

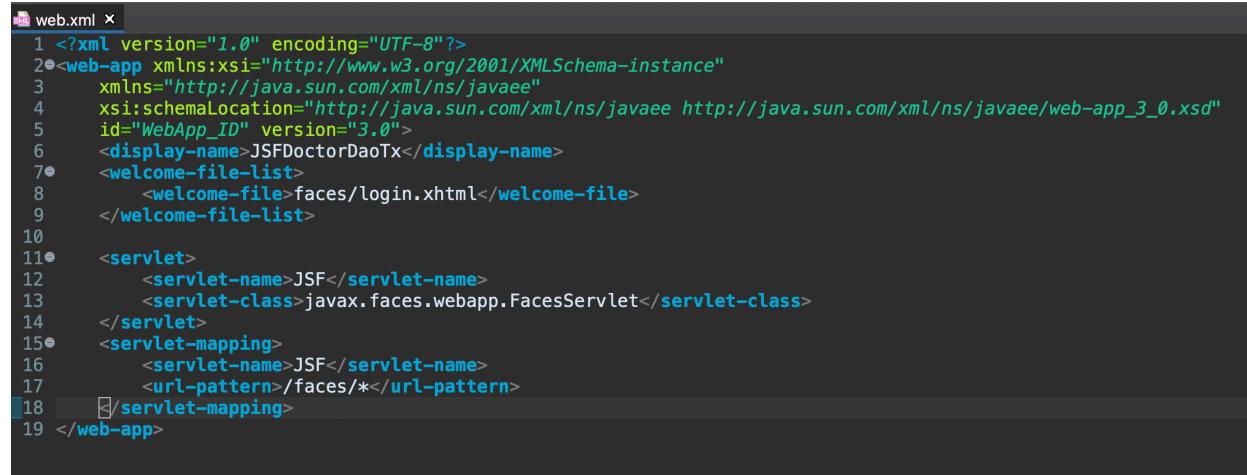
Alumno: Alejandro Herrera Mendoza

Descripción: Sistema web para crear una cuenta de doctor almacenada en base de datos Oracle con el fin de poder autenticarse, ver su información registrada y registrar su itinerario por semana.

Tecnologías: JSF, Oracle Database 11g, PrimeFaces.

Código

Archivo web.xml para realizar la configuración de JSF.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns="http://java.sun.com/xml/ns/javaee"
4   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
5   id="WebApp_ID" version="3.0">
6     <display-name>JSFDoctorDaoTx</display-name>
7     <welcome-file-list>
8       <welcome-file>faces/login.xhtml</welcome-file>
9     </welcome-file-list>
10    <servlet>
11      <servlet-name>JSF</servlet-name>
12      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
13    </servlet>
14    <servlet-mapping>
15      <servlet-name>JSF</servlet-name>
16      <url-pattern>/faces/*</url-pattern>
17    </servlet-mapping>
18  </web-app>
```

Archivo faces-config.xml para configurar las rutas de navegación con el fin moverse de una vista a otra.



```
1 <faces-config xmlns="http://java.sun.com/xml/ns/javaee"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
4   http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
5   version="2.0">
6
7
8    <navigation-rule>
9      <from-view-id>/login.xhtml</from-view-id>
10     <navigation-case>
11       <from-outcome>inicio</from-outcome>
12       <to-view-id>/registro.xhtml</to-view-id>
13     </navigation-case>
14   </navigation-rule>
15   <navigation-rule>
16     <from-view-id>/registro.xhtml</from-view-id>
17     <navigation-case>
18       <from-outcome>registro</from-outcome>
19       <to-view-id>/registro.xhtml</to-view-id>
20     </navigation-case>
21   </navigation-rule>
22   <navigation-rule>
23     <from-view-id>/login.xhtml</from-view-id>
24     <navigation-case>
25       <from-outcome>login</from-outcome>
26       <to-view-id>/login.xhtml</to-view-id>
27     </navigation-case>
28   </navigation-rule>
29
30   <navigation-rule>
31     <from-view-id>/registro.xhtml</from-view-id>
32     <navigation-case>
33       <from-outcome>registrado</from-outcome>
34       <to-view-id>/login.xhtml</to-view-id>
35     </navigation-case>
```

Agregar las .JAR necesarios para hacer uso de las librerías.



Archivo context.xml para configurar la conexión a la base de datos Oracle 11g.

```
<?xml version="1.0" encoding="UTF-8"?>
<context>
    <Resource name="jdbc/oracle" auth="Container"
              type="javax.sql.DataSource" maxActive="50" maxIdle="30"
              maxWait="10000" username="doctor" password="doctor"
              driverClassName="oracle.jdbc.OracleDriver"
              url="jdbc:oracle:thin:@localhost:1521:XE" />
</context>
```

The image shows a code editor window with a dark theme. The file is named "context.xml". The XML code defines a database connection resource named "jdbc/oracle". It specifies the type as "javax.sql.DataSource" and uses the "Container" authentication. The connection details include a maximum of 50 active connections, a maximum idle time of 30 seconds, a maximum wait time of 10,000 milliseconds, and a user named "doctor" with the same password. The driver class is set to "oracle.jdbc.OracleDriver" and the URL is "jdbc:oracle:thin:@localhost:1521:XE".

Clase RegistroMB.java para obtener los datos del formulario de Registro

```
RegistroMB.java
1 package com.bean;
2
3 import javax.faces.bean.ManagedBean;
4
5 @ManagedBean(name = "registro")
6 public class RegistroMB {
7
8     public String nombre;
9     public String apellidos;
10    public String direccion;
11    public String telefono;
12    public String rfc;
13    public String especialidad;
14    public String sexo;
15    public String correo;
16    public String contra;
17
18    public String getNombre() {
19        return nombre;
20    }
21
22    public void setNombre(String nombre) {
23        this.nombre = nombre;
24    }
25
26    public String getApellidos() {
27        return apellidos;
28    }
29
30    public void setApellidos(String apellidos) {
31        this.apellidos = apellidos;
32    }
33
34    public String getDireccion() {
35        return direccion;
36    }
37
38    public void setDireccion(String direccion) {
39        this.direccion = direccion;
40    }
41
42    public String getTelefono() {
43        return telefono;
44    }
45
46    public void setTelefono(String telefono) {
47        this.telefono = telefono;
48    }
49
50}
```

Método para enviar la información del registro al DAO.

```
93     public String registrarDoctor() {
94
95         RegistroService service = new RegistroDao();
96         if (service.createDoctor(
97             new Doctor(nombre, apellidos, direccion, telefono, rfc, especialidad, sexo, correo, contra))) {
98             return "registrado";
99         } else {
100            return "registro";
101        }
102    }
103
104 }
105
106 }
```

Clase RegistroDao.java para realizar las sentencias SQL del registro del doctor.

```
RegistroDao.java x
1 package com.dao;
2
3 import java.sql.Connection;
4
5 public class RegistroDao implements RegistroService {
6
7     private DataSource dataSource;
8
9     public RegistroDao() {
10        dataSource = null;
11        try {
12            Context ctx = new InitialContext();
13            dataSource = (DataSource) ctx.lookup("java:comp/env/jdbc/oracle");
14        } catch (NamingException e) {
15            e.printStackTrace();
16        }
17    }
18
19    @Override
20    public boolean createDoctor(Doctor doctor) {
21        String sql = "INSERT INTO doctor (id_doctor, nombre, apellidos, direccion, telefono, rfc, especialidad, sexo, correo, contra"
22                    + ") VALUES (seq_doc.NEXTVAL,?, ?, ?, ?, ?, ?, ?, ?)";
23
24        Connection conn = null;
25        try {
26            conn = dataSource.getConnection();
27
28            if (conn == null) {
29                throw new SQLException("No se puede obtener la conexión a la base de datos");
30            }
31            PreparedStatement ps = conn.prepareStatement(sql);
32            ps.setString(1, doctor.getNombre());
33            ps.setString(2, doctor.getApellidos());
34            ps.setString(3, doctor.getDireccion());
35            ps.setString(4, doctor.getTelefono());
36            ps.setString(5, doctor.getRfc());
37            ps.setString(6, doctor.getEspecialidad());
38            ps.setString(7, doctor.getSexo());
39            ps.setString(8, doctor.getCorreo());
40            ps.setString(9, doctor.getContra());
41            ps.executeUpdate();
42            ps.close();
43            return true;
44        } catch (SQLException e) {
45            e.printStackTrace();
46            return false;
47        } finally {
48            if (conn != null) {
```

Clase LoginMB.java para recuperar y manipular las credenciales del Doctor.

```
1 package com.bean;
2
3 import java.util.Date;
4
5
6 @ManagedBean(name = "login")
7 @SessionScoped
8 public class LoginMB {
9
10    public String usuario;
11    public String contra;
12    public Doctor doctor;
13    public String mensaje;
14    public String fecha;
15
16    public String getUsuario() {
17        return usuario;
18    }
19
20    public void setUsuario(String usuario) {
21        this.usuario = usuario;
22    }
23
24    public String getContra() {
25        return contra;
26    }
27
28    public void setContra(String contra) {
29        this.contra = contra;
30    }
31
32    public Doctor getDoctor() {
33        return doctor;
34    }
35
36    public void setDoctor(Doctor doctor) {
37        this.doctor = doctor;
38    }
39
40    public String getMensaje() {
41        return mensaje;
42    }
43
44    public void setMensaje(String mensaje) {
45        this.mensaje = mensaje;
46    }
47
48    public String getFecha() {
49        Date fecha = new Date();
50    }
51}
```

Clase para validar las credenciales del Doctor.

```
public String validarUsuario() {
    LoginService service = new LoginDao();
    doctor = service.login(usuario, contra);
    if (doctor != null) {
        return "inicioMenu";
    } else {
        mensaje = "Credenciales incorrectas";
        return "login";
    }
}
```

Clase LoginDao.java para ejecutar la sentencia SQL y verificar las credenciales del Doctor.

```
1 LoginDao.java x
2
3 import java.sql.Connection;
4
5 public class LoginDao implements LoginService {
6
7     private DataSource dataSource;
8
9     public LoginDao() {
10         dataSource = null;
11         try {
12             Context ctx = new InitialContext();
13             dataSource = (DataSource) ctx.lookup("java:comp/env/jdbc/oracle");
14         } catch (NamingException e) {
15             e.printStackTrace();
16         }
17     }
18
19     @Override
20     public Doctor login(String usuario, String contra) {
21         String sql = "SELECT id_doctor, nombre, apellidos, direccion, telefono, rfc, especialidad, sexo, correo, contra FROM doctor WHERE correo=? and contra=?";
22         Doctor doctor = null;
23         Connection conn = null;
24         try {
25             conn = dataSource.getConnection();
26             PreparedStatement ps = conn.prepareStatement(sql);
27             ps.setString(1, usuario);
28             ps.setString(2, contra);
29
30             ResultSet rs = ps.executeQuery();
31
32             if (rs.next()) {
33                 doctor = new Doctor(rs.getInt("id_doctor"), rs.getString("nombre"), rs.getString("apellidos"),
34                                     rs.getString("direccion"), rs.getString("telefono"), rs.getString("rfc"),
35                                     rs.getString("especialidad"), rs.getString("sexo"), rs.getString("correo"),
36                                     rs.getString("contra"));
37             }
38
39             return doctor;
40         } catch (SQLException e) {
41             e.printStackTrace();
42             return null;
43         }
44     }
45
46 }
```

Clase InicioMB.java para obtener y manipular la información del horario a crear.
Uso de @ManagedProperty para obtener los datos de otro ManagedBean.



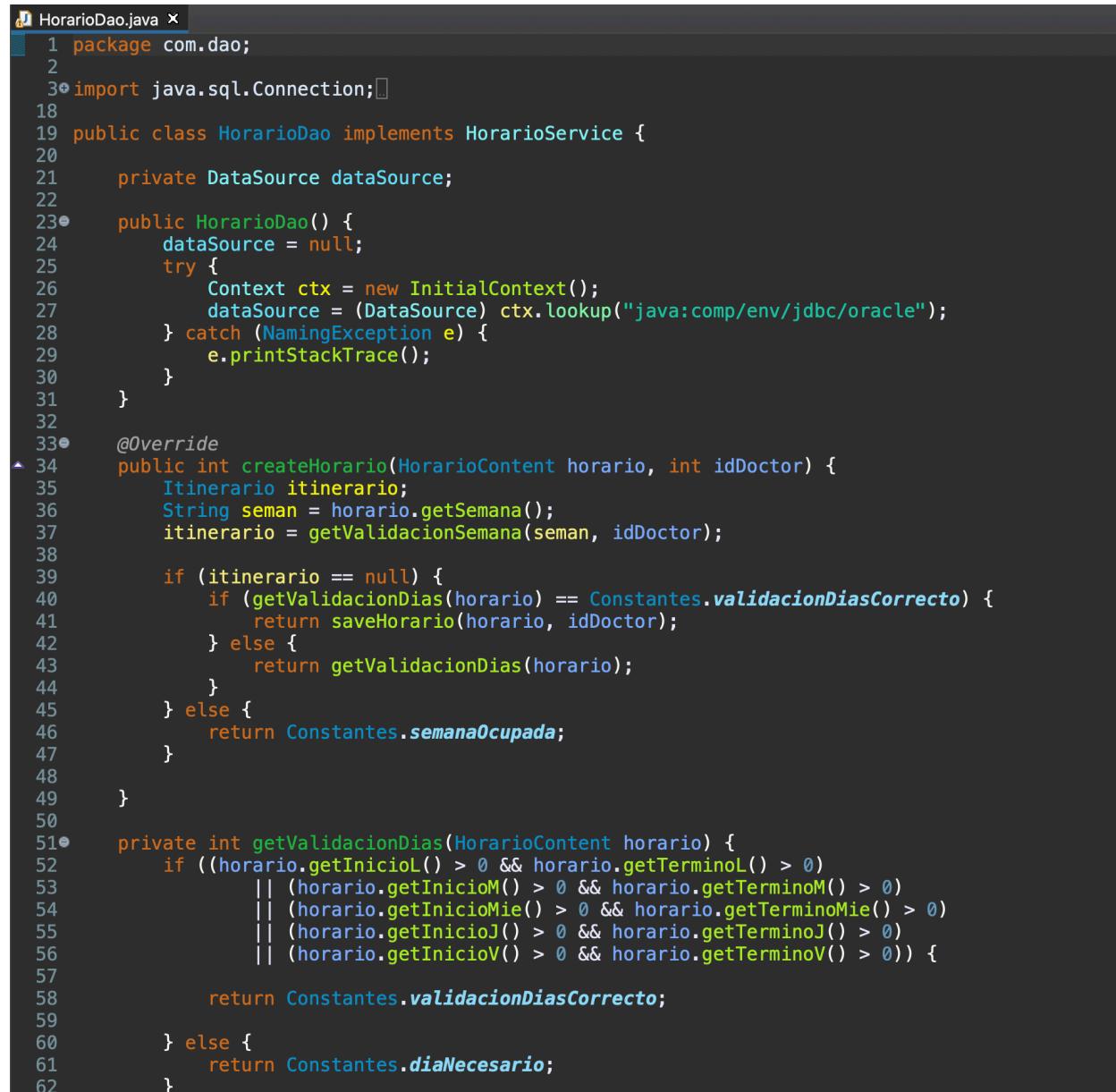
```
1 package com.bean;
2
3 import java.util.List;
4
5
6 @ManagedBean(name = "inicio")
7 public class InicioMB {
8
9     public List<Integer> semanas;
10    public String semana;
11    public String diaLunes;
12    public int inicioL;
13    public int terminoL;
14    public String diaMartes;
15    public int inicioM;
16    public int terminoM;
17    public String diaMiercoles;
18    public int inicioMie;
19    public int terminoMie;
20    public String diaJueves;
21    public int inicioJ;
22    public int terminoJ;
23    public String diaViernes;
24    public int inicioV;
25    public int terminoV;
26    public String mensaje;
27
28    @ManagedProperty("#{login}")
29    private LoginMB login;
30
31    public List<Integer> getSemanas() {
32        Select select = new Select();
33        semanas = select.getSemanas();
34        return semanas;
35    }
36
37    public void setSemanas(List<Integer> semanas) {
38        this.semanas = semanas;
39    }
40
41    public String getSemana() {
42        return semana;
43    }
44
45    public void setSemana(String semana) {
46        this.semana = semana;
47    }
48
49    public String getDiaLunes() {
```

Método para enviar los datos del horario y validar el tipo de respuesta.

```
public String crearHorario() {
    HorarioContent horario = new HorarioContent(semana, diaLunes, inicioL, terminoL, diaMartes, inicioM, terminoM,
        diaMiércoles, inicioMie, terminoMie, diaJueves, inicioJ, terminoJ, diaViernes, inicioV, terminoV);
    HorarioService service = new HorarioDao();

    switch (service.createHorario(horario, login.doctor.getIdDoctor())) {
        case Constantes.diaNecesario:
            mensaje = "Selecciona un día por lo menos";
            return "inicioUpdate";
        case Constantes.errorSaveHorario:
            mensaje = "No se pudo crear el horario";
            return "inicioUpdate";
        case Constantes.semanaOcupada:
            mensaje = "La semana seleccionada ya está ocupada";
            return "inicioUpdate";
        case Constantes.saveHorario:
            return "inicioUpdate";
        case Constantes.validacionDiasCorrecto:
            mensaje = "Días incorrectos";
            return "inicioUpdate";
        default:
            mensaje = "Error inesperado";
            return "inicioUpdate";
    }
}
```

Clase HorarioDao.java para ejecutar las validaciones respectivas al horario y ejecutar las sentencias SQL para crear un horario.



```
HorarioDao.java
1 package com.dao;
2
3 import java.sql.Connection;
4
5 public class HorarioDao implements HorarioService {
6
7     private DataSource dataSource;
8
9     public HorarioDao() {
10         dataSource = null;
11         try {
12             Context ctx = new InitialContext();
13             dataSource = (DataSource) ctx.lookup("java:comp/env/jdbc/oracle");
14         } catch (NamingException e) {
15             e.printStackTrace();
16         }
17     }
18
19     @Override
20     public int createHorario(HorarioContent horario, int idDoctor) {
21         Itinerario itinerario;
22         String seman = horario.getSemana();
23         itinerario = getValidacionSemana(seman, idDoctor);
24
25         if (itinerario == null) {
26             if (getValidacionDias(horario) == Constantes.validacionDiasCorrecto) {
27                 return saveHorario(horario, idDoctor);
28             } else {
29                 return getValidacionDias(horario);
30             }
31         } else {
32             return Constantes.semanaOcupada;
33         }
34     }
35
36     private int getValidacionDias(HorarioContent horario) {
37         if ((horario.getInicioL() > 0 && horario.getTerminoL() > 0)
38             || (horario.getInicioM() > 0 && horario.getTerminoM() > 0)
39             || (horario.getInicioMie() > 0 && horario.getTerminoMie() > 0)
40             || (horario.getInicioJ() > 0 && horario.getTerminoJ() > 0)
41             || (horario.getInicioV() > 0 && horario.getTerminoV() > 0)) {
42
43                 return Constantes.validacionDiasCorrecto;
44
45             } else {
46                 return Constantes.diaNecesario;
47             }
48
49     }
50
51     Itinerario getValidacionSemana(String semana, int idDoctor) {
52         String sql = "select id_itinerario, no_semana, inicioL, terminoL, diaLunes, inicioM, terminoM, diaMartes, inicioMie, terminoMie, diaMiércoles, inicio
53         Connection conn = null;
54         int valor = 0;
55         Itinerario itinerario = null;
56         try {
57             conn = dataSource.getConnection();
58             PreparedStatement ps = conn.prepareStatement(sql);
59             ps.setInt(1, Integer.parseInt(semana));
60             ps.setInt(2, idDoctor);
61             ResultSet rs = ps.executeQuery();
62             if (rs.next()) {
63                 itinerario = new Itinerario(rs.getInt("id_itinerario"), rs.getInt("no_semana"), rs.getInt("inicioL"),
64                     rs.getInt("terminoL"), rs.getString("diaLunes"), rs.getInt("inicioM"), rs.getInt("terminoM"),
65                     rs.getString("diaMartes"), rs.getInt("inicioMie"), rs.getInt("terminoMie"),
66                     rs.getString("diaMiércoles"), rs.getInt("inicioJ"), rs.getInt("terminoJ"),
67                     rs.getString("diaJueves"), rs.getInt("inicioV"), rs.getInt("terminoV"),
68                     rs.getString("diaViernes"), rs.getInt("id_dotor"));
69                 System.out.println("yaaa: " + itinerario);
70             }
71
72             return itinerario;
73         } catch (SQLException e) {
74             System.out.println(e.getMessage());
75             return null;
76         }
77     }
78 }
```

Método para validar semana.



```
Itinerario getValidacionSemana(String semana, int idDoctor) {
    String sql = "select id_itinerario, no_semana, inicioL, terminoL, diaLunes, inicioM, terminoM, diaMartes, inicioMie, terminoMie, diaMiércoles, inicio
    Connection conn = null;
    int valor = 0;
    Itinerario itinerario = null;
    try {
        conn = dataSource.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, Integer.parseInt(semana));
        ps.setInt(2, idDoctor);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            itinerario = new Itinerario(rs.getInt("id_itinerario"), rs.getInt("no_semana"), rs.getInt("inicioL"),
                rs.getInt("terminoL"), rs.getString("diaLunes"), rs.getInt("inicioM"), rs.getInt("terminoM"),
                rs.getString("diaMartes"), rs.getInt("inicioMie"), rs.getInt("terminoMie"),
                rs.getString("diaMiércoles"), rs.getInt("inicioJ"), rs.getInt("terminoJ"),
                rs.getString("diaJueves"), rs.getInt("inicioV"), rs.getInt("terminoV"),
                rs.getString("diaViernes"), rs.getInt("id_dotor"));
            System.out.println("yaaa: " + itinerario);
        }
        return itinerario;
    } catch (SQLException e) {
        System.out.println(e.getMessage());
        return null;
    }
}
```

Método para almacenar el horario en la base de datos.

```
private int saveHorario(HorarioContent horario, int idDoctor) {
    String sql = "insert into itinerario(id_itinerario, no_semana, inicioL, terminoL, diaLunes, inicioM, terminoM, diaMartes, inicioMie, terminoMie, diaMie
    + "VALUES(seq_iti.nextval, ?,?,?,?,?,?,?,?,?,?,?);";
    Connection conn = null;
    try {
        conn = dataSource.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, Integer.parseInt(horario.getSemana()));
        ps.setInt(2, horario.getInicioL());
        ps.setInt(3, horario.getTerminoL());
        ps.setString(4, horario.getDiaLunes());
        ps.setInt(5, horario.getInicioM());
        ps.setInt(6, horario.getTerminoM());
        ps.setString(7, horario.getDiaMartes());
        ps.setInt(8, horario.getInicioMie());
        ps.setInt(9, horario.getTerminoMie());
        ps.setString(10, horario.getDiaMiercoles());
        ps.setInt(11, horario.getInicioJ());
        ps.setInt(12, horario.getTerminoJ());
        ps.setString(13, horario.getDiaJueves());
        ps.setInt(14, horario.getInicioV());
        ps.setInt(15, horario.getTerminoV());
        ps.setString(16, horario.getDiaViernes());
        ps.setInt(17, idDoctor);
        ps.executeUpdate();
        ps.close();
        return Constantes.saveHorario;
    } catch (SQLException e) {
        System.out.println(e.getMessage());
        return Constantes.errorSaveHorario;
    }
}
```

RESULTADOS

Insert title here

localhost:8080/JSFDoctorOracle/faces/registro.xhtml?productId=

Registro

Nombre	Manuel
Apellidos	Mendoza Pérez
Direccion	av 6 sur
Telefono	249264747
Especialidad	Veterinario
Rfc	HDMGD755
Sexo:	<input checked="" type="radio"/> Hombre <input type="radio"/> Mujer
Correo	manuel@gmail.com
Contraseña	manuel

Registrar

Script Output | Query Result | All Rows Fetched: 10 in 0.008 seconds

SQL

ID_DOCTOR	NOMBRE	APELLIDOS	DIRECCION	TELEFONO	RFC	ESPECIALIDAD	SEXO	CORREO	CONTRA
1	Alejandro Herrera	Mendoza	Privada 16 de septiembre	24937474	HEMAHDHF76	Java	Hombre	alex@gmail.com	pass
2	Maria	Rosas	8 norte	249736474	RJFJ76	(null)	(null)	maria@gmail.com	password
3	Marco	Mendoza	8 sur este	24916745474	MARDGF765	(null)	Hombre	marco@gmail.com	marco
4	Martha	Sánchez	av 5 sur	249234657	MDRF655	(null)	Mujer	martha@gmail.com	martha
5	hgjg	hgjg	hgj	hgjg	Hgj	(null)	Hombre	ghjk@gmail.com	Hgjkh
6	ghjg	hgjg	hgj	hgk	Hgk	(null)	Mujer	ghjk@gmail.com	Hgjkg
7	Pedro	Rosas	av 5 sur	2494757	HDGG66	java	Hombre	pedro@gmail.com	pedro
8	Alejandro Herrera	Mendoza	privada 16 de septiembre	24915637	HEMAHG76	java	Hombre	alex@gmail.com	password
9	Alejandro Herrera	Mendoza	privada 16 de septiembre	24915637	HEMAHG76	java	Hombre	alex@gmail.com	password
10	Manuel	Mendoza Pérez	av 6 sur	249264747	HDMGD755	Veterinario	Hombre	manuel@gmail.com	manuel

Login

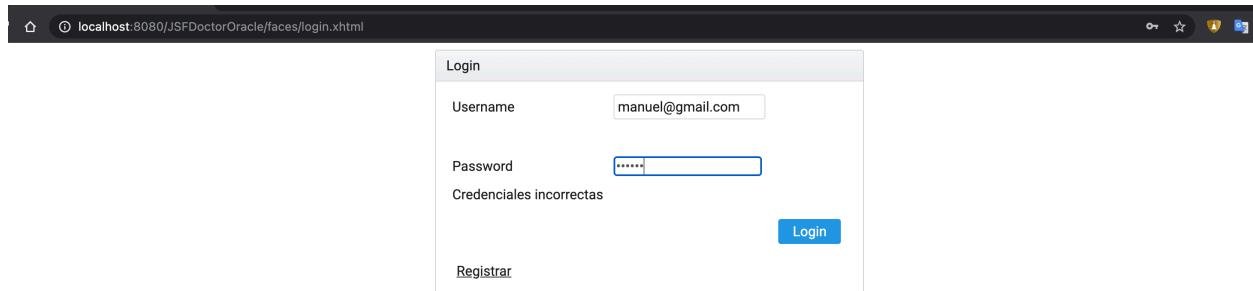
Username

Password

Credenciales incorrectas

Login

Registrar



Bienvenido Doctor(a):



Manuel Mendoza Pérez

Información

Dirección: av 6 sur

Especialidad: Veterinario

Rfc: HDMDG755

Sexo: Hombre

Fecha: Tue Dec 29 16:04:34 CST 2020

Crear horario

Semanas		Hora inicio	Hora termino
1	↓		
Dia			
Lunes		0	0
Martes		0	0
Miercoles		0	0
Jueves		0	0
Viernes		0	0

Crear horario

Semanas		Hora inicio	Hora termino
1	↓		
Dia			
Lunes		0	0
Martes		0	0
Miercoles		0	0
Jueves		0	0
Viernes		0	0

Selecciona un día por lo menos

Crear

Crear horario

Semanas

Dia	Hora inicio	Hora termino
Lunes	8	8
Martes	0	0
Miercoles	0	0
Jueves	0	0
Viernes	0	0

Selecciona un día por lo menos

Crear

Script Output | Query Result | SQL | All Rows Fetched: 4 in 0.005 seconds

HORARIO	NO_SEMANA	INICIOI	TERMINOI	DIALUNES	INICIOM	TERMINOM	DIAMARTES	INICIOMIE	TERMINOMIE	DIAMIERCOLES	INICIOJ	TERMINOJ	DIAJUEVES	INICIOV	TERMINOV	DIAVIERNES	ID_DOCTOR
1	1	3	6	6 Lunes	7	8	8 Martes	8	8	8 Miercoles	8	8	8 Jueves	8	8	8 Viernes	21
2	2	1	9	9 Lunes	9	9	9 Martes	9	9	9 Miercoles	9	9	9 Jueves	9	9	9 Viernes	21
3	3	2	3	3 Lunes	0	0	0 Martes	0	0	0 Miercoles	0	0	0 Jueves	0	0	0 Viernes	21
4	4	1	8	8 Lunes	0	0	0 Martes	0	0	0 Miercoles	0	0	0 Jueves	0	0	0 Viernes	41

Crear horario

Semanas

▼

Dia	Hora inicio	Hora termino
Lunes	0	0
Martes	0	0
Miercoles	0	0
Jueves	0	0
Viernes	0	0

La semana seleccionada ya está ocupada

Crear

Crear horario

Semanas

▼

Dia	Hora inicio	Hora termino
Lunes	9	7
Martes	6	8
Miercoles	5	9
Jueves	6	9
Viernes	4	8

Crear

Script Output x Query Result x

SQL All Rows Fetched: 5 in 0.003 seconds

TIERARIO	NO_SEMANA	INICIOI	TERMINOI	DIALUNES	INICIOM	TERMINOM	DIAMARTES	INICIOMIE	TERMINOMIE	DIAMIERCOLES	INICIOJ	TERMINOJ	DIAJUEVES	INICIOV	TERMINOV	DIAVIERNES	ID_DOCTOR
1	1	3	6	6 Lunes	7	8 Martes	8	8 Miércoles	8	8 Jueves	8	8 Viernes	21				
2	2	1	9	9 Lunes	9	9 Martes	9	9 Miércoles	9	9 Jueves	9	9 Viernes	21				
3	3	2	3	3 Lunes	0	0 Martes	0	0 Miércoles	0	0 Jueves	0	0 Viernes	21				
4	4	1	8	8 Lunes	0	0 Martes	0	0 Miércoles	0	0 Jueves	0	0 Viernes	41				
5	5	3	9	7 Lunes	6	8 Martes	5	9 Miércoles	6	9 Jueves	4	8 Viernes	41				