacamos de la sesion el nif pasada en el login
        model.addAttribute("usuario", usuario);
        model.addAttribute("muestra", muestra);

        List<tMuestra> muestrasList = muestraService.getAll();
        model.addAttribute("muestrasList", muestrasList);

        List<tSolucion> soluciones = solucionService.getAll();
        model.addAttribute("soluciones", soluciones);
        return "viewMuestras";
    }

    @PostMapping("/guardar-muestra")
    public String guardarMuestra(tMuestra nuevaMuestra) {
        muestraService.guardarMuestra(nuevaMuestra);
        return "redirect:/muestra";
    }

```

3. Desarrollo De La Aplicación

Controlador

```java
@RequestMapping("/mostrarMuestra")
public String doShowMostrar(Model model, @RequestParam("muestraId") Integer muestraId, HttpSession sesion) {
    tUsuario usuario = usuarioService.findById((String) sesion.getAttribute("nif")).get(0);
    model.addAttribute("usuario", usuario);

    tMuestra muestra = muestraService.getMuestra(muestraId);
    model.addAttribute("muestra", muestra);

    List<tMuestra> muestrasList = muestraService.getAll();
    model.addAttribute("muestrasList", muestrasList);

    tMuestra nuevaMuestra = new tMuestra();
    model.addAttribute("nuevaMuestra", nuevaMuestra);

    List<tSolucion> soluciones = solucionService.getAll();
    model.addAttribute("soluciones", soluciones);
    return "viewMuestras";
}

@RequestMapping("/login")
public String login(Model model) {
    model.addAttribute("usuario", new tUsuario());
    return "loginView";
}

@PostMapping("/post-login")
public String postLogin(tUsuario usuario, HttpSession sesion) {

    List<tUsuario> user = usuarioService.findById(usuario.getNif());
    if (user.size() == 1 && user.get(0).getPassword().equals(usuario.getPassword())) { // podemos usar el bCrypt

        sesion.setAttribute("nif", user.get(0).getNif());// Al hacer login metemos por la sesión el
                                                         // nif para poder hacer uso del usuario en otras vistas
        return "redirect:/muestra";
    } else {
        return "error";
    }
}

@GetMapping("/exit")
public String exit(){
    return "redirect:/";
}

@GetMapping("/goLogin")
public String goLogin(HttpSession sesion){
    sesion.invalidate();
    return "loginView";
}
```

3. Desarrollo De La Aplicación

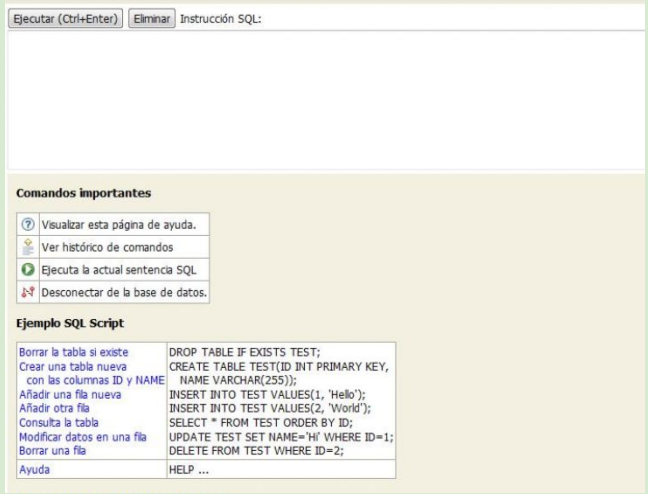
Controlador

```
@GetMapping("/goLogin")
public String goLogin(HttpSession sesion){
    sesion.invalidate();
    return "loginView";
}

@PostMapping("/limpiar")
public String limpiar(){
    return "redirect:/muestra";
}

@RequestMapping("/borrar-muestra/{id}")
public String cargarPaginaBorrar(@PathVariable Integer id) {
    muestraService.delete(id);
    return "redirect:/muestra";

}



@PostMapping("/editar-muestra/{id}")
public String actualizar(@PathVariable Integer id,tMuestra muestraActualizada) {

    muestraService.updateMuestra(id, muestraActualizada.getNIF_Paciente(), muestraActualizada.getCultivo(), muestraActualizada.getSolucion());

    return "redirect:/muestra";
}
```

3. Desarrollo De La Aplicación

Hemos usado la ***dependencia thymeleaf*** , encargada de cargar los datos en la vista.

Para realizar la base de datos , hemos usado ***h2***



4.Demo

Trabajo realizado por:

1. Astudillo Fraga, Pablo.
2. Rosales Santiago, Lucia.
3. Labella Ramírez, Miguel.
4. Alarcon Carrion, Pablo.
5. Jerez Sánchez, Manuel Jesús.
6. Senciales de la Higuera, Pablo.