Alumno: Alejandro Herrera Mendoza

Descripción: Sistema web para registrar la información de un doctor para ser almacenada en un archivo txt. con el fin de poder tener acceso al sistema con su usuario y contraseña antes registradas y poder visualizar su información en una nueva página.

Tecnologías: Jsp, Spring framework 3.2.13.RELEASE, Maven.

Lenguaje de programación: Java

Código

Archivo de configuración para asignar las características principales para el funcionamiento del proyecto en Spring.

Servlet DispatcherServlet encargado de resolver la vistas apropiadas a mostrar.

web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="3.0" xmlns="http://java.sun.com/xml

<welcome-file-list>
        <welcome-file>login.jsp</welcome-file>
    </welcome-file-list>

 <context-param>
     <param-name>contextConfigLocation</param-name>
     <param-value>/WEB-INF/court-service.xml</param-value>
 </context-param>
 <listener>
     <listener-class>
        org.springframework.web.context.ContextLoaderListener
     </listener-class>
 </listener>

<servlet>
 <servlet-name>court</servlet-name>
     <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
     </servlet-class>
 <load-on-startup>1</load-on-startup>
 </servlet>

 <servlet-mapping>
     <servlet-name>court</servlet-name>
     <url-pattern>/</url-pattern>
 </servlet-mapping>
</web-app>
```

Declaración de las clases de servicios que utilizará Spring.

court-service.xml

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">


    <bean id="loginService"
        class="com.apress.springrecipes.court.service.LoginServiceImpl" />
    <bean id="registerService"
        class="com.apress.springrecipes.court.service.RegisterServiceImpl" />


</beans>
```

Sufijos y prefijos para el reconocimiento de las vistas.

```xml
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
</bean>
```

Configuración para mostrar una excepción por default en Spring.

```xml
<bean
    class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
    <property name="exceptionMappings">
        <props>
            <prop
                key="com.apress.springrecipes.court.service.DoctorNotAvailableException">
                doctorNotAvailable
            </prop>
        </props>
    </property>
</bean>

<bean
    class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
    <property name="exceptionMappings">
        <props>
            <prop
                key="com.apress.springrecipes.court.service.CredencialesNotAvailableException">
                loginNotAvailable
            </prop>
        </props>
    </property>
</bean>
```

- Declaración e instancia de los servicios a utilizar.
- En el método GET es necesario hacer la instancia del objeto del formulario para que pueda ser reconocido en la vista.
- Método POST se obtiene el objeto lleno o vacío pero para verificarlo se utiliza el servicio validator para verificar que todos los campos vengan con su información respectiva.

```java
public class LoginController {

    private LoginService service;
    private CredencialesValidator validator;

    @Autowired
    public LoginController(LoginService service, CredencialesValidator validator) {
        this.service = service;
        this.validator = validator;
    }

    @RequestMapping(method = RequestMethod.GET)
    public String login(Model model) {
        Credenciales credenciales = new Credenciales();
        model.addAttribute("credenciales", credenciales);
        return "login";
    }

    @RequestMapping(method = RequestMethod.POST)
    public String validation(@ModelAttribute("credenciales") Credenciales credenciales, BindingResult result,
            SessionStatus status, Model model) {
        Date date = new Date();
        validator.validate(credenciales, result);
        if (result.hasErrors()) {
            model.addAttribute("credenciales", credenciales);
            return "login";
        } else {
            if (service.getValidation(credenciales)) {
                model.addAttribute("fecha", date);
                model.addAttribute("infoDoctor", service.getInfoDoctor());
                return "inicio";
            } else {
                model.addAttribute("credenciales", credenciales);
                model.addAttribute("mensaje", Constantes.mensaje);
                return "login";
            }
        }

    }
```

- Clase que implementa el contrato del servicio.
- En caso de no poder guardarse la información del doctor en el archivo lanzará una excepción por default controlada por Spring.

```java
public class RegisterServiceImpl implements RegisterService {

    @Override
    public boolean createDoctor(Doctor doctor) throws DoctorNotAvailableException {
        GuardarInformacion doc = new GuardarInformacion();
        if (!doc.save(doctor)) {
            throw new DoctorNotAvailableException(doctor.getNombre(), doctor.getApellidos(), doctor.getDireccion(),
                    doctor.getTelefono(), doctor.getRfc(), doctor.getEspecialidad(), doctor.getSexo(),
                    doctor.getContra(), doctor.getContra());

        } else {
            return true;
        }
    }

}
```

Clase con el fin de tener los atributos que van a ser verificados en el formulario por el validator.

DoctorNotAvailableException.java

```java
package com.apress.springrecipes.court.service;

public class DoctorNotAvailableException extends RuntimeException {

    private String nombre;
    private String apellidos;
    private String direccion;
    private String telefono;
    private String rfc;
    private String especialidad;
    private String sexo;
    private String correo;
    private String contra;

    public DoctorNotAvailableException() {
        super();
    }

    public DoctorNotAvailableException(String nombre, String apellidos, String direccion, String rfc,
            String especialidad, String sexo, String telefono, String correo, String contra) {
        super();
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.direccion = direccion;
        this.telefono = telefono;
        this.rfc = rfc;
        this.especialidad = especialidad;
        this.sexo = sexo;
        this.correo = correo;
        this.contra = contra;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
```

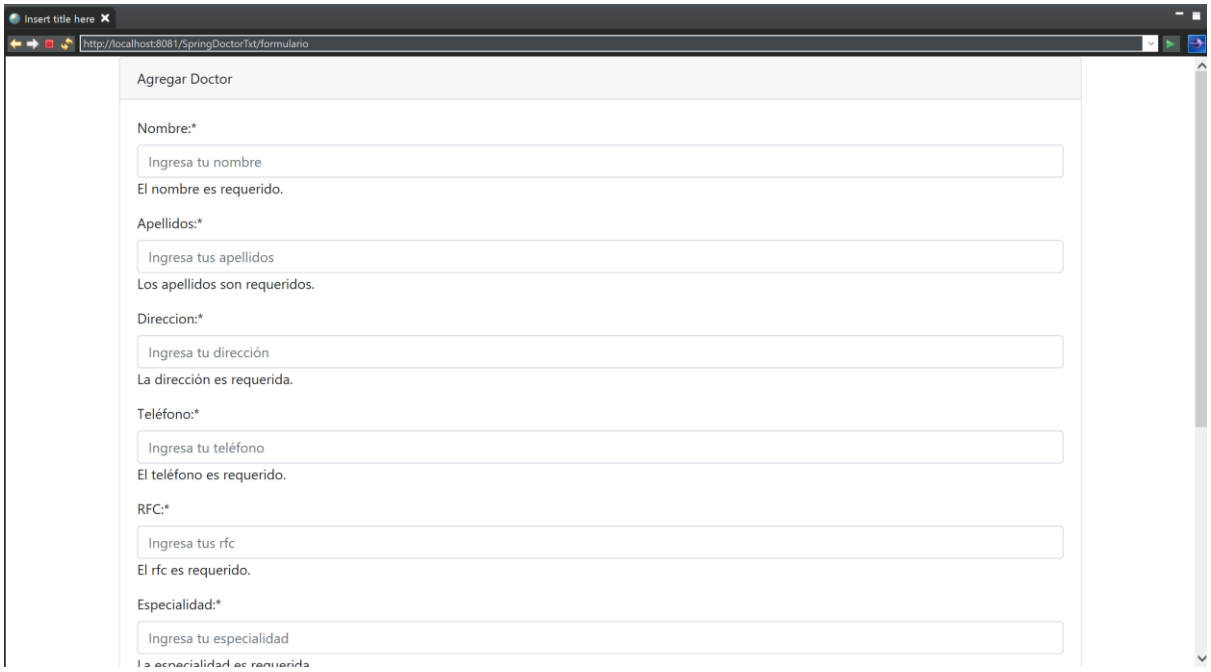
Clase Validator para verificar los campos requeridos y devolver un mensaje de erro para cada uno de ellos.

DoctorValidator.java

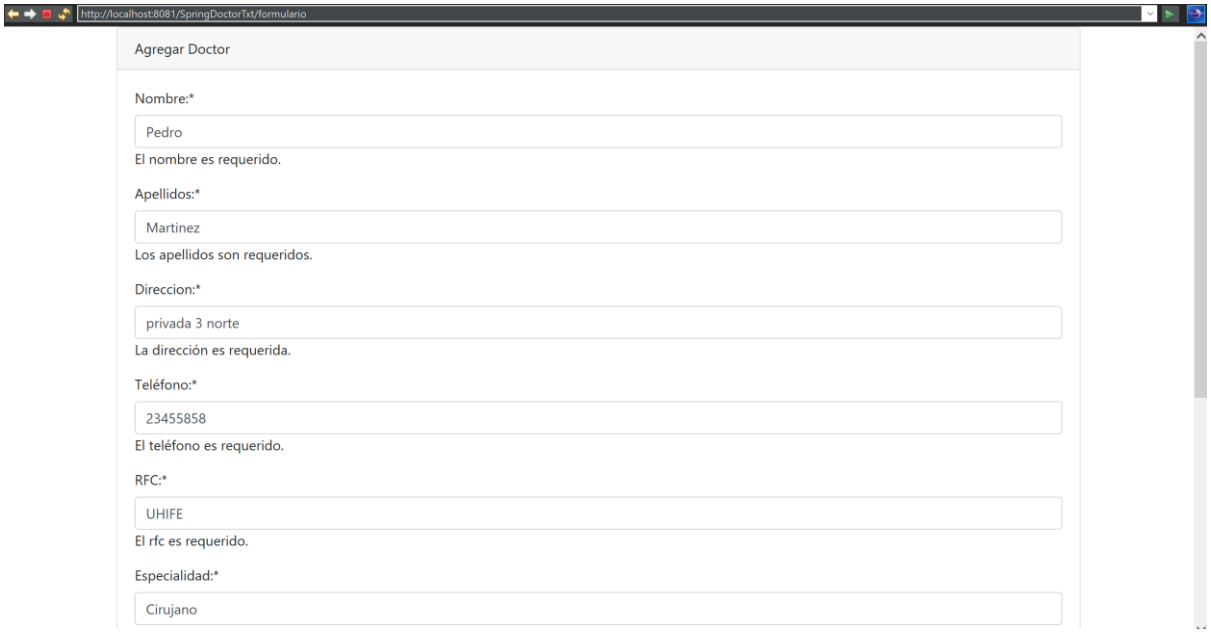
```java
package com.apress.springrecipes.court.domain;

import org.springframework.stereotype.Component;

@Component
public class DoctorValidator implements Validator {

    @Override
    public boolean supports(Class<?> arg0) {
        return Doctor.class.isAssignableFrom(arg0);
    }

    @Override
    public void validate(Object target, Errors errors) {
        ValidationUtils.rejectIfEmpty(errors, "nombre", "required.nombre", "El nombre es requerido.");
        ValidationUtils.rejectIfEmpty(errors, "apellidos", "required.apellidos", "Los apellidos son requeridos.");
        ValidationUtils.rejectIfEmpty(errors, "direccion", "required.direccion", "La dirección es requerida.");
        ValidationUtils.rejectIfEmpty(errors, "telefono", "required.telefono", "El teléfono es requerido.");
        ValidationUtils.rejectIfEmpty(errors, "rfc", "required.rfc", "El rfc es requerido.");
        ValidationUtils.rejectIfEmpty(errors, "especialidad", "required.especialidad", "La especialidad es requerida.");
        ValidationUtils.rejectIfEmpty(errors, "sexo", "required.sexo", "El tipo de sexo es reuqrido.");
        ValidationUtils.rejectIfEmpty(errors, "correo", "required.correo", "El correo es requerido.");
        ValidationUtils.rejectIfEmpty(errors, "contra", "required.contra", "La contraseña es requerida.");
        Doctor doctor = (Doctor) target;
    }

}
```

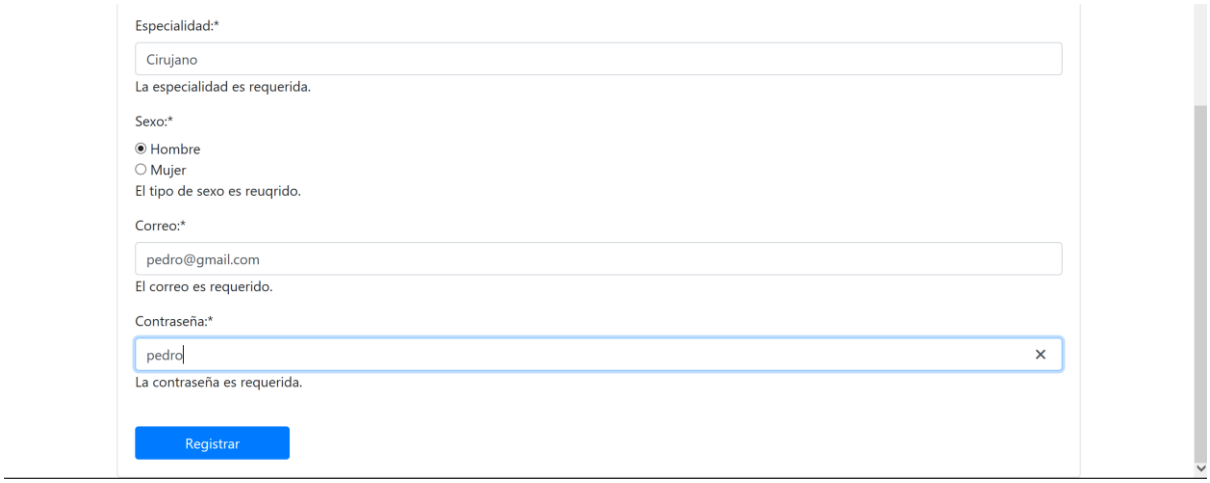
Resultados

Validación de los campos requeridos para el formulario de registro.

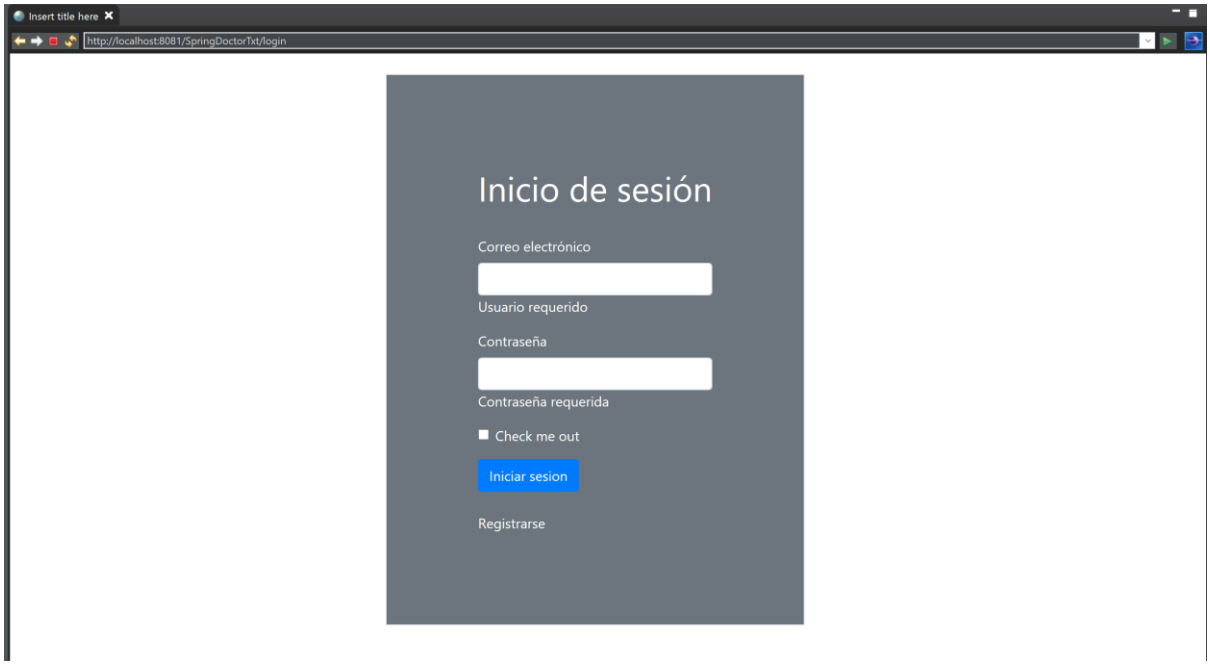


Captura de la información

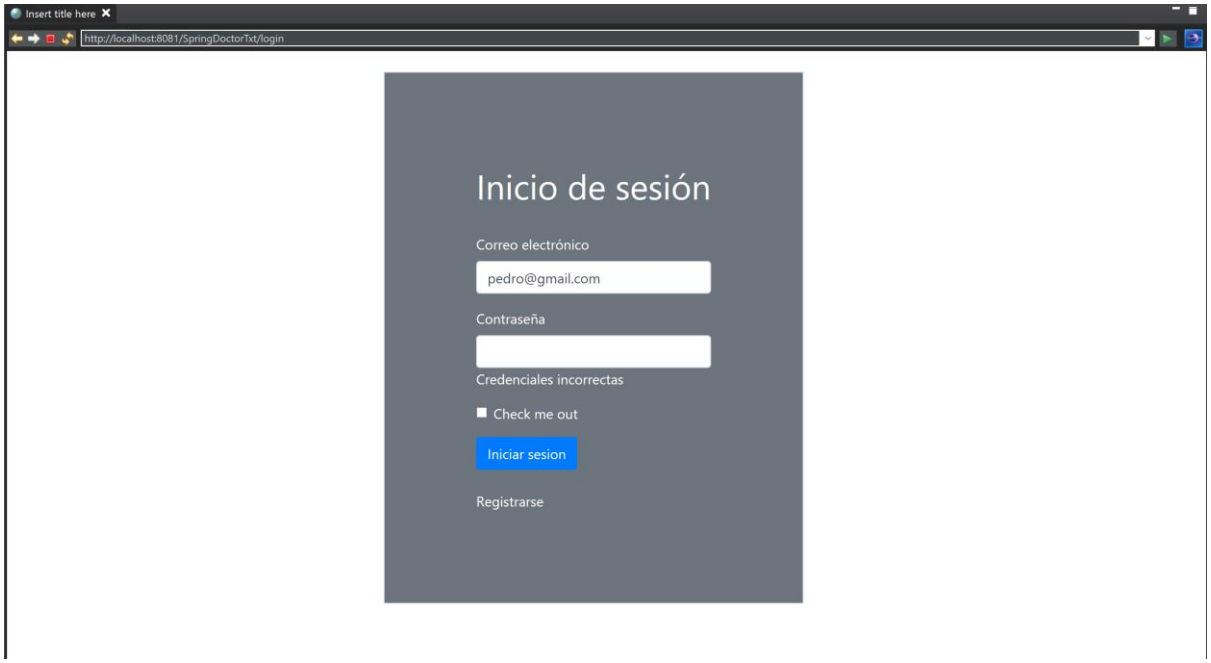




Campos requeridos para el formulario de inicio de sesión.



Validación de credenciales.



Inicio de sesión correcta.