Facultad Regional Rosario
Universidad Tecnológica Nacional

APROBACIÓN DIRECTA - LENGUAJE DE PROGRAMACIÓN JAVA

TRABAJO PRÁCTICO FINAL CENTRO DE ESTÉTICA

ALUMNOS

51445 - DE BERNARDO, AARÓN

51242 - GRAMAGLIA, FRANCISCA

PROFESORES

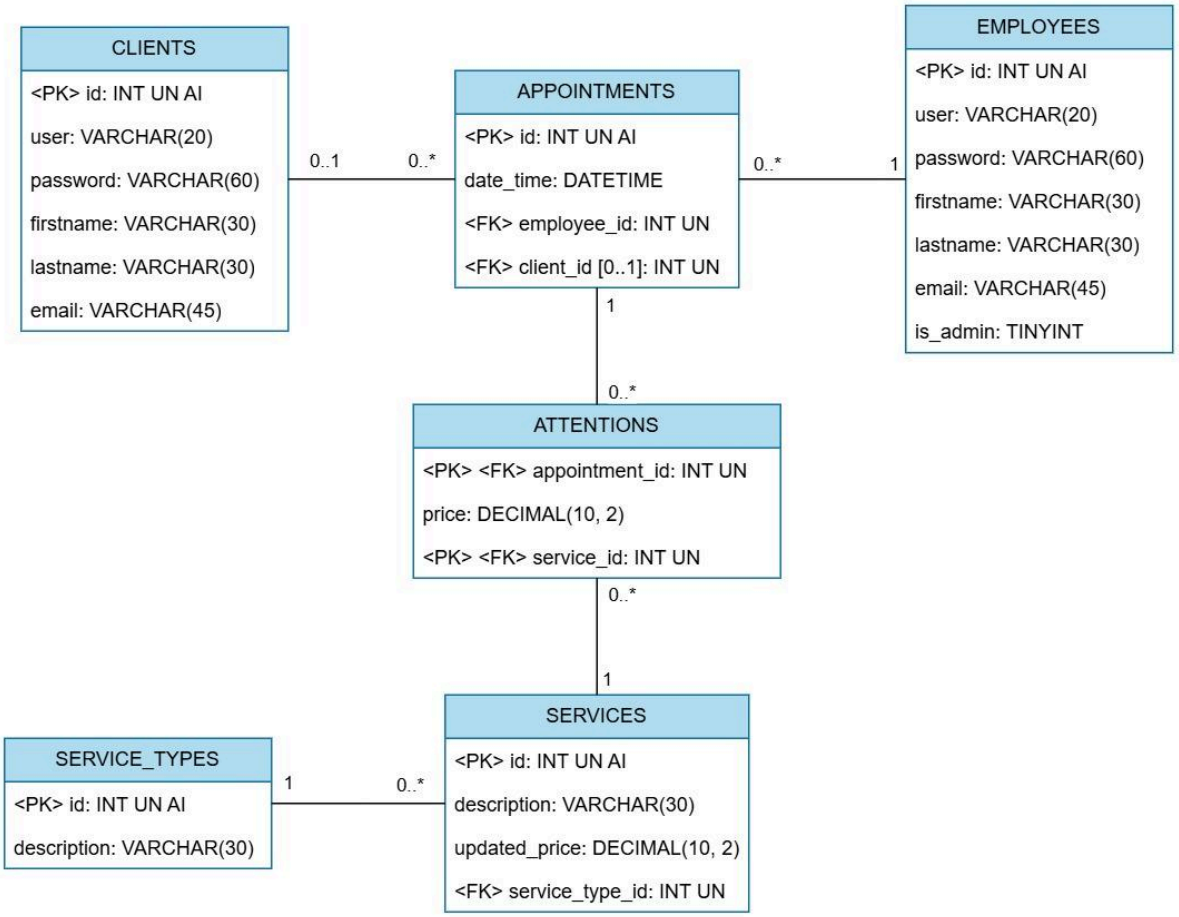
MECA, ADRIAN

TABACMAN, RICARDO

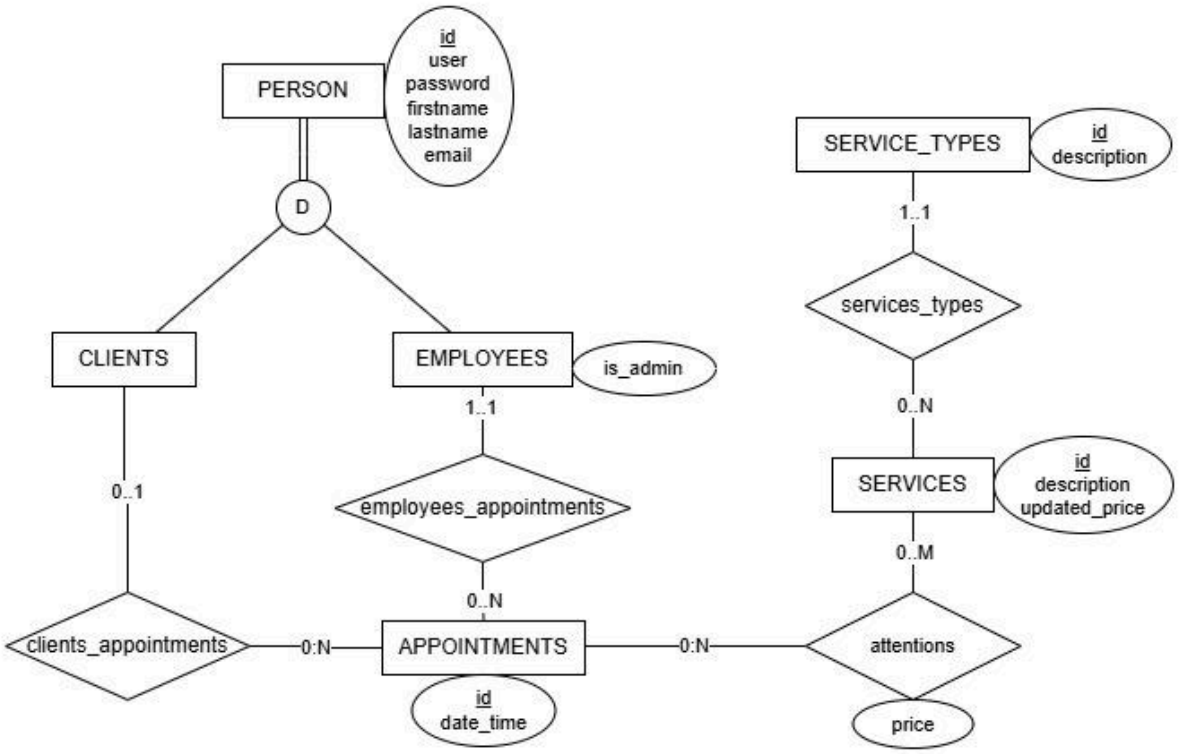
Índice

Diagrama de Clase **3**
Modelo de Datos **4**
CU resumen reestructurado **5**
CU de usuario principales sin reestructurar **6**
Caso de Uso 05 - Generar un turno 6
Caso de Uso 06 - Reservar un turno 11
Capturas de pantalla de todas las pantallas **14**
Fragmentos de código - CUU06 Reservar un turno **23**
Información adicional **34**

Diagrama de Clase



Modelo de Datos



CU resumen reestructurado

CU - Caso de Uso - Centro de Estética

Nivel de la meta: **Resumen** - Alcance del Caso de Uso: **Sistema** - Caja: **Negra**

Instanciación: **Real** - Interacción: **Semántica** - Usabilidad: **No Contemplada**

ACTORES Primario: **Cliente** - Iniciador: **Empleado** (Administrador - Empleado)

PRECONDICIONES: <vacío>

DISPARADOR: El empleado desea generar un turno.

POSTCONDICIONES: Éxito: El cliente fue atendido y las atenciones realizadas fueron registradas.

Fracaso: El cliente no pudo ser atendido. Las atenciones no fueron registradas.

Paso	**Acción**
1	El empleado se loguea en el sistema invocando al CUU01 - Iniciar sesión
2	El empleado registra los tipos de servicios invocando al CUU02 - Registrar tipo de servicio
3	El empleado crea los servicios invocando al CUU03 - Registrar servicio
4	El empleado registra al nuevo cliente invocando al CUU04 - Registrar cliente
5	El empleado genera un turno invocando al CUU05 - Generar un turno
6	El cliente se loguea en el sistema invocando al CUU01 - Iniciar sesión
7	El cliente reserva un turno invocando al CUU06 - Reservar un turno
8	El cliente sale del sistema invocando al CUU07 - Cerrar sesión
9	El empleado registra las atenciones realizadas invocando al CUU08 - Registrar atención
10	El empleado sale del sistema invocando al CUU07 - Cerrar sesión

CU de usuario principales sin reestructurar

CU - Caso de Uso - Centro de Estética

Caso de Uso 05 - Generar un turno

Nivel de la meta: **Usuario** - Alcance del Caso de Uso: **Sistema** - Caja: **Negra**

Instanciación: **Real** - Interacción: **Dialogal** - Usabilidad: **No Contemplada**

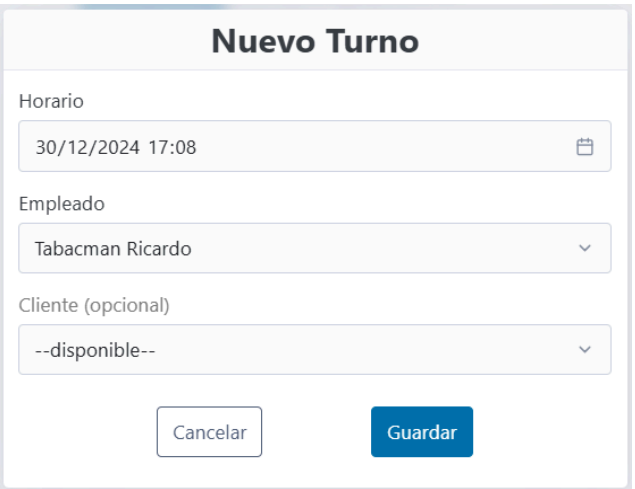
ACTORES Primario: **Administrador o Empleado** - Iniciador: **Administrador o Empleado**

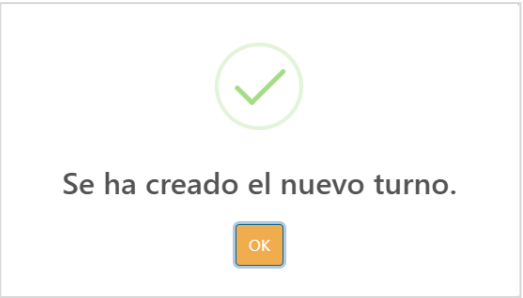
PRECONDICIONES: El administrador o empleado está logueado

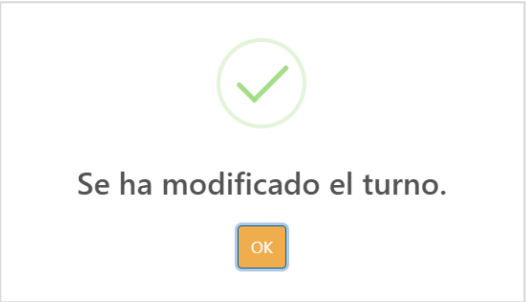
DISPARADOR: Se desea generar un turno.

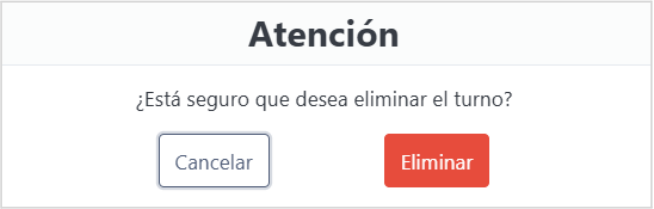
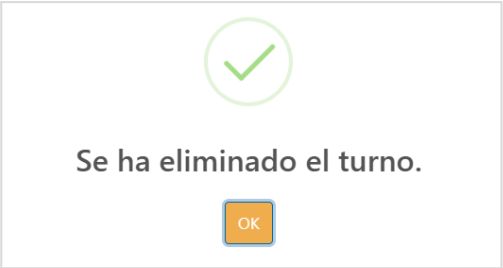
POSTCONDICIONES: Éxito: Se ha generado un turno.
Éxito alternativo: Se ha modificado o eliminado un turno.
Fracaso: No se ha generado, modificado ni eliminado un turno.

Paso		Usuario / Condicion	Sistema
1			Muestra fecha y hora, empleado y el cliente, en caso de que haya, de todos los turnos próximos a la fecha y hora actual.
	Sist		
2		El administrador o empleado presiona el botón “Nuevo Turno”.	Muestra en una ventana emergente el formulario para crear un nuevo turno.

	Usu		
	Sist		
3		El administrador o empleado completa los datos del formulario y presiona el botón “Guardar”.	Valida que haya seleccionado una fecha y hora y un empleado. Valida que la fecha y hora sean posterior a la fecha y hora actual. Registra el nuevo turno. Muestra mensaje de confirmación.
	Usu		

	Sist		
	3.a	No se ha seleccionado una fecha y hora y/o empleado.	
	3.a.1		Informa la situación al usuario. Vuelve al paso 3.
	3.b	La fecha y hora es anterior a la fecha actual.	
	3.b.1		Informa la situación al usuario. Vuelve al paso 3.
	3.c <previo>	**El administrador o empleado desea modificar un turno.**	
	3.c.1	El administrador o empleado selecciona un turno y presiona el botón “Modificar”.	Muestra en una ventana emergente el formulario para modificar el turno seleccionado.
	Usu		

	Sist		
	3.c.2	El administrador o empleado modifica los datos y presiona el botón “Guardar”.	Valida que los campos estén completados. Valida que la fecha y hora sean posterior a la fecha y hora actual. Actualiza el turno seleccionado. Muestra mensaje de confirmación. Fin CU.
	Usu		
	Sist		
	3.c.2.a	El administrador o empleado no completó todos los campos.	

	3.c.2.a.1		Informa la situación al usuario. Vuelve al paso 3.c.2.
	3.c.2.b	La fecha y hora es anterior a la fecha actual.	
	3.c.2.b.1		Informa la situación al usuario. Vuelve al paso 3.c.2.
	3.d <previo>	**El administrador o empleado desea eliminar un turno.**	
	3.d.1	El administrador o empleado selecciona un turno y presiona el botón “Eliminar”.	Muestra en una ventana emergente el formulario para eliminar el turno seleccionado.
	Usu		
	Sist		
	3.d.2	El administrador o empleado presiona el botón “Eliminar”.	Elimina el turno seleccionado. Muestra mensaje de confirmación. Fin CU.
	Sist		

CU - Caso de Uso - Centro de Estética

Caso de Uso 06 - Reservar un turno

Nivel de la meta: **Usuario** - Alcance del Caso de Uso: **Sistema** - Caja: **Negra**

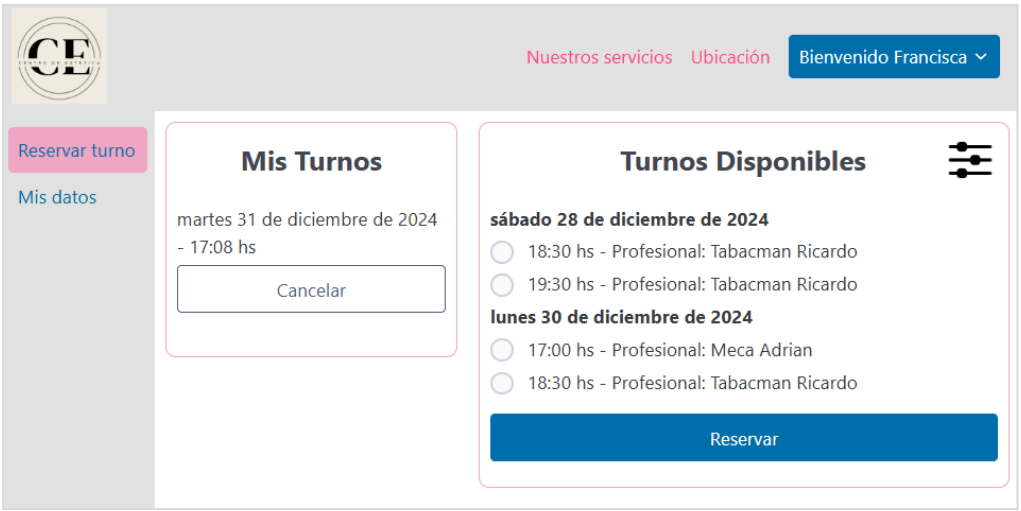
Instanciación: **Real** - Interacción: **Dialogal** - Usabilidad: **No Contemplada**

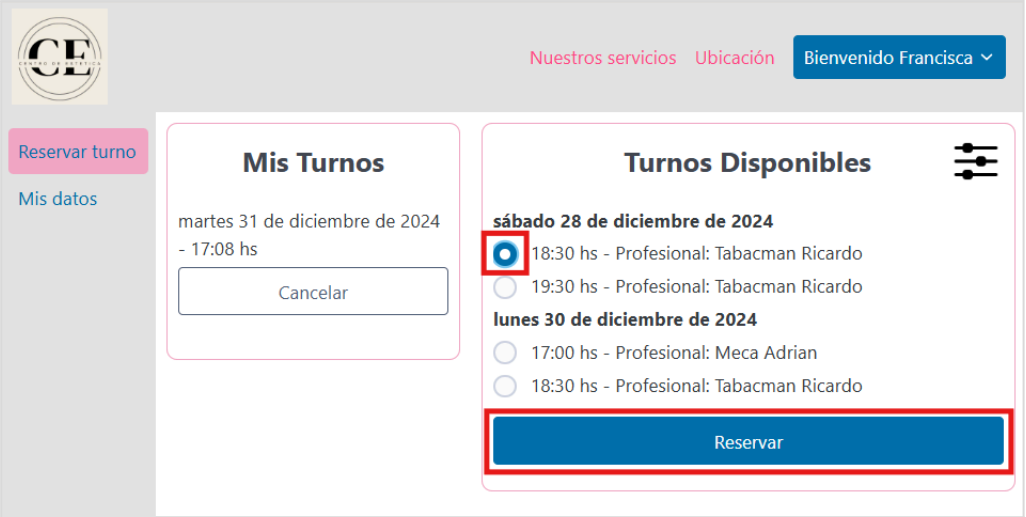
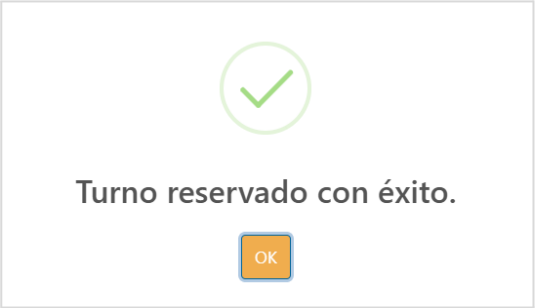
ACTORES Primario: **Cliente** - Iniciador: **Cliente**

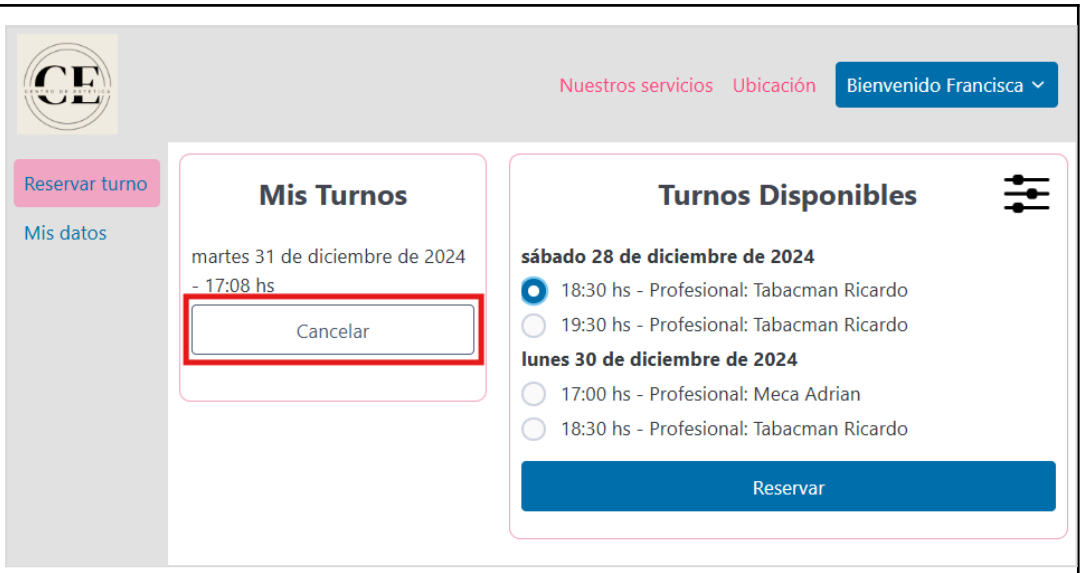
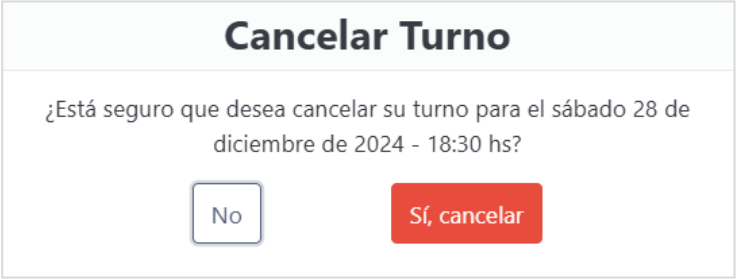
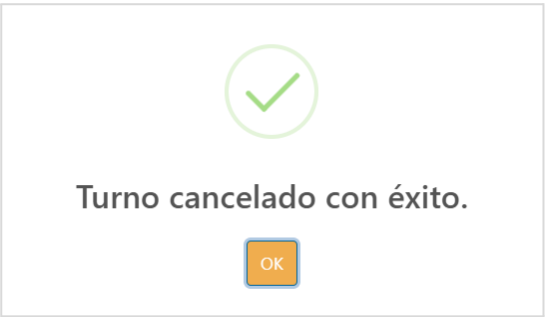
PRECONDICIONES: El cliente se encuentra logueado.

DISPARADOR: El cliente desea reservar un turno.

POSTCONDICIONES: Éxito: Se ha reservado el turno.
Éxito alternativo: Se ha cancelado un turno.
Fracaso: No se ha reservado el turno.

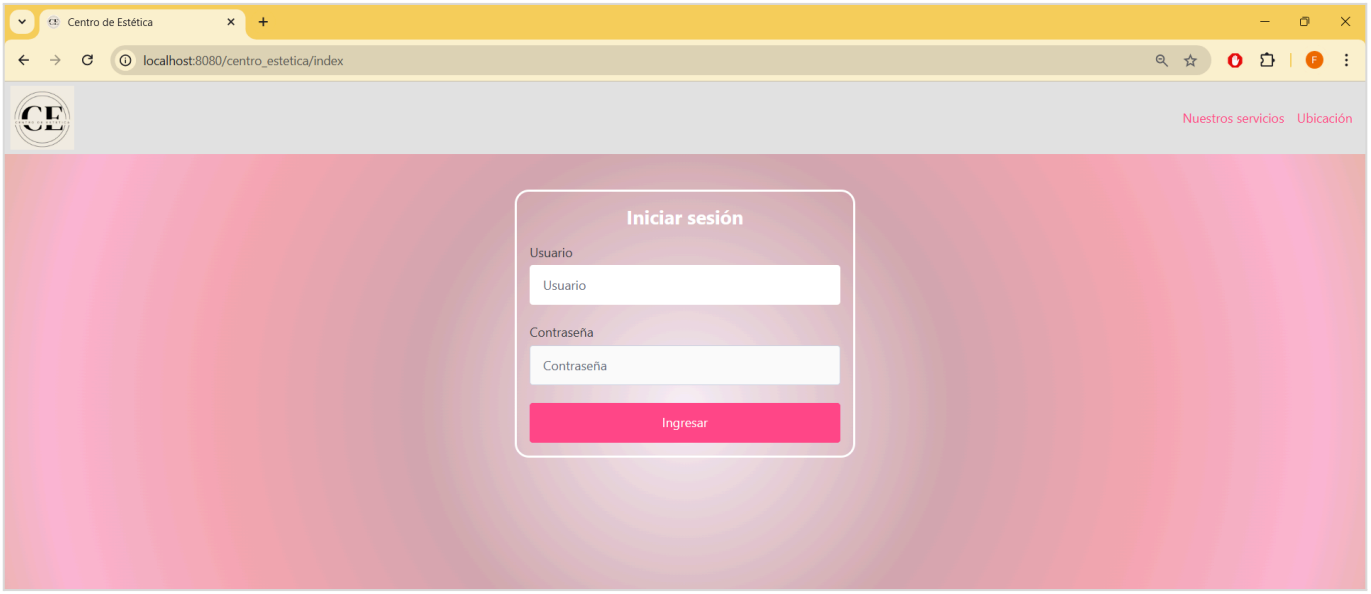
Paso		Usuario / Condicion	Sistema
1			Muestra los turnos del cliente con fecha y hora posterior a la actual, en caso de que haya, con su fecha y hora. Muestra los turnos disponibles, en caso de que haya, y para cada fecha muestra la hora y profesional del turno.
	Sist		
2		El cliente selecciona para una fecha una hora y un profesional, y presiona el botón “Reservar”.	Valida que el turno seleccionado siga disponible. Actualiza el turno. Muestra mensaje de confirmación y envía un mail al usuario confirmando la reserva del turno.

	Usu		
	Sist		
	2.a	El cliente no ha seleccionado un turno.	
	2.a.1		Informa la situación al usuario. Vuelve al paso 2.
	2.b	El turno seleccionado no sigue disponible.	
	2.b.1		Informa la situación al usuario. Vuelve al paso 2.
	2.c	No hay turnos disponibles.	
	2.c.1		Fin CU.
	2.d <previo>	**El cliente desea cancelar un turno.**	
	2.d.1	El cliente presiona el botón “Cancelar” que se encuentra debajo de cada turno de la sección “Mis Turnos”.	Muestra en una ventana emergente el formulario para cancelar el turno seleccionado con la fecha y hora de dicho turno.

	Usu		
	Sist		
	2.d.1.a	El cliente no tiene turnos.	
	2.d.1.a.1		Fin CU.
	2.d.2	El cliente presiona el botón “Sí, cancelar”.	Actualiza el turno. Muestra mensaje de confirmación y envía un mail al usuario confirmando la cancelación del turno.
	Sist		

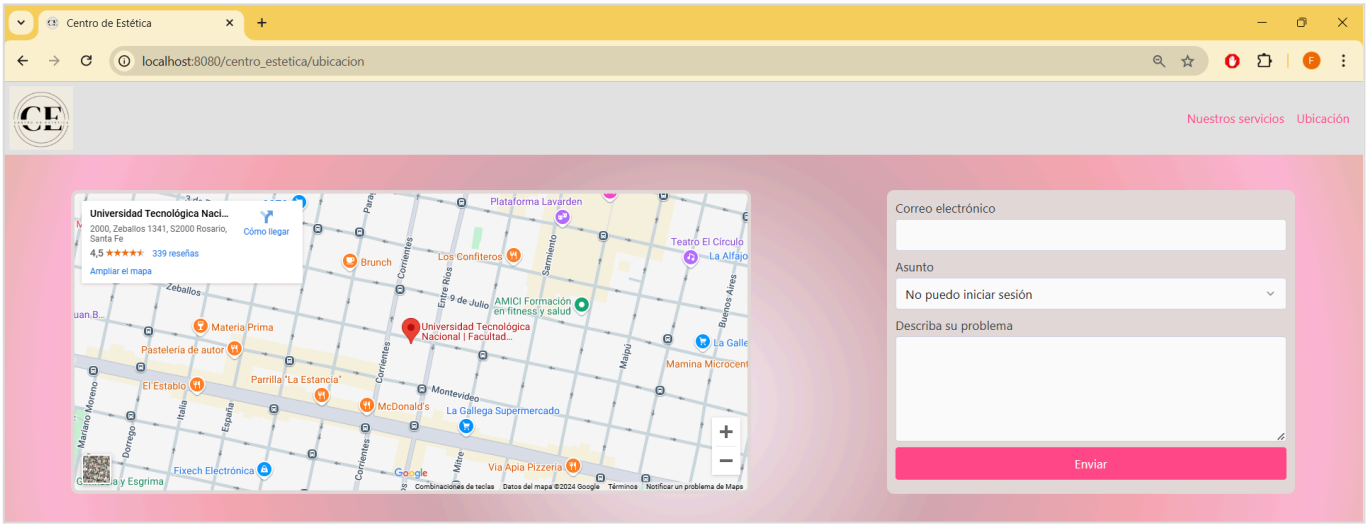
Capturas de pantalla de todas las pantallas

Pantalla de Inicio de Sesión

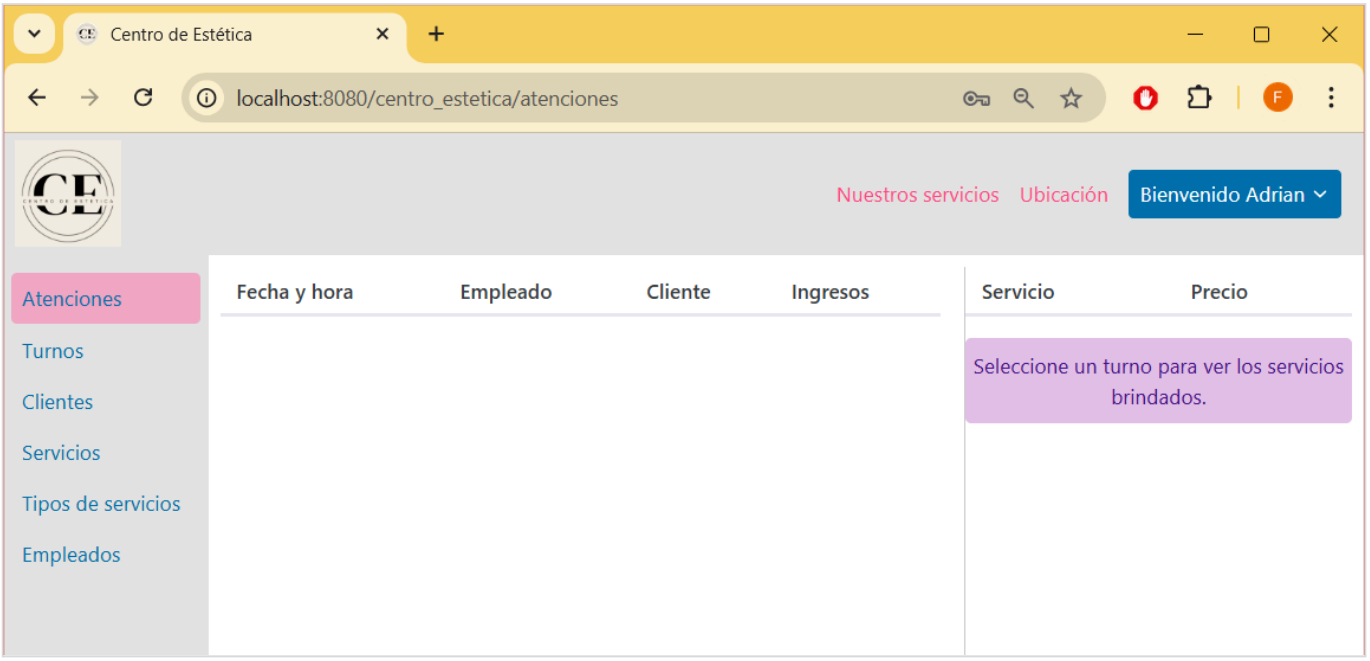


Pantalla Nuestros Servicios - Barra Superior

Pantalla Ubicacion - Barra Superior

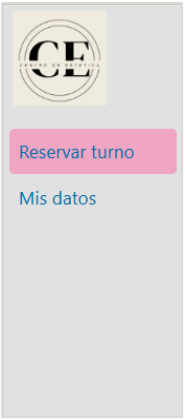


Menú lateral para usuario administrador



Menú lateral para empleado

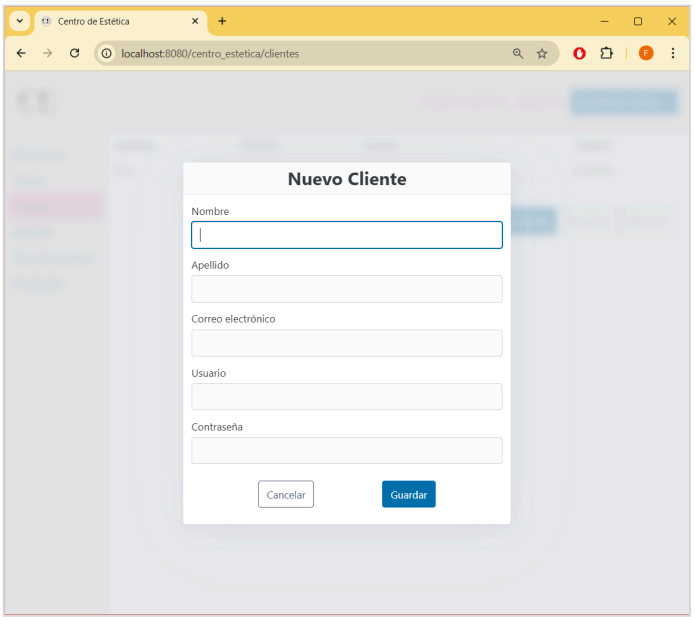
Menú lateral para cliente

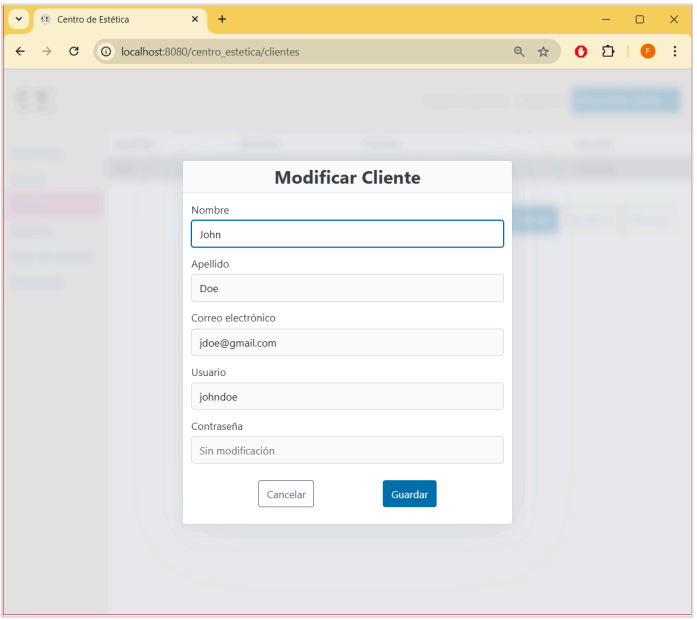


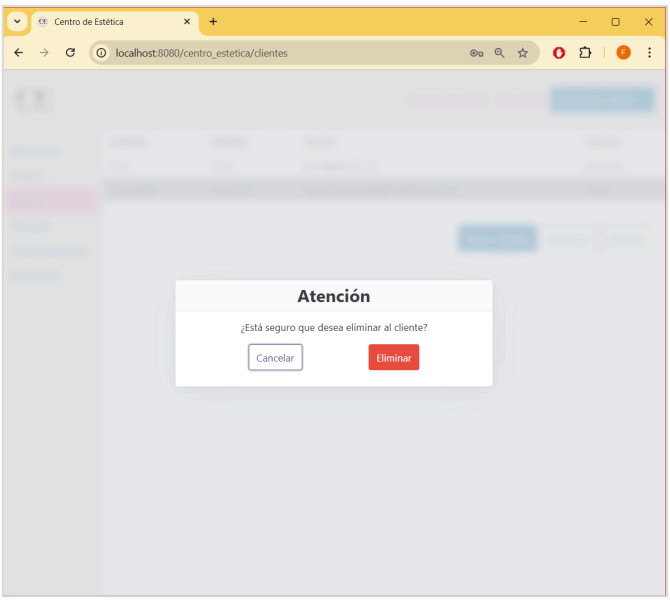
Pantalla Atenciones - Administrador

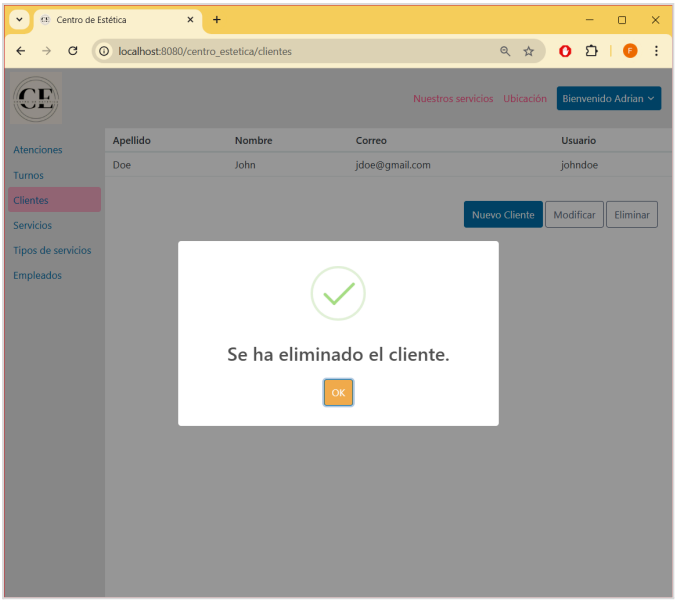
Pantalla Turnos - Administrador

Pantallas ABM Cliente



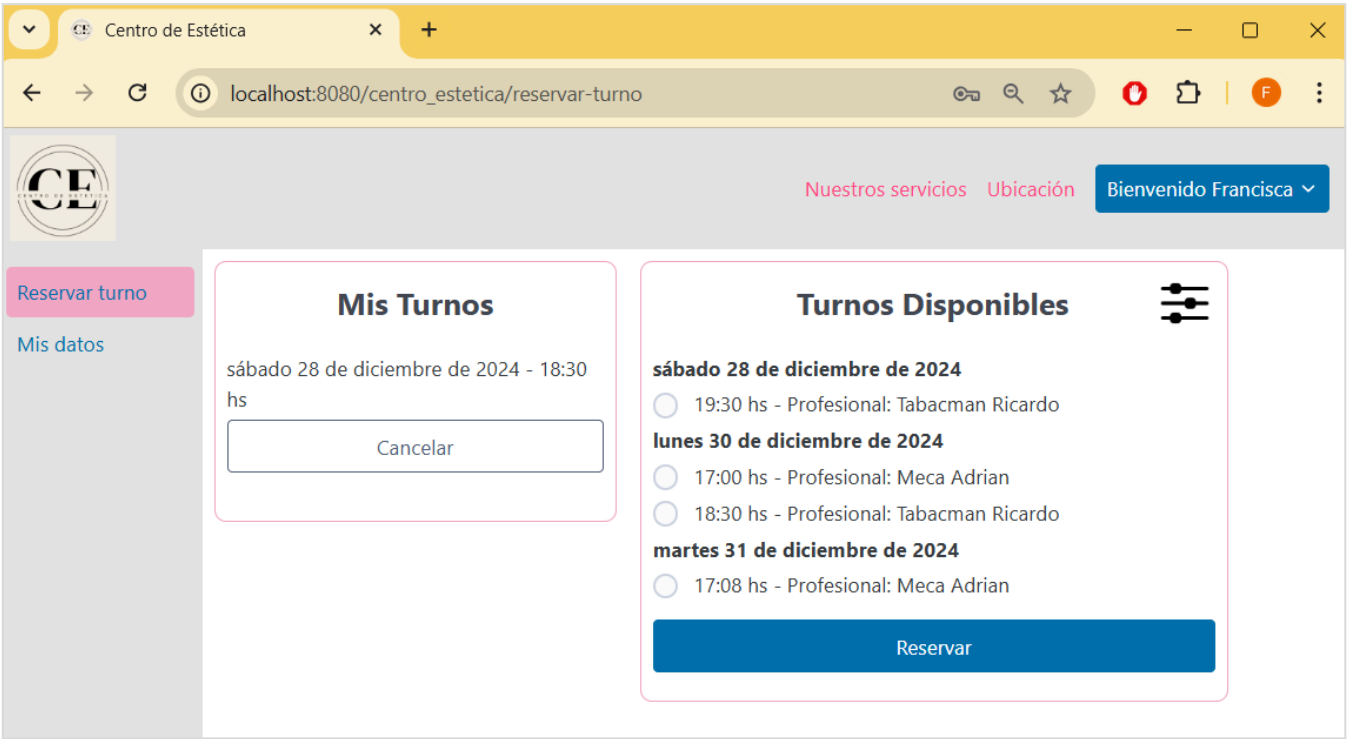




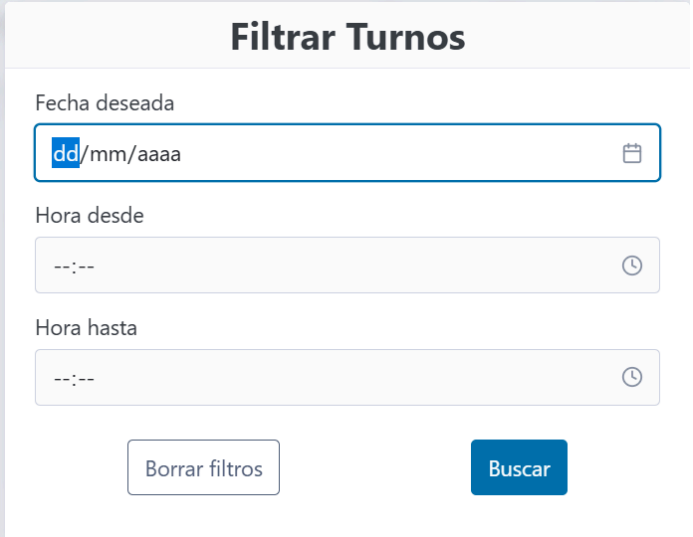


Nota: El resto de los ABM son similares.

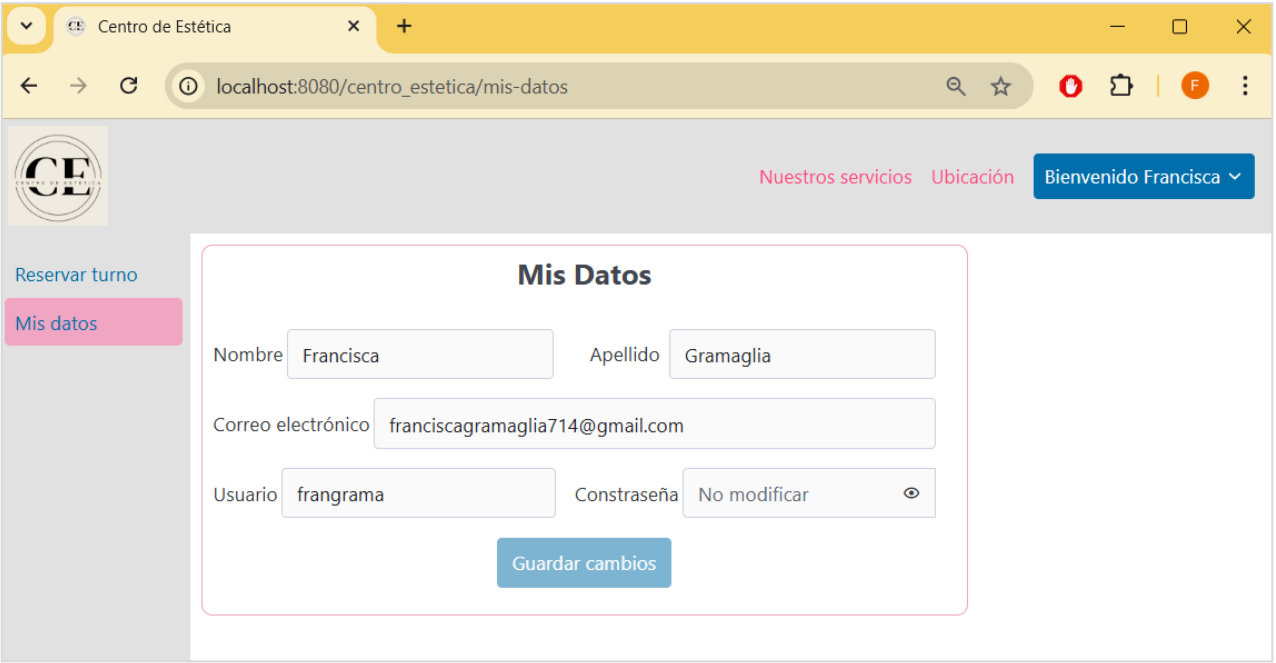
Pantalla Reservar Turno - Cliente



Cuadro de diálogo para filtrar turnos



Pantalla Mis Datos - Cliente



Pantalla de atenciones en pantalla de un celular.

QR Mercadopago generado en pantalla atenciones.

Pantalla Mail - Confirmación de Reserva

Turno reservado - Centro de Estética

From: <centroestetica2024@gmail.com>
To: <tony@stark.com>

Show Headers

HTML | HTML Source | Text | Raw | Spam Analysis | HTML Check 1 | Tech Info

Estimado/a Stark Tony,

Este es un correo de confirmación para informarte que tu turno ha sido reservado correctamente.

Detalles de tu cita:

- **Fecha y hora del turno:** lunes 23 de diciembre de 2024 - 09:00 hs
- **Profesional a cargo:** Banner Bruce

Si tienes alguna pregunta o necesitas realizar algún cambio en tu cita, no dudes en ponerte en contacto con nosotros.

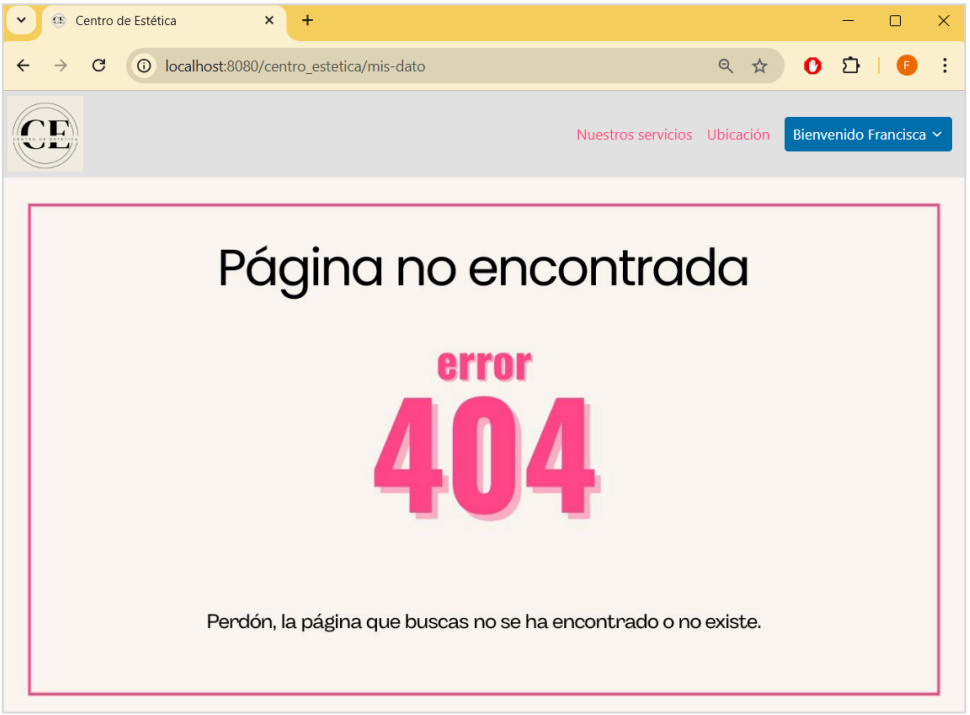
Esperamos verte pronto.

Saludos cordiales,

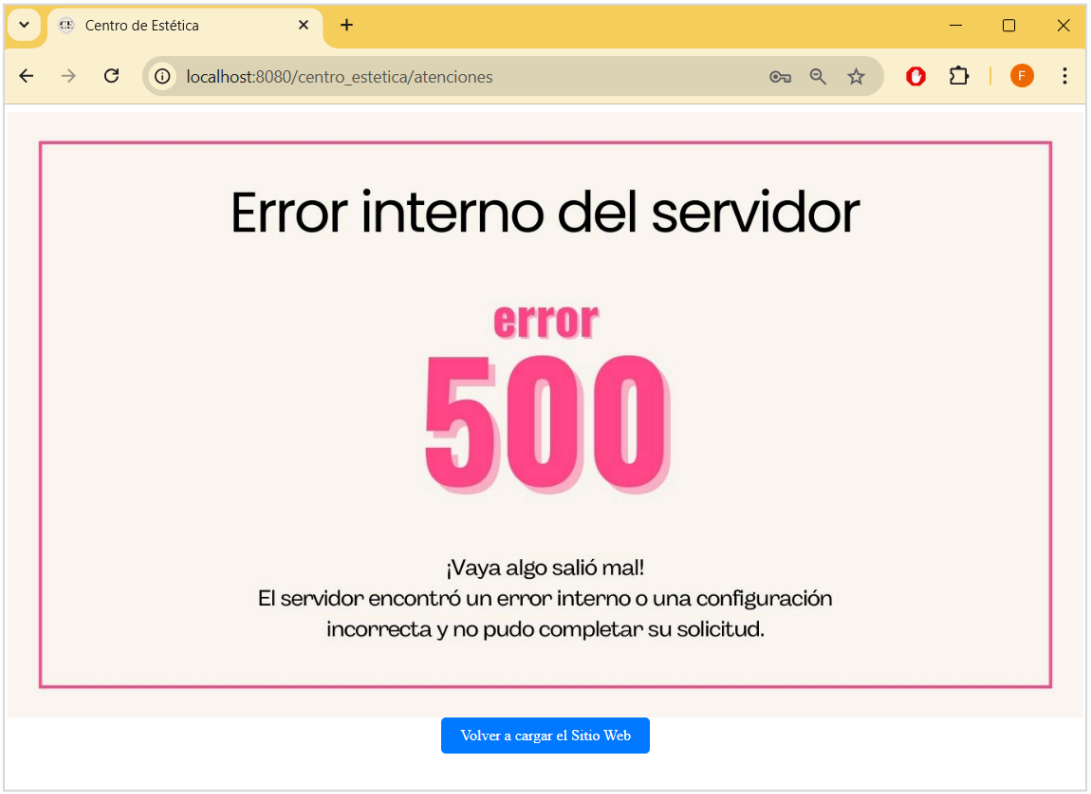
Centro de Estética
Tel: 0341 272-4611

Pantalla Mail - Cancelación de Reserva

Pantalla de error 404



Pantalla de error 500



Fragmentos de código - CUU06 Reservar un turno

BookAppointment.java

```java
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    Person user = (Person) request.getSession().getAttribute("user");

    if (user == null || user instanceof Employee) {
        response.sendRedirect("index");
        return;
    }

    // user is Client
    AppointmentLogic appointmentLogic = new AppointmentLogic();

    try {
        AppointmentFilter filter = new AppointmentFilter();

        if (request.getParameter("date") != null && !request.getParameter("date").isEmpty())
            filter.setDate(LocalDate.parse(request.getParameter("date")));

        if (request.getParameter("start-time") != null && !request.getParameter("start-time").isEmpty())
            filter.setStartTime(LocalTime.parse(request.getParameter("start-time")));

        if (request.getParameter("end-time") != null && !request.getParameter("end-time").isEmpty())
            filter.setEndTime(LocalTime.parse(request.getParameter("end-time")));

        Map<String, LinkedList<Appointment>> appointments = appointmentLogic.listAvailable(filter);
        request.setAttribute("availableAppointments", appointments);

        LinkedList<Appointment> clientAppointments = appointmentLogic.searchByClient((Client) user);
        request.setAttribute("clientAppointments", clientAppointments);

    } catch (Exception e) {
    }

    request.getRequestDispatcher("WEB-INF/client/book-appointment.jsp").forward(request, response);
}
```

```java
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    Person user = (Person) request.getSession().getAttribute("user");

    if (user == null || user instanceof Employee) {
        response.sendRedirect("index");
        return;
    }

    // user is Client
    AppointmentLogic appointmentLogic = new AppointmentLogic();
    Appointment appointment = new Appointment();

    String action = request.getParameter("action");
    try {
        if (action.equals("book")) {
            Alert alert = new Alert("error", "No se pudo reservar el turno.");

            appointment.setId(Integer.parseInt(request.getParameter("appointment-id")));
            appointment.setClient((Client) request.getSession().getAttribute("user"));
            String imagePath = getServletContext().getRealPath("/resources/CE.jpg");

            appointment = appointmentLogic.book(appointment, imagePath);
            if (appointment != null)
                alert = new Alert("success", "Turno reservado con éxito.");

            request.setAttribute("alert", alert);

        } else if (action.equals("unbook")) {
            Alert alert = new Alert("error", "No se pudo cancelar el turno.");

            appointment.setId(Integer.parseInt(request.getParameter("appointment-id")));
            appointment.setClient((Client) request.getSession().getAttribute("user"));
            String imagePath = getServletContext().getRealPath("/resources/CE.jpg");

            appointment = appointmentLogic.unbook(appointment, imagePath);
            if (appointment != null)
                alert = new Alert("success", "Turno cancelado con éxito.");

            request.setAttribute("alert", alert);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    this.doGet(request, response);
}
```

AppointmentLogic.java

```java
public class AppointmentLogic {
    private AppointmentData appointmentData;

    public AppointmentLogic() {
        appointmentData = new AppointmentData();
    }

    public LinkedList<Appointment> list() {
        return appointmentData.list();
    }

    public LinkedList<Appointment> listPast(Employee e){
        if (e.isAdmin())
            return appointmentData.listPast(null);
        else
            return appointmentData.listPast(e);
    }

    public Map<String, LinkedList<Appointment>> listAvailable(AppointmentFilter filter) {
        LinkedList<Appointment> appointments = appointmentData.listAvailable(filter);

        Map<String, LinkedList<Appointment>> groupedByDate = appointments.stream()
                .collect(Collectors.groupingBy(
                    appointment -> appointment.getFormattedDate(),
                    LinkedHashMap::new,
                    Collectors.toCollection(LinkedList::new)
                ));

        return groupedByDate;
    }

public Appointment create(Appointment a) {
    if (a.getDateTime().isBefore(LocalDateTime.now()))
        return null;

    return appointmentData.add(a);
}

public Appointment update(Appointment a, Employee e) {
    if (a.getDateTime().isBefore(LocalDateTime.now()))
        return null;

    if (e.isAdmin())
        return appointmentData.update(a, null);
    else
        return appointmentData.update(a, e);
}

public Appointment delete(Appointment a, Employee e) {
    if (e.isAdmin())
        return appointmentData.delete(a, null);
    else
        return appointmentData.delete(a, e);
}

public LinkedList<Appointment> searchByClient(Client c) {
    return appointmentData.searchByClient(c);
}

public Appointment calculateTotalIncome(Appointment a) {
    return appointmentData.calculateTotalIncome(a);
}
```

```java
public Appointment book(Appointment a, String imagePath) {
    final Appointment appointment = appointmentData.book(a);

    if (appointment == null)
        return null;

    CompletableFuture.runAsync(() -> {
        try {
            String to = appointment.getClient().getEmail();
            String subject = "Turno reservado - Centro de Estética";

            EmailSender emailSender = new EmailSender(to, subject);

            String htmlContent = "<h2>Estimado/a " + a.getClient().getFullname() +  ",</h2>"
                    + "<p>Este es un correo de confirmación para informarte que tu turno ha sido reservado correctamente.</p>"
                    + "<h3>Detalles de tu cita:</h3>"
                    + "<ul>"
                    + "<li><strong>Fecha y hora del turno:</strong> " + a.getFullFormattedDateTime() + "</li>"
                    + "<li><strong>Profesional a cargo:</strong> " + a.getEmployee().getFullname() + "</li>"
                    + "</ul>"
                    + "<p>Si tienes alguna pregunta o necesitas realizar algún cambio en tu cita, no dudes en ponerte en contacto con nosotros.</p>"
                    + "<p>Esperamos verte pronto.</p>"
                    + "<br>"
                    + "<p><strong>Saludos cordiales,</strong></p>"
                    + "<p>Centro de Estética<br>"
                    + "Tel: 0341 272-4611<br>"
                    + "<img src='cid:logo' style='width:75px; height:75px; top-border:10px;'>";

            String text = "Estimado/a " + a.getClient().getFullname() + ",\n\n"
                    + "Este es un correo de confirmación para informarte que tu turno ha sido reservado correctamente.\n\n"
                    + "Detalles de tu cita:\n"
                    + "- Fecha y hora del turno: " + a.getFullFormattedDateTime() + "\n"
                    + "- Profesional a cargo: " + a.getEmployee().getFullname() + "\n\n"
                    + "Si tienes alguna pregunta o necesitas realizar algún cambio en tu cita, no dudes en ponerte en contacto con nosotros.\n\n"
                    + "Esperamos verte pronto.\n\n"
                    + "Saludos cordiales,\n"
                    + "Centro de Estética\n"
                    + "Tel: 0341 272-4611\n";

            emailSender.send(htmlContent, text, imagePath);
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }).exceptionally(ex -> {
        System.out.println("Failed to send email: " + ex.getMessage());
        return null;
    });

    return a;
}
```

AppointmentData.java

```java
public Appointment book(Appointment appointment)
{
    DbConnector db = new DbConnector();
    Connection cn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        cn = db.getConnection();
        cn.setAutoCommit(false);

        pstmt = cn.prepareStatement(""
                + " UPDATE appointments"
                + " SET client_id = ?"
                + " WHERE id = ?"
                + "     AND client_id IS NULL"
                + "     AND date_time > NOW();");

        pstmt.setInt(1, appointment.getClient().getId());
        pstmt.setInt(2, appointment.getId());

        if (pstmt.executeUpdate() == 0) {
            cn.rollback();
            return null;
        }

        pstmt = cn.prepareStatement(""
                + " SELECT cli.firstname, cli.lastname, cli.email"
                + "     , emp.firstname, emp.lastname"
                + "     , app.date_time"
                + " FROM appointments app"
                + " INNER JOIN clients cli"
                + "     ON cli.id = app.client_id"
                + " INNER JOIN employees emp"
                + "     ON emp.id = app.employee_id"
                + " WHERE app.id = ?;");

        pstmt.setInt(1, appointment.getId());
        rs = pstmt.executeQuery();

        if (rs.next()) {
            Client c = new Client();
            c.setFirstname(rs.getString(1));
            c.setLastname(rs.getString(2));
            c.setEmail(rs.getString(3));
            appointment.setClient(c);

            Employee e = new Employee();
            e.setFirstname(rs.getString(4));
            e.setLastname(rs.getString(5));
            appointment.setEmployee(e);

            appointment.setDateTime(rs.getObject(6, LocalDateTime.class));

            cn.commit();
            return appointment;
        }
```

```java
public Appointment unbook(Appointment appointment) {
    DbConnector db = new DbConnector();
    Connection cn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        cn = db.getConnection();
        pstmt = cn.prepareStatement(""
                + " UPDATE appointments"
                + " SET client_id = NULL"
                + " WHERE id = ?"
                + "     AND client_id = ?"
                + "     AND date_time > NOW();");

        pstmt.setInt(1, appointment.getId());
        pstmt.setInt(2, appointment.getClient().getId());

        if (pstmt.executeUpdate() == 0)
            return null;

        pstmt = cn.prepareStatement(""
                + " SELECT date_time"
                + " FROM appointments"
                + " WHERE id = ?;");

        pstmt.setInt(1, appointment.getId());
        rs = pstmt.executeQuery();

        if (rs.next())
            appointment.setDateTime(rs.getObject(1, LocalDateTime.class));

        return appointment;

    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
    finally {
        try {
            if (rs != null)
                rs.close();
            if (pstmt != null)
                pstmt.close();
            db.releaseConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```java
public Appointment add(Appointment appointment) {
    DbConnector db = new DbConnector();
    Connection cn;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        cn = db.getConnection();
        pstmt = cn.prepareStatement("INSERT INTO appointments(date_time, employee_id, client_id) VALUES (?,?,?)", Statement.RETURN_GENERATED_KEYS);
        pstmt.setObject(1, appointment.getDateTime());
        pstmt.setInt(2, appointment.getEmployee().getId());

        if (appointment.getClient().getId() == 0)
            pstmt.setNull(3, java.sql.Types.BIGINT);
        else
            pstmt.setInt(3, appointment.getClient().getId());

        pstmt.executeUpdate();
        rs = pstmt.getGeneratedKeys();

        if (rs != null && rs.next())
        {
            appointment.setId(rs.getInt(1));
            return appointment;
        }
        return null;

    } catch (SQLException e) {
        System.out.println(e.getMessage());
        return null;
    }
    finally {
        try {
            if (rs != null)
                rs.close();
            if (pstmt != null)
                pstmt.close();
            db.releaseConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```java
            cn.rollback();
            return null;

        } catch (SQLException e) {
            e.printStackTrace();
            try {
                cn.rollback();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }

            return null;
        }
        finally {
            try {
                if (cn != null)
                    cn.setAutoCommit(true);
                if (rs != null)
                    rs.close();
                if (pstmt != null)
                    pstmt.close();
                db.releaseConnection();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

```java
public LinkedList<Appointment> listAvailable(AppointmentFilter appointFilter)
{
    StringBuilder filter = new StringBuilder();

    if (appointFilter.getDate() != null)
        filter.append("AND DATE(date_time) = ? ");

    if (appointFilter.getStartTime() != null)
        filter.append("AND TIME(date_time) >= ? ");

    if (appointFilter.getEndTime() != null)
        filter.append("AND TIME(date_time) <= ? ");

    DbConnector db = new DbConnector();
    Connection cn;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    LinkedList<Appointment> appointments = new LinkedList<>();

    try {
        cn = db.getConnection();
        pstmt = cn.prepareStatement(""
                + " SELECT app.id, app.date_time"
                + "      , emp.lastname, emp.firstname"
                + " FROM appointments app"
                + " INNER JOIN employees emp"
                + "      ON emp.id = app.employee_id"
                + " WHERE app.client_id IS NULL"
                + " AND app.date_time > NOW()"
                +   filter
                + " ORDER BY app.date_time;");

        int i = 1;
        if (appointFilter.getDate() != null)
            pstmt.setDate(i++, Date.valueOf(appointFilter.getDate()));

        if (appointFilter.getStartTime() != null)
            pstmt.setTime(i++, Time.valueOf(appointFilter.getStartTime()));

        if (appointFilter.getEndTime() != null)
            pstmt.setTime(i++, Time.valueOf(appointFilter.getEndTime()));

        rs = pstmt.executeQuery();

        while (rs.next()) {
            Appointment appointment = new Appointment();
            appointment.setId(rs.getInt(1));
            appointment.setDateTime(rs.getObject(2, LocalDateTime.class));
```

Appointment.java

```java
public class Appointment {
    private int id;
    private LocalDateTime dateTime;
    private Employee employee;
    private Client client;
    private double totalIncome;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    @JsonIgnore
    public LocalDateTime getDateTime() { return dateTime; }
    public void setDateTime(LocalDateTime dateTime) { this.dateTime = dateTime; }

    public Employee getEmployee() { return employee; }
    public void setEmployee(Employee employee) { this.employee = employee; }

    public Client getClient() { return client; }
    public void setClient(Client client) { this.client = client; }

    @JsonIgnore
    public String getTotalIncome() {
        if (this.totalIncome == 0) return "";

        Locale locale = Locale.forLanguageTag("es-AR");
        return NumberFormat.getCurrencyInstance(locale).format(this.totalIncome);
    }
    @JsonIgnore
    public double getDoubleTotalIncome() {
        return this.totalIncome;
    }
    public void setTotalIncome(double totalIncome) { this.totalIncome = totalIncome; }
```

```java
@JsonIgnore
public Boolean isModifiable(Employee emp) {
    if (emp.isAdmin() || emp.getId() == this.getEmployee().getId())
        return true;

    return false;
}

@JsonIgnore
public String getFormattedDateTime() {
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm");
    return this.getDateTime().format(formatter);
}

@JsonIgnore
public String getFormattedDate() {
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("EEEE dd 'de' MMMM 'de' yyyy");
    return this.getDateTime().format(formatter);
}

@JsonIgnore
public String getFullFormattedDateTime() {
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("EEEE dd 'de' MMMM 'de' yyyy - HH:mm 'hs'");
    return this.getDateTime().format(formatter);
}

@JsonIgnore
public String getFormattedTime() {
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm 'hs'");
    return this.getDateTime().format(formatter);
}
```

book-appointment.jsp

```jsp
<%= generateHead(true, null, "<link rel='stylesheet' type='text/css' href='styles/book-appointment.css'>") %>
<%
    LocalDateTime currentDateTime = LocalDateTime.now();
    @SuppressWarnings("unchecked")
    Map<String, LinkedList<Appointment>> appointments = (Map<String, LinkedList<Appointment>>) request.getAttribute("availableAppointments");
    @SuppressWarnings("unchecked")
    LinkedList<Appointment> clientAppointments = (LinkedList<Appointment>) request.getAttribute("clientAppointments");
%>

<body>
    <jsp:include page="../common/topbar.jsp"/>

    <div class="container-fluid main-container">
        <jsp:include page="../common/sidebar.jsp"/>
        <div class="content-div">
            <div id="my-appointments-div">
                <h3 style="text-align:center;">Mis Turnos</h3>
<%
if (clientAppointments.isEmpty()) {
    out.println("<p>No tiene ningún turno reservado.</p>");
}
else {
%>
                <div class="table-div">
                    <ul>
<%
    for (Appointment a : clientAppointments) {
                        out.println("<li><label>"+ a.getFullFormattedDateTime()+ "</label>");

        if (a.getDateTime().isAfter(currentDateTime)) {
                        out.println("<button type='button' class='secondary-btn' "
                            + "data-appointment-id='" + a.getId() + "' "
                            + "data-datetime='" + a.getFullFormattedDateTime() + "' "
                            + "onclick='openUnbookModal(this)' style='width:100%'>Cancelar</button>");

        }
                        out.println("</li>");
    }
%>
                    </ul>
                </div>
<%  } %>
            </div>
```

```
            <div id="available-div">
                <h3>
                    <span>Turnos Disponibles</span>
                    <button type="button" id="open-filters-btn"><img src="resources/icons/filter.png" alt="Mostrar filtros"></button>
                </h3>
<%
if (appointments.isEmpty()) {
    out.println("<p>No hay turnos disponibles :(</p>");
}
else {
%>
                <form action="reservar-turno" method="post" id="book-form">
                    <input type="hidden" name="action" value="book">
                    <div class="table-div">
                        <ul>
<%
    for (Map.Entry<String, LinkedList<Appointment>> entry : appointments.entrySet()) { %>
                            <li><b><%= entry.getKey() %></b></li>
<%
        for (Appointment a : entry.getValue()) {
            out.println(String.format(""
                + "<li><label>"
                + "<input type='radio' name='appointment-id' value='%s' required> %s - Profesional: %s"
                + "</label></li>",
                a.getId(), a.getFormattedTime(), a.getEmployee().getFullname()));
        }
    }
%>
                        </ul>
                    </div>
                    <button type="submit">Reservar</button>
                </form>
<%
}
%>
            </div>
        </div>
    </div>
```

```
<!-- Modals -->
<dialog id="filters-modal" class="modal">
    <article>
        <header>
            <h2>Filtrar Turnos</h2>
        </header>

        <form action="reservar-turno" method="get" id="filters-form">
            <label for="date">Fecha deseada</label>
            <input type="date" name="date" id="date" <%= request.getParameter("date") != null ? "value=" + request.getParameter("date") : "" %>>

            <label for="start-time">Hora desde</label>
            <input type="time" name="start-time" id="start-time" <%= request.getParameter("start-time") != null ? "value=" + request.getParameter("start-time") : "" %>>

            <label for="end-time">Hora hasta</label>
            <input type="time" name="end-time" id="end-time" <%= request.getParameter("end-time") != null ? "value=" + request.getParameter("end-time") : "" %>>

            <footer>
                <button type="button" class="secondary-btn" id="clear-filters-btn">Borrar filtros</button>
                <button type="submit">Buscar</button>
            </footer>
        </form>
    </article>
</dialog>
```

Información adicional

- Las acciones de alta, baja y modificación se realizan a través de modals (cuadros de diálogo), lo que evita que el usuario sea redirigido a otras páginas.
- Hay validaciones de datos (no string vacíos, validaciones del negocio, precios no negativos, fechas no anteriores a la fecha corriente) tanto en el navegador del cliente como en el servidor.
- El sitio se encuentra desarrollado para adaptarse a 3 tamaños de pantallas (responsive behavior).
- La API de Mercado Pago fue integrada para permitir pagos mediante códigos QR.
- Las confirmaciones por correo electrónico son utilizadas al momento de reservar turnos, cancelarlos y realizar consultas. Una persona que no se encuentra logueada en el sitio, puede realizar consultas a través de un formulario.
- Páginas amigables para errores 404 y 500.
- En la pantalla de ABMC Appointment (turno), al seleccionar un turno se cargan dinámicamente sus atenciones.
- El proyecto es de tipo Maven, lo que facilita la administración de las dependencias, despliegue y testeos automatizados.