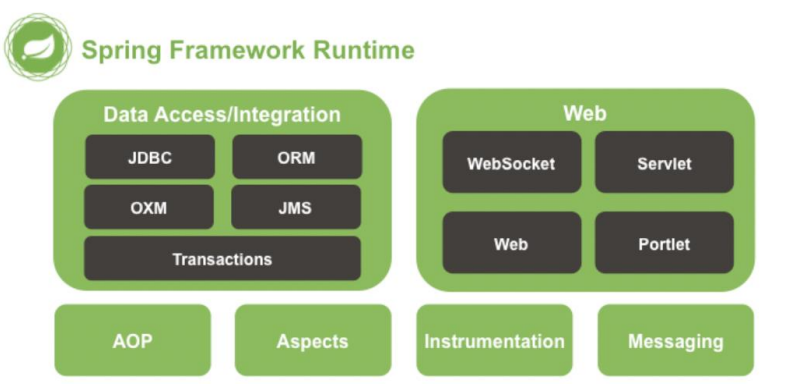

spring

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 2003, 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.

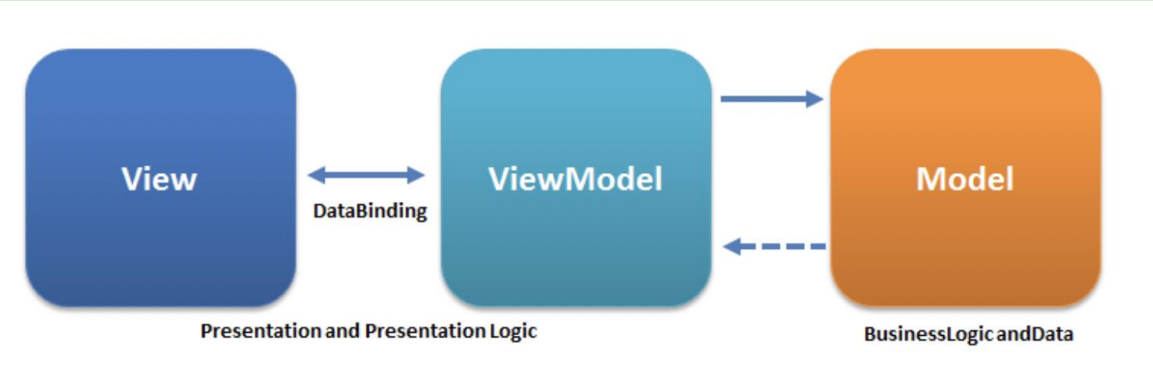
3. Desarrollo De La Aplicación

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 .

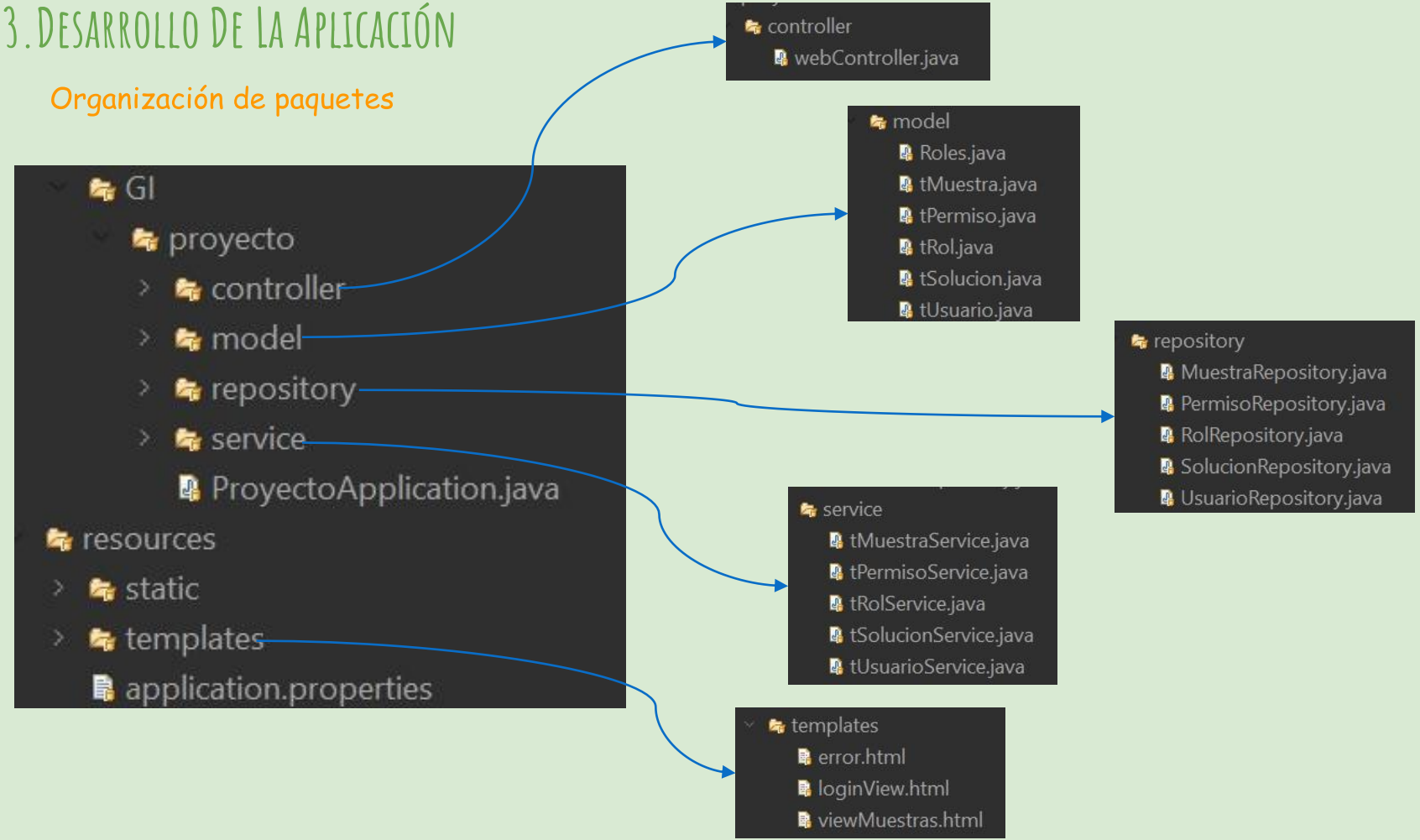
- ***Vista:***Presenta el 'modelo'.Encargada de definir visualización que el usuario observa en la interfaz

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



3. Desarrollo De La Aplicación

Organización de paquetes



3. Desarrollo De La Aplicación

Modelo claves

```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() {}
```

- Notación @Entitty
- Notación @Id
- Notación @GeneratedValue
- Notación @Nullable
- Constructor vacío, getters y setters.

3. Desarrollo De La Aplicación

Modelo claves

Notaciones que representan las relaciones

```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() {}
```

```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() {
```

3. Desarrollo De La Aplicación

Repository claves

```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 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);
}
```

- JpaRepository y sus métodos ya implementados
- Notación @Query
- Notación @Modifying
- Notación @Transactional

3. Desarrollo De La Aplicación

Service claves

```java
@Service
public class tMuestraService{
        @Autowired
        MuestraRepository muestraRepository;

        public List<tMuestra> getAll(){
            return muestraRepository.findAll();
        }

        public void guardarMuestra(tMuestra muestra) {
            muestraRepository.saveAndFlush(muestra);
        }

        public tMuestra getMuestra(Integer muestraId){
            return muestraRepository.findById(muestraId).orElse(null);
        }

        public void delete(Integer id) {
            muestraRepository.deleteById(id);

        }

        public void updateMuestra(Integer id, String nif, String cultivo, tSolucion solucion) {
            muestraRepository.updateMuestra(id, nif, cultivo, solucion);
        }

}
```

- Notación @Service
- Notación @AutoWired
- Métodos

3. Desarrollo De La Aplicación

Controlador claves

```java
@Controller
public class webController {
    @Autowired
    tMuestraService muestraService;

    @Autowired
    tSolucionService solucionService;

    @Autowired
    tUsuarioService usuarioService;

    @Autowired
    tPermisoService permisoService;

    @GetMapping("/muestra")
    public String listarMuestras(Model model, HttpSession sesion) {
        tMuestra muestra = new tMuestra();
        tUsuario usuario = usuarioService.findById((String) sesion.getAttribute("nif")).get(0);


        tPermiso permiso = permisoService.findPermiso(usuario.getRolName());
        model.addAttribute("usuario", usuario);
        model.addAttribute("permiso", permiso);
        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";
    }
```

- Notación @Controller
- Notación @Autowired
- Notación @GetMapping
- Notación @PostMapping
- Model
- HttpSession

3. Desarrollo De La Aplicación

Controlador HttpSession

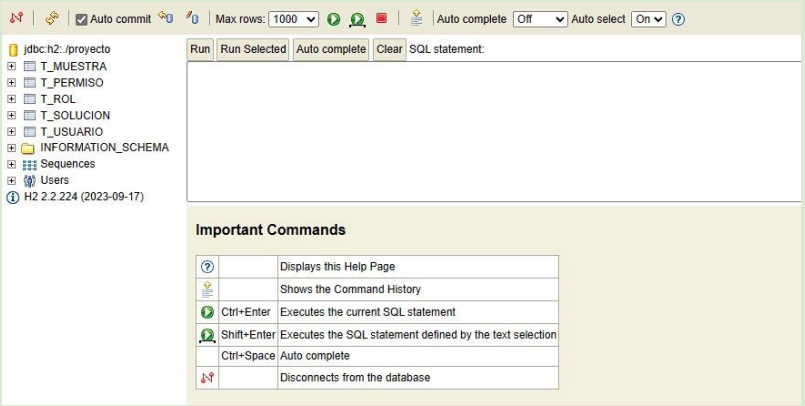
```java
@RequestMapping(value = {"/login","","/"})
public String login(Model model) {
    model.addAttribute("usuario", new tUsuario());
    return "loginView";
}

@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())) {
        tPermiso permiso = permisoService.findPermiso(user.get(0).getRolName());
        if (permiso.getAcceso()) {
            sesion.setAttribute("nif", user.get(0).getNif()); // Al hacer login metemo
                                                              // nif para poder hace
                                                              // vistas

            return "redirect:/muestra";
        }else{
            return "error";
        }

    } else {
        return "error";
    }
}
```

```java
@Controller
public class webController {
    @Autowired
    tMuestraService muestraService;

    @Autowired
    tSolucionService solucionService;

    @Autowired
    tUsuarioService usuarioService;

    @Autowired
    tPermisoService permisoService;

    @GetMapping("/muestra")
    public String listarMuestras(Model model, HttpSession sesion) {
        tMuestra muestra = new tMuestra();
        tUsuario usuario = usuarioService.findById((String) sesion.getAttribute("nif")).get(0);


        tPermiso permiso = permisoService.findPermiso(usuario.getRolName());
        model.addAttribute("usuario", usuario);
        model.addAttribute("permiso", permiso);
        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

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

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



3. Desarrollo De La Aplicación

Uso de thymeleaf

```java
@Controller
public class webController {
    @Autowired
    tMuestraService muestraService;

    @Autowired
    tSolucionService solucionService;

    @Autowired
    tUsuarioService usuarioService;

    @Autowired
    tPermisoService permisoService;

    @GetMapping("/muestra")
    public String listarMuestras(Model model, HttpSession sesion) {
        tMuestra muestra = new tMuestra();
        tUsuario usuario = usuarioService.findById((String) sesion.getAttribute("nif")).get(0);


        tPermiso permiso = permisoService.findPermiso(usuario.getRolName());
        model.addAttribute("usuario", usuario);
        model.addAttribute("permiso", permiso);
        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";
    }
```

```html
<table border="1">
    <thead>
        <tr>
            <th>ID</th>
            <th>NIF</th>
            <th>Cultivo</th>
            <th>Solucion</th>
        </tr>
    </thead>
    <tbody>
        <!-- Use Thymeleaf iteration to populate table rows -->
        <tr th:each="muestra : ${muestrasList}">
            <td th:text="${muestra.getID()}"></td>
            <td th:text="${muestra.getNIF_Paciente()}"></td>
            <td th:text="${muestra.getCultivo()}"></td>
            <td th:text="${muestra.Solucion.getSolucion()}"></td>
            <td>
                <form th:action="@{/mostrarMuestra}" method="post" id="formId">
                    <input type="hidden" name="muestraId" th:value="${muestra.getID()}" />
                    <button type="submit" onclick="submitForm(this)">Mostrar</button>
                </form>
            </td>

        </tr>
    </tbody>
</table>
```

3. Desarrollo De La Aplicación

Uso de thymeleaf

```java
@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";
}
```

```java
@GetMapping("/muestra")
public String listarMuestras(Model model, HttpSession sesion) {
    tMuestra muestra = new tMuestra();
    tUsuario usuario = usuarioService.findById((String) sesion.getAttribute("nif")).get(0);


    tPermiso permiso = permisoService.findPermiso(usuario.getRolName());
    model.addAttribute("usuario", usuario);
    model.addAttribute("permiso", permiso);
    model.addAttribute("muestra", muestra);

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

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

```html
<form th:action="@{/editar-muestra/{id}(id=${muestra.getID()})}" method="post" th:object="${muestra}"
    id="muestraForm">
    <!-- Campo para NIF del Paciente -->
    <label>NIF <input required type="text" id="NIF_Paciente" th:field="*{NIF_Paciente}"
            placeholder="NIF" /></label><br />

    <!-- Campo para el Cultivo -->
    <label>Cultivo <input required type="text" id="Cultivo" th:field="*{Cultivo}"
            placeholder="Cultivo" /></label><br />

    <!-- Campo para la Solución -->
    <label>Solucion:
        <select style="width:20ch; height:10ch;" name="Solucion" id="Solucion" th:field="*{Solucion}"
            th:size="${soluciones.size()}" required>
            <option th:each="solucion : ${soluciones}" th:value="${solucion.getId()}"
                th:text="${solucion.getSolucion()}" th:selected="*{Solucion}"></option>
        </select>
    </label><br />

    <button type="submit" th:if="${permiso.getModificar() and muestra.getID() != null} ">Actualizar</button>

    <button type="button" onclick="validarYEnviar('insertar')"
        th:if="${permiso.getInsertar()}">Insertar</button>
</form>
```

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.

Para consultar el código: https://github.com/pSenciales/GestionInformacion