



## **Materia: Java**

### **Comisión: Turno Noche**

**Año de Cursado: 2022**

**Numero de entrega: 1.00**  
**Título: Park-IT**  
**Fecha: 11-11-2022**

#### **Integrantes:**

- Ruiz Di Paolo, Juan Diego
- Carignani, Esteban Agustín
- Medina, Daniel Alexander
- Oficialdegui, Martín

#### **Profesores:**

- Meca, Adrián
- Tabacman, Ricardo

## Tabla de Contenidos

Definición y alcance del sistema .....	3
Definición .....	3
Alcance .....	3
Definiciones técnicas .....	4
Desarrollo .....	4
Arquitectura .....	4
Diagramas .....	5
Diagrama de clases .....	5
Modelo de datos .....	6
Casos de uso No ABMC .....	7
Registrar alquiler .....	7
Caso de uso .....	7
Fragmento de código .....	8
Pagar alquiler .....	9
Caso de uso .....	9
Fragmento de código .....	11
Registro de servicio .....	12
Caso de uso .....	12
Fragmento de código .....	14
Pago de Servicio .....	15
Caso de uso .....	15
Fragmento de código .....	16
Consultar Servicios Cliente .....	17
Caso de uso .....	17
Fragmento de código .....	18
Ingreso a Lista de Espera .....	19
Caso de uso .....	19
Fragmento de código .....	21

## Definición y alcance del sistema

### ***Definición***

El sistema surge de la necesidad de gestionar y controlar el alquiler de cocheras en un estacionamiento.

### ***Alcance***

Park-IT se encarga de la gestión de los alquileres de cada una de las cocheras del estacionamiento, llevando y persistiendo la información de los vehículos que la ocupan, los clientes a quienes pertenecen, la situación de este y las fechas de vigencia.

Además, permite gestionar la solicitud de servicios aplicables a un vehículo en una fecha dada. También cuenta con la posibilidad de mantener informado al cliente sobre la situación de su alquiler, y los servicios contratados.

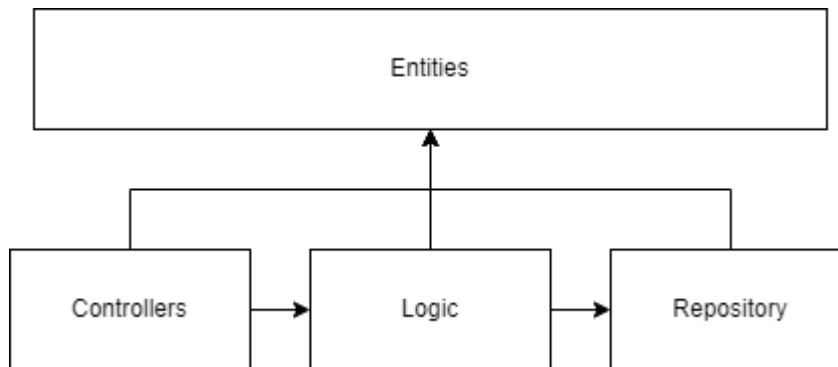
## Definiciones técnicas

### ***Desarrollo***

El sistema fue desarrollado en lenguaje Java 1.8, utilizando Eclipse como IDE, MySQL como motor de Base de Datos junto con Workbench.

### ***Arquitectura***

La arquitectura del sistema es web, con una división en capas de la siguiente manera:

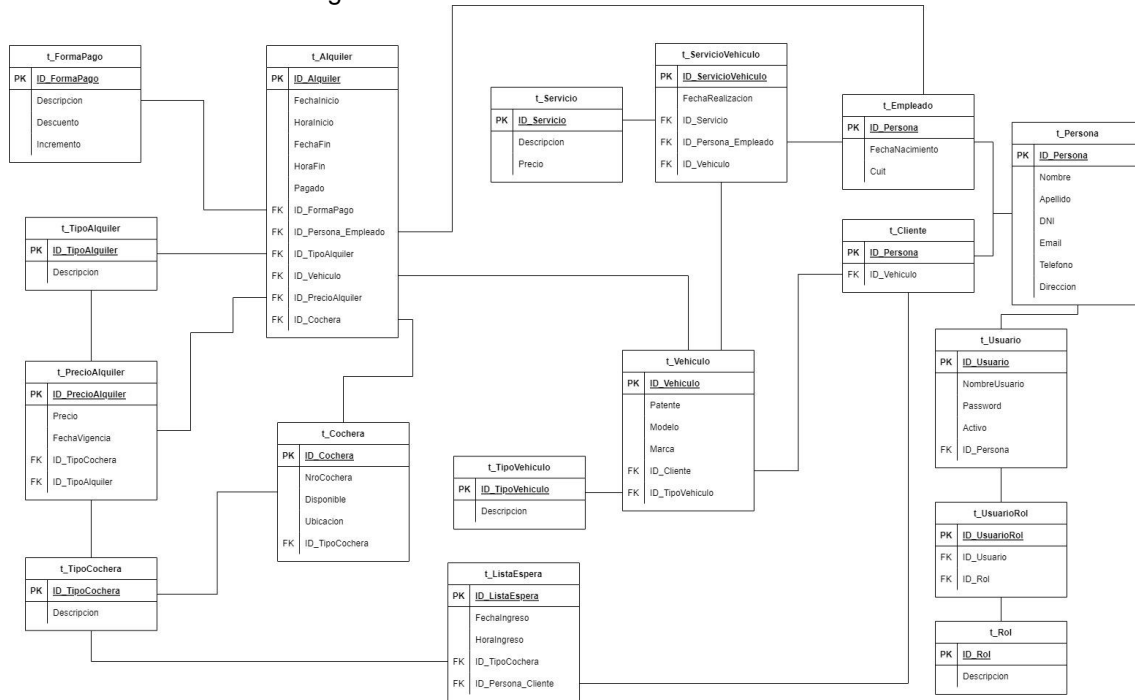


Corre en un servidor Tomcat 9 y hosteada en Heroku de forma gratuita.

## Diagramas

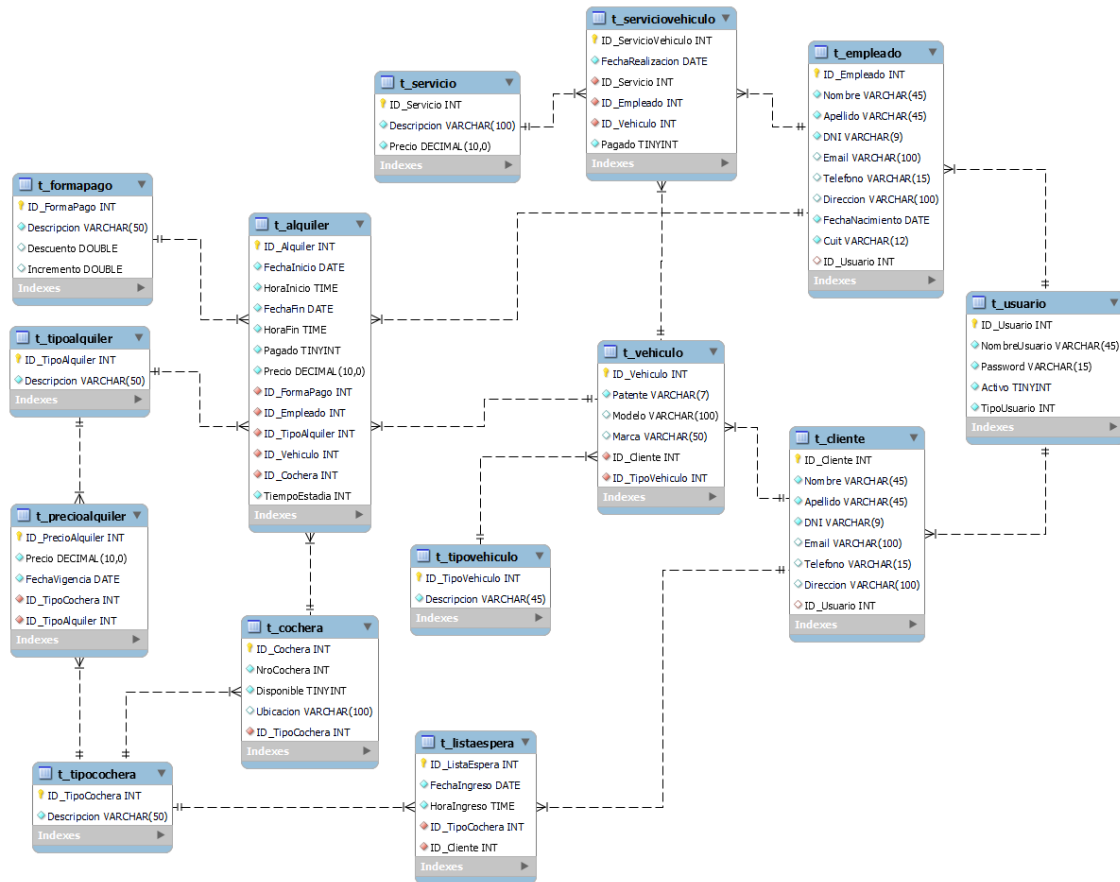
### Diagrama de clases

El modelo de datos es el siguiente:



## Modelo de datos

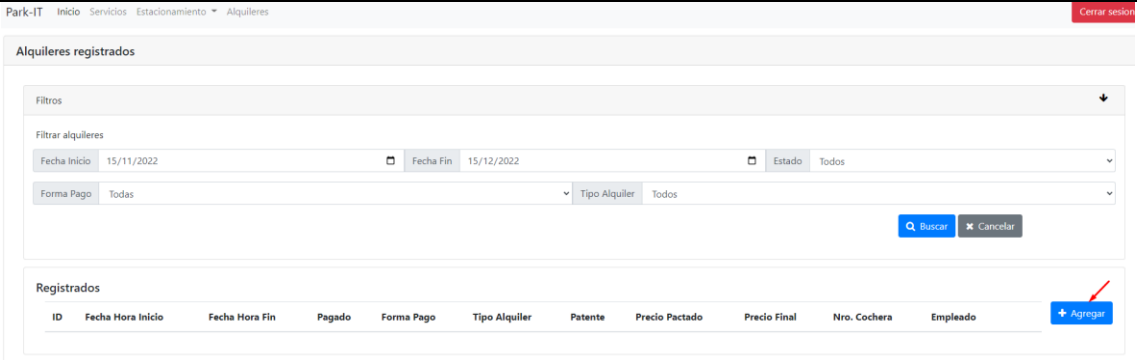
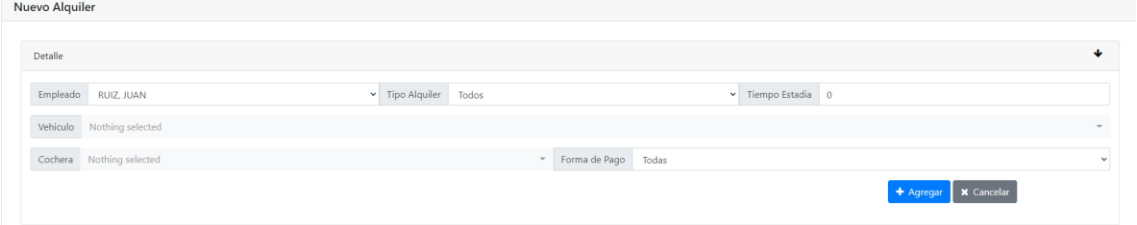
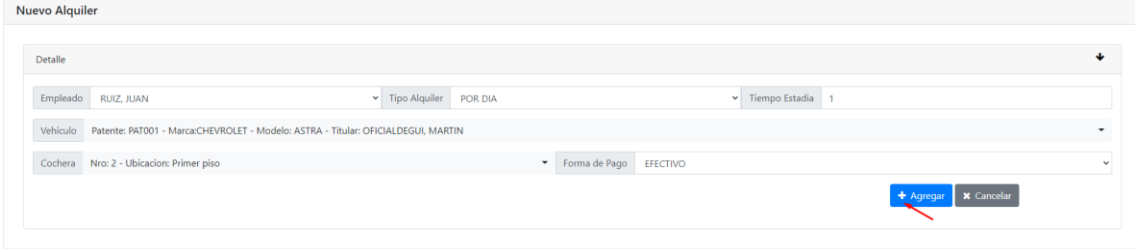
El modelo de datos lo obtenemos desde el Workbench y resulta en:



## Casos de uso No ABMC

### Registrar alquiler

#### Caso de uso

Paso	Usuario	Sistema
1	El empleado ingresa a la opción de registrar un nuevo alquiler.	Solicita la selección de Empleado, Tipo de Alquiler, Tiempo de Estadía, Vehículo (Listado de vehículos que no tengan un alquiler vigente), Cochera (listado de cocheras disponibles) y Forma de Pago.
Usu.		
Sis		
2	El empleado completa los datos solicitados.	El sistema valida que no exista un alquiler vigente para el vehículo seleccionado. Registra el alquiler para la fecha del día y calcula la fecha de fin en base a al Tipo de Alquiler y Tiempo de Estadía. Registra el alquiler.
		

Registrados											+ Agregar	
ID	Fecha Hora Inicio	Fecha Hora Fin	Pagado	Forma Pago	Tipo Alquiler	Patente	Precio Pactado	Precio Final	Nro. Cochera	Empleado		
34	15-11-2022 20:31	17-11-2022 20:31	<input type="checkbox"/>	EFFECTIVO	POR DIA	PAT001	1000.0	1700.0	2	RUIZ, JUAN		
											Pagar	

## Fragmento de código

### 1 AgregarAlquilerController

```
@WebServlet("/PagarAlquiler")
public class PagarAlquilerController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private AdministrarAlquilerLogic logic;
    private Alquiler alquiler;

    public PagarAlquilerController() {
        super();
        this.logic = AdministrarAlquilerLogic.getInstance();
        this.alquiler = new Alquiler();
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        if (this.alquiler.getID() == 0)
            this.alquiler.setID(request.getParameter("PagarID"));

        this.logic.getByID(this.alquiler);

        request.setAttribute("AlquilerSeleccionado", this.alquiler);
        request.getRequestDispatcher("PagarAlquiler.jsp").include(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Boolean esPagar = request.getParameter("Pagar") != null;
        Boolean esVolver = request.getParameter("Volver") != null;

        try {
            if (esPagar)
                this.pagar(request);
            if (esVolver)
                response.sendRedirect("AdministrarAlquiler");

            request.setAttribute("ErrorMessage", "");
        } catch (ValidationException ex) {
            request.setAttribute("ErrorMessage", ex.getMessage());
        }

        if (!esVolver)
            doGet(request, response);
    }

    private void pagar(HttpServletRequest request) throws ValidationException {
        this.logic.pagar(this.alquiler);
    }
}
```



## 2 AdministrarAlquilerLogic

```
@Override
public void add(Alquiler alquiler) throws ValidationException {
    this.validateAdd(alquiler);

    alquiler.setPrecio(PrecioAlquilerLogic.getInstancia().obtenerPrecioVigente(alquiler.getTipoAlquiler(), alquiler.getCochera().getTipoCochera()));

    this.Repository.add(alquiler);
}

@Override
protected void validateAdd(Alquiler alquiler) throws ValidationException {
    this.validarCamposRequeridos(alquiler);
    this.validarAlquilerPorVehiculo(alquiler);
}

private void validarAlquilerPorVehiculo(Alquiler alquiler) throws ValidationException {
    Alquiler alquilerExistente = this.Repository.getAlquilerPorVehiculo(alquiler.getVehiculo());

    if (alquilerExistente == null) return;

    if (alquilerExistente.getID() != alquiler.getID())
        throw new ValidationException("Ya existe un alquiler registrado para el vehiculo " + alquiler.getVehiculo().getPatente() +
            ". En la fecha " + DateFormatter.getFormattedDate(alquiler.getFechaInicio()));
}

private void validarCamposRequeridos(Alquiler alquiler) throws ValidationException {
    if (alquiler.getEmpleado() == null || alquiler.getEmpleado().getID() == 0)
        throw new ValidationException("Debe seleccionar un empleado");

    if (alquiler.getTipoAlquiler() == null || alquiler.getTipoAlquiler().getID() == 0)
        throw new ValidationException("Debe seleccionar un tipo de alquiler");

    if (alquiler.getTiempoEstadia() == 0)
        throw new ValidationException("El tiempo de estadia no puede ser 0");

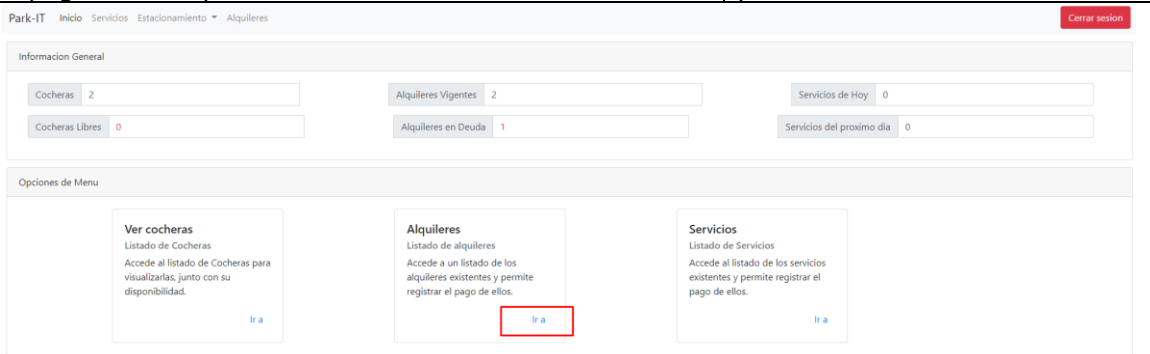
    if (alquiler.getVehiculo() == null || alquiler.getVehiculo().getID() == 0)
        throw new ValidationException("Debe seleccionar un vehiculo");

    if (alquiler.getCochera() == null || alquiler.getCochera().getID() == 0)
        throw new ValidationException("Debe seleccionar una cochera");

    if (alquiler.getFormaPago() == null || alquiler.getFormaPago().getID() == 0)
        throw new ValidationException("Debe seleccionar una forma de pago");
}
```

## Pagar alquiler

### Caso de uso

Paso	Usuario	Sistema
1	El empleado ingresa a la opción de Alquileres para registrar el pago de un Alquiler existente.	Muestra el listado de Alquileres, permitiendo filtrar.
Usu.		

Sis

Park-ITInicioServiciosEstacionamientoAlquileres

Cerrar sesión

Alquileres registrados

Filtros

Filtrar alquileres

Fecha Inicio16/11/2022Fecha Fin16/12/2022EstadoTodosForma PagoTodasTipo AlquilerTodos

BuscarCancelar

Registrados

ID	Fecha Hora Inicio	Fecha Hora Fin	Pagado	Forma Pago	Tipo Alquiler	Patente	Precio Pactado	Precio Final	Nro. Cochera	Empleado	
34	15-11-2022 23:31	17-11-2022 23:31	<input type="checkbox"/>	EFFECTIVO	POR DIA	PAT001	1000.0	1700.0	2	RUIZ, JUAN	<div><div></div><div></div><div>Pagar</div></div>

+ Agregar

2

El empleado presiona la opción “Pagar” del alquiler correspondiente.

El sistema muestra el detalle del alquiler, con el precio calculado en base al tipo de alquiler, el tiempo de estadía y la forma de pago elegidas.

Registrados

ID	Fecha Hora Inicio	Fecha Hora Fin	Pagado	Forma Pago	Tipo Alquiler	Patente	Precio Pactado	Precio Final	Nro. Cochera	Empleado	
34	15-11-2022 23:31	17-11-2022 23:31	<input type="checkbox"/>	EFFECTIVO	POR DIA	PAT001	1000.0	1700.0	2	RUIZ, JUAN	<div><div></div><div></div><div>Pagar</div></div>

+ Agregar

Detalle

Fecha de Inicio15-11-2022Hora de Inicio23:31Fecha de Fin17-11-2022Hora de Fin23:31

EmpleadoRUIZ, JUANTipo AlquilerPOR DIACocheraNro: 2 - Ubicacion: Primer piso

VehículoPatente: PAT001 - Marca:CHEVROLET - Modelo: ASTRA - Titular: OFICIALDEGUI, MARTIN

Tiempo Estadía2Forma de PagoEFFECTIVO-15.0 %Precio1000.0

Precio Final1700.0

Confirmar pago

Volver

3

El empleado confirma el pago.

Sistema registra el pago y notifica al empleado.

Usu

Detalle

Fecha de Inicio15-11-2022Hora de Inicio23:31Fecha de Fin17-11-2022Hora de Fin23:31

EmpleadoRUIZ, JUANTipo AlquilerPOR DIACocheraNro: 2 - Ubicacion: Primer piso

VehículoPatente: PAT001 - Marca:CHEVROLET - Modelo: ASTRA - Titular: OFICIALDEGUI, MARTIN

Tiempo Estadía2Forma de PagoEFFECTIVO-15.0 %Precio1000.0

Precio Final1700.0

Confirmar pago

Volver

Sis

Detalle

Fecha de Inicio15-11-2022Hora de Inicio23:31Fecha de Fin17-11-2022Hora de Fin23:31

EmpleadoRUIZ, JUANTipo AlquilerPOR DIACocheraNro: 2 - Ubicacion: Primer piso

VehículoPatente: PAT001 - Marca:CHEVROLET - Modelo: ASTRA - Titular: OFICIALDEGUI, MARTIN

Tiempo Estadía2Forma de PagoEFFECTIVO-15.0 %Precio1000.0

Precio Final1700.0

Pago registrado con éxito!

Confirmar pago

Volver

## Fragmento de código

### 3 PagarAlquilerController

```
@WebServlet("/PagarAlquiler")
public class PagarAlquilerController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private AdministrarAlquilerLogic logic;
    private Alquiler alquiler;

    public PagarAlquilerController() {
        super();
        this.logic = AdministrarAlquilerLogic.getInstance();
        this.alquiler = new Alquiler();
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        if (this.alquiler.getID() == 0)
            this.alquiler.setID(request.getParameter("PagarID"));

        this.logic.getByID(this.alquiler);

        request.setAttribute("AlquilerSeleccionado", this.alquiler);
        request.getRequestDispatcher("PagarAlquiler.jsp").include(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Boolean esPagar = request.getParameter("Pagar") != null;
        Boolean esVolver = request.getParameter("Volver") != null;

        try {
            if (esPagar)
                this.pagar(request);
            if (esVolver)
                response.sendRedirect("AdministrarAlquiler");

            request.setAttribute("ErrorMessage", "");
        } catch (ValidationException ex) {
            request.setAttribute("ErrorMessage", ex.getMessage());
        }

        if (!esVolver)
            doGet(request, response);
    }

    private void pagar(HttpServletRequest request) throws ValidationException {
        this.logic.pagar(this.alquiler);
    }
}
```

#### 4 AdministrarAlquilerLogic

```
public class AdministrarAlquilerLogic extends Logic<Alquiler, AlquilerRepository> {

    public AdministrarAlquilerLogic() {}

    private static AdministrarAlquilerLogic Instancia;

    public static AdministrarAlquilerLogic getInstancia() {}

    public void add(Alquiler alquiler) throws ValidationException {}

    @Override
    protected void validateAdd(Alquiler alquiler) throws ValidationException {
        this.validarCamposRequeridos(alquiler);
        this.validarAlquilerPorVehiculo(alquiler);
    }

    protected void validateDelete(Alquiler alquiler) throws ValidationException {}

    protected void validateUpdate(Alquiler alquiler) throws ValidationException {}

    public LinkedList<Alquiler> searchByFilter(FiltroAlquileres filtro){}

    public void pagar(Alquiler alquiler) throws ValidationException {
        if (alquiler.isPagado())
            throw new ValidationException("El alquiler ya fue pagado");

        if (alquiler.getPrecio() == 0)
            throw new ValidationException("El precio es inválido");

        this.Repository.guardarPago(alquiler);
    }

    public LinkedList<Alquiler> getAlquileresVigentes(){}

    public LinkedList<Alquiler> getAlquileresUsuario(int id){}

    public int getCantidadAlquileresVigentes() {}

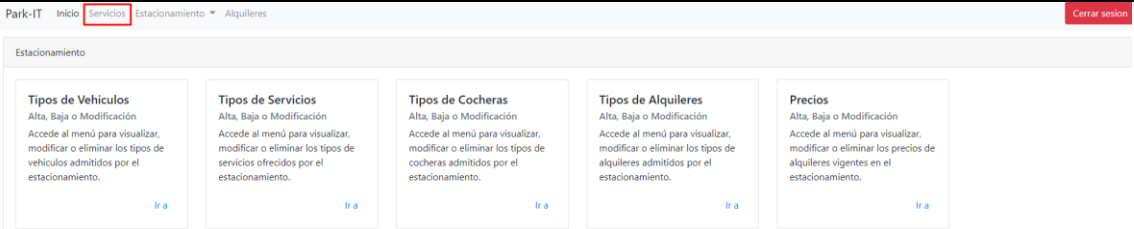
    public int getCantidadAlquileresImpagos() {}

    private void validarAlquilerPorVehiculo(Alquiler alquiler) throws ValidationException {}

    private void validarCamposRequeridos(Alquiler alquiler) throws ValidationException {}
}
```

## Registro de servicio

### Caso de uso

Paso	Usuario	Sistema
1	El empleado ingresa a la opción de Servicios para registrar un nuevo servicio deseado por un cliente.	Muestra un listado de servicios registrados, permitiendo agregar nuevos, modificar o filtrar.
Usu.		

Sis	<div>Servicios programados para Vehiculos</div> <div><div>Filtros</div><div>Detalle</div></div> <div>Registrados</div> <table><thead><tr><th>ID</th><th>Fecha Realizacion</th><th>Servicio</th><th>Empleado</th><th>Vehiculo</th><th>Precio</th><th>Pagado</th><th>Pagar</th></tr></thead><tbody><tr><td>14</td><td>2022-11-15</td><td>LAVADO DE AUTO</td><td>RUIZ, JUAN</td><td>Patente: GCP 231 - Marca:FORD - Modelo: GALAXY - Titular: MEDINA, DANIEL</td><td>\$700.0</td><td><input type="checkbox"/></td><td><div><div></div><div></div><div></div></div><div>Pagar</div></td></tr></tbody></table>		ID	Fecha Realizacion	Servicio	Empleado	Vehiculo	Precio	Pagado	Pagar	14	2022-11-15	LAVADO DE AUTO	RUIZ, JUAN	Patente: GCP 231 - Marca:FORD - Modelo: GALAXY - Titular: MEDINA, DANIEL	\$700.0	<input type="checkbox"/>	<div><div></div><div></div><div></div></div> <div>Pagar</div>
ID	Fecha Realizacion	Servicio	Empleado	Vehiculo	Precio	Pagado	Pagar											
14	2022-11-15	LAVADO DE AUTO	RUIZ, JUAN	Patente: GCP 231 - Marca:FORD - Modelo: GALAXY - Titular: MEDINA, DANIEL	\$700.0	<input type="checkbox"/>	<div><div></div><div></div><div></div></div> <div>Pagar</div>											
2	El empleado selecciona la opción de registrar un nuevo servicio para un cliente y vehículo específico.	El sistema solicita el ingreso de la Fecha en la que debe realizarse, el Servicio elegido, el Vehículo y el empleado que lo realizará.																
	<div>Detalle</div> <div>Registrados</div> <table><thead><tr><th>ID</th><th>Fecha Realizacion</th><th>Servicio</th><th>Empleado</th><th>Vehiculo</th><th>Precio</th><th>Pagado</th><th>Pagar</th></tr></thead><tbody><tr><td>14</td><td>2022-11-15</td><td>LAVADO DE AUTO</td><td>RUIZ, JUAN</td><td>Patente: GCP 231 - Marca:FORD - Modelo: GALAXY - Titular: MEDINA, DANIEL</td><td>\$700.0</td><td><input type="checkbox"/></td><td><div><div></div><div></div><div></div></div><div>Pagar</div></td></tr></tbody></table>		ID	Fecha Realizacion	Servicio	Empleado	Vehiculo	Precio	Pagado	Pagar	14	2022-11-15	LAVADO DE AUTO	RUIZ, JUAN	Patente: GCP 231 - Marca:FORD - Modelo: GALAXY - Titular: MEDINA, DANIEL	\$700.0	<input type="checkbox"/>	<div><div></div><div></div><div></div></div> <div>Pagar</div>
ID	Fecha Realizacion	Servicio	Empleado	Vehiculo	Precio	Pagado	Pagar											
14	2022-11-15	LAVADO DE AUTO	RUIZ, JUAN	Patente: GCP 231 - Marca:FORD - Modelo: GALAXY - Titular: MEDINA, DANIEL	\$700.0	<input type="checkbox"/>	<div><div></div><div></div><div></div></div> <div>Pagar</div>											
	<div>Detalle</div> <div>Servicio Vehiculo</div> <div><div>Fecha de Realizacion</div><div>dd/mm/aaaa</div><div>Servicio</div><div>LAVADO DE AUTO</div><div>Vehiculo</div><div>Patente: GCP 231 - Marca:FORD - Modelo: GALAXY - Titular: MEDINA, DANIEL</div><div>Empleado</div><div>RUIZ, JUAN</div><div><div>Guardar</div><div>Cancelar</div></div></div>																	
3	El empleado completa los datos y selecciona guardar.	Sistema valida que no exista un mismo servicio para el vehiculo seleccionado en la misma fecha. Sistema registra el servicio ingresado.																

## Fragmento de código

### 5 AdministrarServiciosVehiculosController

```
@WebServlet("/AdministrarServiciosVehiculos")
public class AdministrarServiciosVehiculosController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private ServicioVehiculoLogic Logic;
    private EmpleadoLogic EmpleadoLogicService;
    private AdministrarServicioLogic ServicioLogic;
    private VehiculoLogic VehiculoLogicService;
    private ClienteLogic ClienteLogicService;
    private ServicioVehiculo ServicioVehiculo;

    public AdministrarServiciosVehiculosController() {}

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {}

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Boolean esEliminar = request.getParameter("Eliminar") != null;
        Boolean esAgregar = request.getParameter("Guardar") != null;
        Boolean esModificar = request.getParameter("Modificar") != null;
        Boolean esBuscar = request.getParameter("Buscar") != null;
        Boolean esPagar = request.getParameter("PagarID") != null;

        try {
            if (esEliminar)
                this.eliminar(request);
            if (esAgregar)
                this.agregar(request);
            if (esModificar)
                this.modificar(request);
            if (esBuscar)
                this.buscar(request);
            if (esPagar)
                this.pagar(request);
        } catch (ValidationException ex) {
            request.setAttribute("ErrorMessage", ex.getMessage());
            Log.registrarFineLog(ex);
        }

        this.doGet(request, response);
    }

    private void agregar(HttpServletRequest request) throws ValidationException {
        this.ServicioVehiculo = new ServicioVehiculo();
        this.ServicioVehiculo.setFechaRealizacion(LocalDate.parse(request.getParameter("FechaRealizacion")));

        Empleado empleado = new Empleado();
        empleado.setID(request.getParameter("EmpleadoID"));
        this.ServicioVehiculo.setEmpleado(this.EmpleadoLogicService.getByID(empleado));

        Servicio servicio = new Servicio();
        servicio.setID(request.getParameter("ServicioID"));
        this.ServicioVehiculo.setServicio(this.ServicioLogic.getByID(servicio));

        Vehiculo vehiculo = new Vehiculo();
        vehiculo.setID(request.getParameter("VehiculoID"));
        this.ServicioVehiculo.setVehiculo(this.VehiculoLogicService.getByID(vehiculo));

        this.Logic.add(this.ServicioVehiculo);
    }
}
```

## 6 ServicioVehiculoLogic

```
public class ServicioVehiculoLogic extends Logic<ServicioVehiculo, ServicioVehiculoRepository> {

    public ServicioVehiculoLogic() {
        this.Repository = ServicioVehiculoRepository.getInstancia();
    }

    private static ServicioVehiculoLogic Instancia;

    public static ServicioVehiculoLogic getInstancia() {[]

    @Override
    protected void validateAdd(ServicioVehiculo servicioVehiculo) throws ValidationException {
        this.validateRequiredFields(servicioVehiculo);
        this.validateDate(servicioVehiculo);
        this.validateExistingService(servicioVehiculo);
    }

    protected void validateDelete(ServicioVehiculo servicioVehiculo) {}

    protected void validateUpdate(ServicioVehiculo servicioVehiculo) throws ValidationException {}

    private void validateRequiredFields(ServicioVehiculo servicioVehiculo) throws ValidationException{
        if (servicioVehiculo.getFechaRealizacion() == null || servicioVehiculo.getFechaRealizacion() == LocalDate.MIN)
            throw new ValidationException("La Fecha del Servicio es requerida");

        if (servicioVehiculo.getEmpleado() == null)
            throw new ValidationException("El Empleado es requerido");

        if (servicioVehiculo.getVehiculo() == null)
            throw new ValidationException("El Vehiculo es requerido");

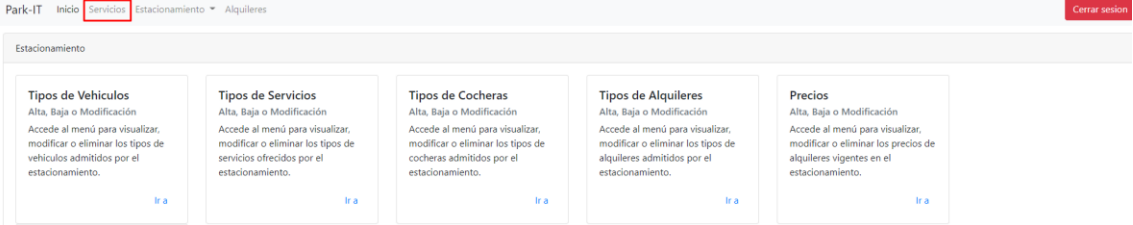
        if (servicioVehiculo.getServicio() == null)
            throw new ValidationException("El Servicio es requerido");
    }

    private void validateDate(ServicioVehiculo servicioVehiculo) throws ValidationException {
        if (servicioVehiculo.getFechaRealizacion().isBefore(LocalDate.now()))
            throw new ValidationException("No puede registrar un servicio para una fecha anterior a la actual");
    }

    private void validateExistingService(ServicioVehiculo servicioVehiculo) throws ValidationException {
        if (this.Repository.getExistingService(servicioVehiculo).getID() != 0)
            throw new ValidationException("Ya existe un mismo servicio registrado para ese vehiculo en la fecha elegida");
    }
}
```

## Pago de Servicio

### Caso de uso

Paso	Usuario	Sistema
1	El empleado ingresa a la opción de servicios para registrar el pago de un servicio registrado.	Muestra un listado de servicios registrados, permitiendo agregar nuevos, modificar o filtrar.
Usu.		

Sis

Servicios programados para Vehiculos

Filtros

Detalle

Registrados

ID	Fecha Realizacion	Servicio	Empleado	Vehiculo	Precio	Pagado	Pagar
14	2022-11-15	LAVADO DE AUTO	RUIZ, JUAN	Patente: GCP 231 - Marca:FORD - Modelo: GALAXY - Titular: MEDINA, DANIEL	\$700.0	<input type="checkbox"/>	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div>&lt;/</div></div>

## Fragmento de código

### 7 AdministrarServiciosVehiculosController

```
@WebServlet("/AdministrarServiciosVehiculos")
public class AdministrarServiciosVehiculosController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private ServicioVehiculoLogic Logic;
    private EmpleadoLogic EmpleadoLogicService;
    private AdministrarServicioLogic ServicioLogic;
    private VehiculoLogic VehiculoLogicService;
    private ClienteLogic ClienteLogicService;
    private ServicioVehiculo ServicioVehiculo;

    public AdministrarServiciosVehiculosController() {}

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {}

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Boolean esEliminar = request.getParameter("Eliminar") != null;
        Boolean esAgregar = request.getParameter("Guardar") != null;
        Boolean esModificar = request.getParameter("Modificar") != null;
        Boolean esBuscar = request.getParameter("Buscar") != null;
        Boolean esPagar = request.getParameter("PagarID") != null;

        try {
            if (esEliminar)
                this.eliminar(request);
            if (esAgregar)
                this.agregar(request);
            if (esModificar)
                this.modificar(request);
            if (esBuscar)
                this.buscar(request);
            if (esPagar)
                this.pagar(request);
        } catch (ValidationException ex) {
            request.setAttribute("ErrorMessage", ex.getMessage());
            Log.registrarFineLog(ex);
        }

        this.doGet(request, response);
    }

    private void pagar(HttpServletRequest request) {
        ServicioVehiculo servicioVehiculo = new ServicioVehiculo();
        servicioVehiculo.setID(request.getParameter("PagarID"));
        this.Logic.indicarPago(servicioVehiculo);
    }
}
```



## 8 ServiciosVehiculoLogic

```
public class ServicioVehiculoLogic extends Logic<ServicioVehiculo, ServicioVehiculoRepository> {

    public ServicioVehiculoLogic() {
        this.Repository = ServicioVehiculoRepository.getInstancia();
    }

    private static ServicioVehiculoLogic Instancia;

    public static ServicioVehiculoLogic getInstancia() {

    }

    protected void validateAdd(ServicioVehiculo servicioVehiculo) throws ValidationException {

    }

    protected void validateDelete(ServicioVehiculo servicioVehiculo) {

    }

    protected void validateUpdate(ServicioVehiculo servicioVehiculo) throws ValidationException {

    }

    private void validateRequiredFields(ServicioVehiculo servicioVehiculo) throws ValidationException {

    }

    private void validateDate(ServicioVehiculo servicioVehiculo) throws ValidationException {

    }

    private void validateExistingService(ServicioVehiculo servicioVehiculo) throws ValidationException {

    }

    public LinkedList<ServicioVehiculo> getServiciosPorFecha(LocalDate fecha){

    }

    public int getCantidadDeServiciosParaLaFecha(LocalDate fecha) {

    }

    public LinkedList<ServicioVehiculo> searchByClient(Cliente cliente, int pagadoID){

    }

    public void indicarPago(ServicioVehiculo servicioVehiculo) {
        servicioVehiculo.setPagado(true);
        this.Repository.indicarPago(servicioVehiculo);
    }

    public LinkedList<ServicioVehiculo> getServiciosUsuario(int id) {

    }

}
```

## Consultar Servicios Cliente

### Caso de uso

Paso	Usuario	Sistema
1	El cliente se loguea en el sistema para visualizar los servicios que tiene registrados.	Muestra un listado de servicios registrados para el cliente logueado.
	<b>Usu.</b> <div> <div>Iniciar sesion</div> <div>Nombre de usuario</div> <div>daniel</div> <div>Contraseña</div> <div>.....</div> <div>Ingresar</div> </div>	

Sis

Park-ITInicioServicios

Cerrar sesión

Alquileres

Patente	Nro. Cochera	Fecha Hora Inicio	Fecha Hora Fin	Tipo Alquiler	Forma Pago	Precio	Pagado
GCP 231	1	13-11-2022 22:56	14-11-2022 2:56	POR HORA	EFFECTIVO	408.0	No

Servicios

Fecha Realizacion	Servicio	Patente	Vehiculo	Precio	Pagado
15-11-2022	LAVADO DE AUTO	GCP 231	FORD - GALAXY	\$700.0	Si

## Fragmento de código

### 9 InformacionClienteController

```
@WebServlet("/InformacionCliente")
public class InformacionClienteController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private ServicioVehiculoLogic ServicioVehiculoLogic;
    private AdministrarAlquilerLogic AdministrarAlquilerLogic;
    private EmpleadoLogic EmpleadoLogicService;
    private AdministrarServicioLogic ServicioLogic;
    private VehiculoLogic VehiculoLogicService;
    private ClienteLogic ClienteLogicService;
    private ServicioVehiculo ServicioVehiculo;

    public InformacionClienteController() {}
    @SuppressWarnings("unchecked")
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        LinkedList<ServicioVehiculo> serviciosVehiculos = new LinkedList<ServicioVehiculo>();
        LinkedList<Alquiler> alquileres = new LinkedList<>();
        Usuario usuario = (Usuario) request.getSession().getAttribute("usuario");
        alquileres = this.AdministrarAlquilerLogic.getAlquileresUsuario(usuario.getID());
        request.setAttribute("ListaAlquileres", alquileres);
        serviciosVehiculos = this.ServicioVehiculoLogic.getServiciosUsuario(usuario.getID());

        request.setAttribute("ListaServiciosVehiculos", serviciosVehiculos);
        request.setAttribute("ListaVehiculos", this.VehiculoLogicService.getAll());
        request.setAttribute("ListaServicios", this.ServicioLogic.getAll());
        request.setAttribute("ListaEmpleados", this.EmpleadoLogicService.getAll());
        request.setAttribute("ListaClientes", this.ClienteLogicService.getAll());
        request.getRequestDispatcher("InformacionCliente.jsp").include(request, response);
    }
}
```

## 10 ServicioVehiculoLogic

```
public class ServicioVehiculoLogic extends Logic<ServicioVehiculo, ServicioVehiculoRepository> {

    public ServicioVehiculoLogic() {
        this.Repository = ServicioVehiculoRepository.getInstancia();
    }

    private static ServicioVehiculoLogic Instancia;

    public static ServicioVehiculoLogic getInstancia() {}

    protected void validateAdd(ServicioVehiculo servicioVehiculo) throws ValidationException {}

    protected void validateDelete(ServicioVehiculo servicioVehiculo) {}

    protected void validateUpdate(ServicioVehiculo servicioVehiculo) throws ValidationException {}

    private void validateRequiredFields(ServicioVehiculo servicioVehiculo) throws ValidationException {}

    private void validateDate(ServicioVehiculo servicioVehiculo) throws ValidationException {}

    private void validateExistingService(ServicioVehiculo servicioVehiculo) throws ValidationException {}

    public LinkedList<ServicioVehiculo> getServiciosPorFecha(LocalDate fecha){}

    public int getCantidadDeServiciosParaLaFecha(LocalDate fecha) {}

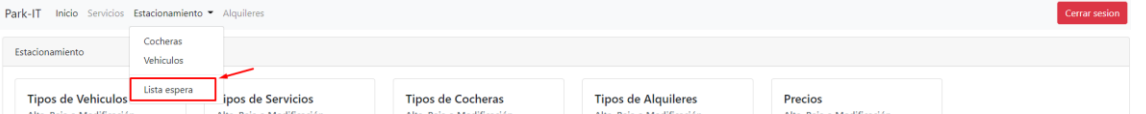

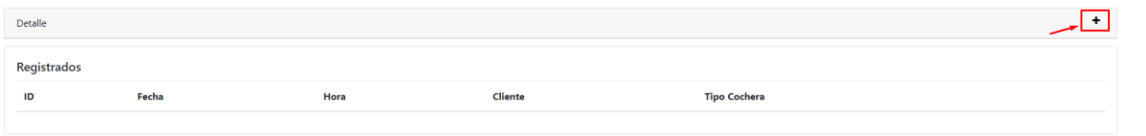
    public LinkedList<ServicioVehiculo> searchByClient(Cliente cliente, int pagadoID){}

    public void indicarPago(ServicioVehiculo servicioVehiculo) {
        servicioVehiculo.setPagado(true);
        this.Repository.indicarPago(servicioVehiculo);
    }

    public LinkedList<ServicioVehiculo> getServiciosUsuario(int id) {
        return this.Repository.getServicioUsuario(id);
    }
}
```

## Ingreso a Lista de Espera

### Caso de uso

Paso	Usuario	Sistema
1	El empleado selecciona la opción de "Lista de Espera" para registrar un nuevo cliente en ella.	Sistema muestra el Listado de Clientes en Espera. Permitiendo agregar, modificar o eliminar.
Usu.		
Sis		
2	El empleado elige la opción de agregar.	Sistema solicita Cliente y Tipo de cochera que se busca registrar.
Usu.		

	Sis	<div><div>Detalle</div><div>Nuevo registro</div><div><div>Cientes</div><div>MEDINA, DANIEL</div></div><div><div>Tipo Cochera</div><div>AUTO</div></div><div><div>Guardar</div><div>Cancelar</div></div></div>											
3		El empleado completa los datos solicitados.	Sistema valida que todos los campos esten completos y registra el Cliente y Tipo de Cochera en la Lista de Espera.										
	Usu.	<div><div>Detalle</div><div>Nuevo registro</div><div><div>Cientes</div><div>MEDINA, DANIEL</div></div><div><div>Tipo Cochera</div><div>AUTO</div></div><div><div>Guardar</div><div>Cancelar</div></div></div>											
	Sis	<div><div>Detalle</div><div>Registrados</div><table><thead><tr><th>ID</th><th>Fecha</th><th>Hora</th><th>Cliente</th><th>Tipo Cochera</th></tr></thead><tbody><tr><td>4</td><td>2022-11-16</td><td>00:30:05</td><td>MEDINA, DANIEL</td><td>AUTO</td></tr></tbody></table></div>		ID	Fecha	Hora	Cliente	Tipo Cochera	4	2022-11-16	00:30:05	MEDINA, DANIEL	AUTO
ID	Fecha	Hora	Cliente	Tipo Cochera									
4	2022-11-16	00:30:05	MEDINA, DANIEL	AUTO									

## Fragmento de código

### 11 ListaEsperaController

```
@WebServlet("/ListaEspera")
public class ListaEsperaController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private ListaEsperaLogic Logic;
    private ClienteLogic ClienteLogicService;
    private AdministrarTipoCocheraLogic TipoCocheraLogic;
    private ListaEspera ListaEspera;

    public ListaEsperaController() {}

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {}

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Boolean esEliminar = request.getParameter("Eliminar") != null;
        Boolean esAgregar = request.getParameter("Guardar") != null;
        Boolean esModificar = request.getParameter("Modificar") != null;

        try {
            if (esEliminar)
                this.eliminar(request);
            if (esAgregar)
                this.agregar(request);
            if (esModificar)
                this.modificar(request);
        } catch (ValidationException ex) {
            request.setAttribute("ErrorMessage", ex.getMessage());
            Log.registrarFineLog(ex);
        }

        this.doGet(request, response);
    }

    private void agregar(HttpServletRequest request) throws ValidationException {

        Cliente cliente= new Cliente();
        cliente.setID(request.getParameter("ClienteID"));
        this.ListaEspera.setCliente(this.ClienteLogicService.getByID(cliente));

        TipoCochera tipoCochera= new TipoCochera();
        tipoCochera.setID(request.getParameter("TipoCocheraID"));
        this.ListaEspera.setTipoCochera(this.TipoCocheraLogic.getByID(tipoCochera));

        this.Logic.add(this.ListaEspera);
    }

    private void modificar(HttpServletRequest request ) throws ValidationException {}

    private void eliminar(HttpServletRequest request) throws ValidationException {}
}
```

## 12 ListaEsperaLogic

```
public class ListaEsperaLogic extends Logic<ListaEspera, ListaEsperaRepository> {

    public ListaEsperaLogic() {
        this.Repository = ListaEsperaRepository.getInstancia();
    }

    private static ListaEsperaLogic Instancia;

    public static ListaEsperaLogic getInstancia() {
        if (Instancia == null) {
            Instancia = new ListaEsperaLogic();
        }

        return Instancia;
    }

    @Override
    protected void validateAdd(ListaEspera listaEspera) throws ValidationException {
        this.validateRequiredFields(listaEspera);

        if (listaEspera.getFechaIngreso().isBefore(LocalDate.now()))
            throw new ValidationException("La Fecha de Ingreso no puede ser anterior a la fecha actual");
    }

    protected void validateDelete(ListaEspera listaEspera) {}

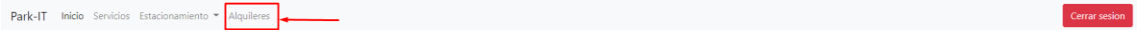
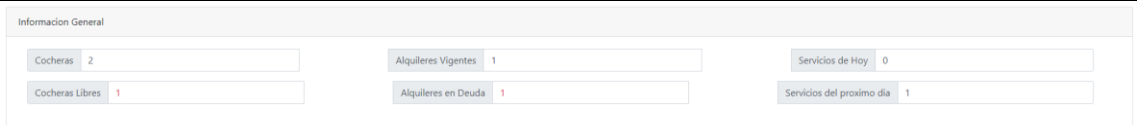
    protected void validateUpdate(ListaEspera listaEspera) throws ValidationException {}

    private void validateRequiredFields(ListaEspera listaEspera) throws ValidationException {
        if (listaEspera.getCliente() == null)
            throw new ValidationException("El Cliente es requerido");

        if (listaEspera.getTipoCochera() == null)
            throw new ValidationException("El Tipo de Cochera es requerido");
    }
}
```

## Visualizar estadísticas generales

### Caso de uso

Paso	Usuario	Sistema
1	El empleado selecciona la opción de "Alquileres" porque desea conocer el estado actual del Estacionamiento.	Sistema muestra información del Total de Cocheras vs. Cocheras Libres, Alquileres Vigentes vs. Alquileres sin pagar y Servicios de Hoy vs. Servicios de los próximos días.
Usu.		
Sis		

## Fragmento de código

### 13 AlquileresMainController

```
@WebServlet("/AlquileresMain")
public class AlquileresMainController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private HashMap<String, Accion> Acciones;
    private AdministrarAlquilerLogic Logic;
    private CocheraLogic CocheraLogicService;
    private ServicioVehiculoLogic ServicioVehiculoLogicService;

    public AlquileresMainController() {}

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setAttribute("CocherasLibres", this.CocheraLogicService.getCantidadCocherasLibres());
        request.setAttribute("Cocheras", this.CocheraLogicService.getCantidadCocheras());
        request.setAttribute("AlquileresVigentes", this.Logic.getCantidadAlquileresVigentes());
        request.setAttribute("AlquileresEnDeuda", this.Logic.getCantidadAlquileresImpagos());
        request.setAttribute("ServiciosDeHoy", this.ServicioVehiculoLogicService.getCantidadDeServiciosParaLaFecha(LocalDate.now()));
        request.setAttribute("ServicioDeManiana", this.ServicioVehiculoLogicService.getCantidadDeServiciosParaLaFecha(LocalDate.now().plusDays(1)));
        request.getRequestDispatcher("AlquileresMain.jsp").include(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {}
}
```

### 14 CocheraLogic

```
public class CocheraLogic extends Logic<Cochera, CocheraRepository> {

    private static CocheraLogic Instancia;

    public static CocheraLogic getInstancia() {
        if (Instancia == null) {
            Instancia = new CocheraLogic();
        }

        return Instancia;
    }

    public CocheraLogic() {
        this.Repository = CocheraRepository.getInstancia();
    }

    protected void validateAdd(Cochera cochera) throws ValidationException {}

    protected void validateDelete(Cochera cochera) {}

    protected void validateUpdate(Cochera cochera) throws ValidationException {}

    private void validateRequiredFields(Cochera cochera) throws ValidationException {}

    private void validateExistingNumber(Cochera cochera) throws ValidationException {}

    public LinkedList<Cochera> getCocherasLibres(){
        return this.Repository.getCocherasLibres();
    }

    public int getCantidadCocherasLibres() {
        return this.getCocherasLibres().size();
    }

    public int getCantidadCocheras() {
        return this.getAll().size();
    }
}
```

## 15 CocheraRepository

```
public LinkedList<Cochera> getCocherasLibres() {
    String query = "CALL sp_getCocherasLibres";
    PreparedStatement stmt = null;
    ResultSet rs = null;

    LinkedList<Cochera> lista = new LinkedList<Cochera>();

    try {
        stmt = DbConnector.getInstance().getConn().prepareStatement(query);
        rs = stmt.executeQuery();

        if (rs == null ) return lista;

        while (rs.next()) {
            Cochera cochera = this.getNewEntity();
            this.mapResult(rs, cochera);
            lista.add(cochera);
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally
    {
        this.closeConnection(stmt, rs);
    }

    return lista;
}
```



## 16 AdministrarAlquilerLogic

```
public class AdministrarAlquilerLogic extends Logic<Alquiler, AlquilerRepository> {  
    public AdministrarAlquilerLogic() {}  
    private static AdministrarAlquilerLogic Instancia;  
    public static AdministrarAlquilerLogic getInstancia() {}  
    public void add(Alquiler alquiler) throws ValidationException {}  
    protected void validateAdd(Alquiler alquiler) throws ValidationException {}  
    protected void validateDelete(Alquiler alquiler) throws ValidationException {}  
    protected void validateUpdate(Alquiler alquiler) throws ValidationException {}  
    public LinkedList<Alquiler> searchByFilter(FiltroAlquileres filtro){}  
    public void pagar(Alquiler alquiler) throws ValidationException {}  
    public LinkedList<Alquiler> getAlquileresVigentes(){}  
    public LinkedList<Alquiler> getAlquileresUsuario(int id){}  
    public int getCantidadAlquileresVigentes() {  
        return this.getAlquileresVigentes().size();  
    }  
    public int getCantidadAlquileresImpagos() {  
        int cantidadImpagos = 0;  
        for (Alquiler alquiler : this.getAlquileresVigentes()) {  
            if (alquiler.estaVencido())  
                cantidadImpagos++;  
        }  
        return cantidadImpagos;  
    }  
    private void validarAlquilerPorVehiculo(Alquiler alquiler) throws ValidationException {}  
    private void validarCamposRequeridos(Alquiler alquiler) throws ValidationException {}  
}
```

## 17 AlquilerRepository

```
public LinkedList<Alquiler> getAlquileresVigentes() {
    String query = String.join(" ", this.BASE_QUERY, "WHERE Pagado=?");

    PreparedStatement stmt = null;
    ResultSet rs = null;

    LinkedList<Alquiler> lista = new LinkedList<Alquiler>();
    try {
        stmt = DbConnector.getInstance().getConn().prepareStatement(query);
        stmt.setBoolean(1, false);
        rs = stmt.executeQuery();

        if (rs == null ) return lista;

        while (rs.next()) {
            Alquiler entity = this.getNewEntity();
            this.mapResult(rs, entity);
            lista.add(entity);
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally
    {
        this.closeConnection(stmt, rs);
    }

    return lista;
}
```

## 18 ServicioVehiculoLogic

```
public class ServicioVehiculoLogic extends Logic<ServicioVehiculo, ServicioVehiculoRepository> {  
  
    public ServicioVehiculoLogic() {  
        this.Repository = ServicioVehiculoRepository.getInstancia();  
    }  
  
    private static ServicioVehiculoLogic Instancia;  
  
    public static ServicioVehiculoLogic getInstancia() {  
  
    }  
  
    protected void validateAdd(ServicioVehiculo servicioVehiculo) throws ValidationException {  
  
    }  
  
    protected void validateDelete(ServicioVehiculo servicioVehiculo) {  
  
    }  
  
    protected void validateUpdate(ServicioVehiculo servicioVehiculo) throws ValidationException {  
  
    }  
  
    private void validateRequiredFields(ServicioVehiculo servicioVehiculo) throws ValidationException {  
  
    }  
  
    private void validateDate(ServicioVehiculo servicioVehiculo) throws ValidationException {  
  
    }  
  
    private void validateExistingService(ServicioVehiculo servicioVehiculo) throws ValidationException {  
  
    }  
  
    public LinkedList<ServicioVehiculo> getServiciosPorFecha(LocalDate fecha){  
  
    }  
  
    public int getCantidadDeServiciosParaLaFecha(LocalDate fecha) {  
        return this.Repository.getServiciosPorFecha(fecha).size();  
    }  
  
    public LinkedList<ServicioVehiculo> searchByClient(Cliente cliente, int pagadoID){  
  
    }  
  
    public void indicarPago(ServicioVehiculo servicioVehiculo) {  
  
    }  
  
    public LinkedList<ServicioVehiculo> getServiciosUsuario(int id) {  
  
    }  
}
```

## 19 ServicioVehiculoRepository

```
public LinkedList<ServicioVehiculo> getServiciosPorFecha(LocalDate fecha){  
    String query = String.join(" ", this.BASE_QUERY, "WHERE FechaRealizacion=?");  
  
    PreparedStatement stmt = null;  
    ResultSet rs = null;  
  
    LinkedList<ServicioVehiculo> lista = new LinkedList<>();  
    try {  
        stmt = DbConnector.getInstancia().getConn().prepareStatement(this.getQuery(query, this.PrepareBaseQuery(this.getNewEntity())));  
        stmt.setString(1, fecha.toString());  
        rs = stmt.executeQuery();  
  
        if (rs == null ) return lista;  
  
        while (rs.next()) {  
            ServicioVehiculo entity = this.getNewEntity();  
            this.mapResult(rs, entity);  
            lista.add(entity);  
        }  
    }  
    catch (SQLException ex) {  
        ex.printStackTrace();  
    }  
    finally  
    {  
        this.closeConnection(stmt, rs);  
    }  
  
    return lista;  
}
```

## Casos de Uso Re-estructurados

### ***Administración de Alquiler***

#### **Pasos:**

1. Cliente llega al Estacionamiento para registrar un Alquiler.
2. CU: *Registrar Alquiler*.
3. Empleado informa al Cliente el monto y el Cliente procede al pago de este.
4. CU: *Pagar Alquiler*.

### ***Gestión de Servicios***

#### **Pasos:**

1. Cliente llega al Estacionamiento solicitando la necesidad de un servicio
2. CU: *Registro de Servicio*.
3. El empleado informa el monto total del servicio y el cliente procede al pago de este.
4. CU: *Pagar Servicio*.

*En cualquier momento*

5. CU: *Consultar Servicios Cliente*.