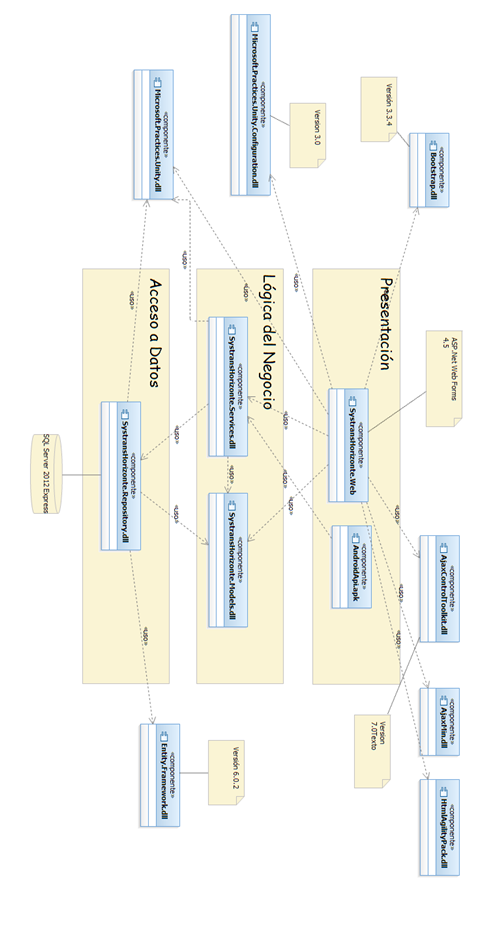
SYSTRANS HORIZONTE  
Estructura de Diseño



**Subsistema**

**Ventas**

El propósito del sistema es llevar un control de venta de pasaje y/o encomienda más rápidos, para aumentar de esta manera las ventas y acelerar los procesos en la empresa de transporte Horizonte. También es importante para disminuir la cantidad de tiempo que tomaría realizar los procesos manualmente. Un registro de pasaje y/o encomienda es más rápido realizarlo en una computadora, que realizarlo manualmente.

**Reservas**

El propósito del sistema es llevar un control de estaciones, para poder realizar un registro de reservas, modificación, eliminación y validación.

**Encomiendas**

El propósito del sistema es llevar un control de encomiendas, para poder realizar un registro de encomiendas, modificación, eliminación y validación de encomiendas.

**Usuarios**

El propósito del sistema es llevar un control de usuarios, para poder realizar un registro de usuarios, modificación, eliminación y validación de usuarios.

**Vehículos**

El propósito del sistema es llevar un control de vehículos, para poder realizar un registro de vehículos, modificación, eliminación y validación.

**Horarios**

El propósito del sistema es llevar un control de horarios, para poder realizar un registro de horarios, modificación, eliminación y validación.

**Estación**

El propósito del sistema es llevar un control de estaciones, para poder realizar un registro de estaciones, modificación, eliminación y validación.

**Asientos**

El propósito del sistema es llevar un control de estaciones, para poder realizar un registro de asientos, modificación, eliminación y validación.

**Carga**

El propósito del sistema es llevar un control de estaciones, para poder realizar un registro de cargas, modificación, eliminación y validación.

**Empleados**

El propósito del sistema es llevar un control de estaciones, para poder realizar un registro de empleados, modificación, eliminación y validación.

**Clientes**

El propósito del sistema es llevar un control de estaciones, para poder realizar un registro de estaciones, modificación, eliminación y validación.

**PATRONES**

**MVC**

**Visión de conjunto**

Este patrón plantea la separación del problema en tres capas: la capa modelo, que representa la realidad; la capa controlador, que conoce los métodos, atributos del modelo, recibe y realiza lo que el usuario quiere hacer; y la capa vista, que muestra un aspecto del modelo y es utilizada por la capa anterior para interaccionar con el usuario.

**Estructura**

La comunicación entre el modelo, el controlador y la vista es de forma referenciada.

**Comportamiento**

El controlador va a recibir toda la data de la base de datos, y a través de los modelos va poder saber el esquema de la base de datos, con esta data se van a abastecer de información los métodos del controlador si fuese necesario, para luego mostrarlo en pantalla a través de la vista.

**Relación de Requisitos**

**Registrar Venta**

**Vista de los participantes**

Venta Vista. UI del módulo venta.

Venta Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Venta Service. Interfaz con métodos de venta.

Venta Service. Métodos implementados de la Interfaz service.

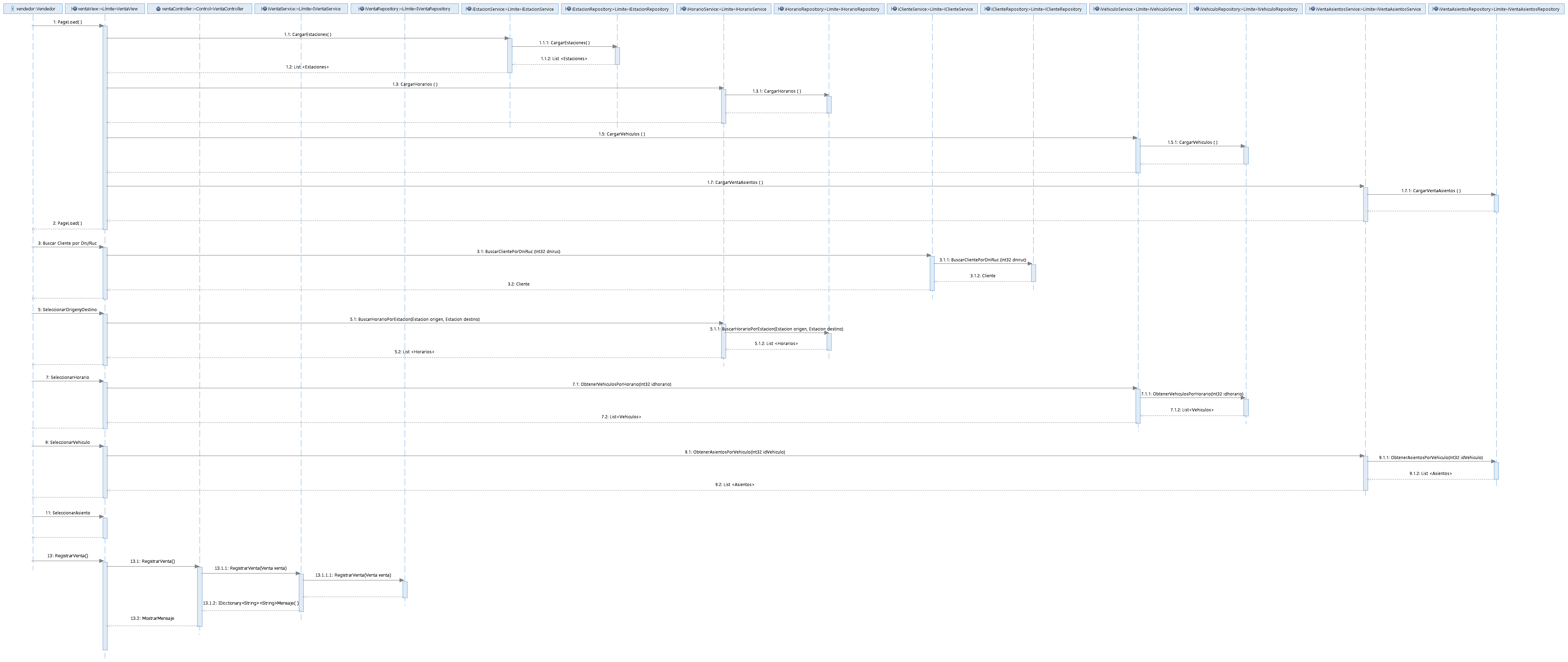
Interfaz Venta Repository. Interfaz con métodos de venta.

Venta Repository. Métodos implementados de la Interfaz Repository.

Venta Validator. Metodos para validar venta.

Entidad Venta. Clase con atributos de venta.

**Escenario Básico**



**Modificar Venta**

**Vista de los participantes**

Venta Vista. UI del módulo venta.

Venta Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Venta Service. Interfaz con métodos de venta.

Venta Service. Métodos implementados de la Interfaz service.

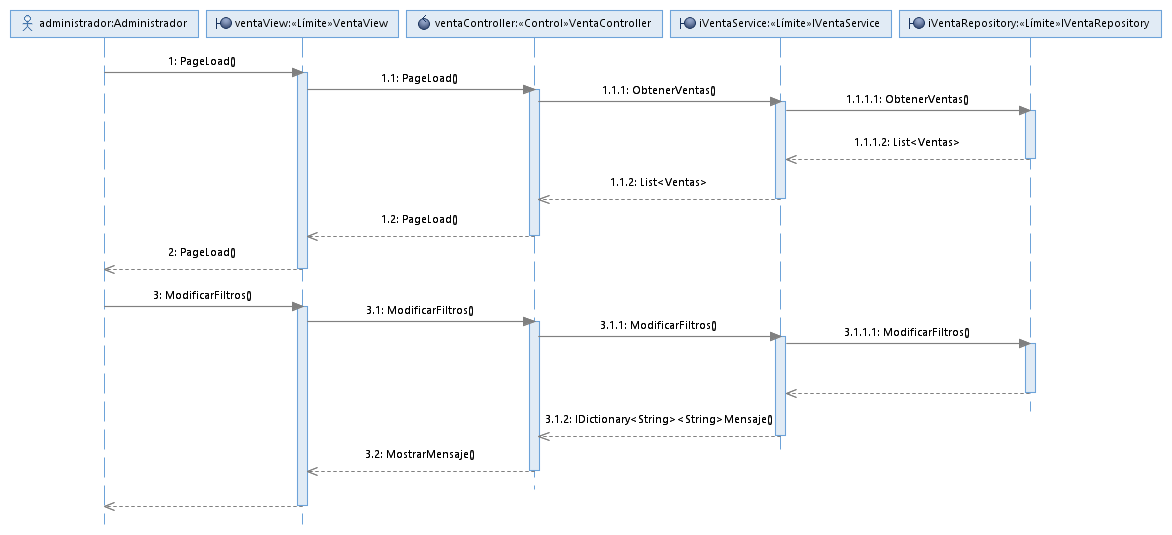
Interfaz Venta Repository. Interfaz con métodos de venta.

Venta Repository. Métodos implementados de la Interfaz Repository.

Venta Validator. Metodos para validar venta.

Entidad Venta. Clase con atributos de venta.

**Escenario Básico**



**Buscar Venta**

**Vista de los participantes**

Venta Vista. UI del módulo venta.

Venta Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Venta Service. Interfaz con métodos de venta.

Venta Service. Métodos implementados de la Interfaz service.

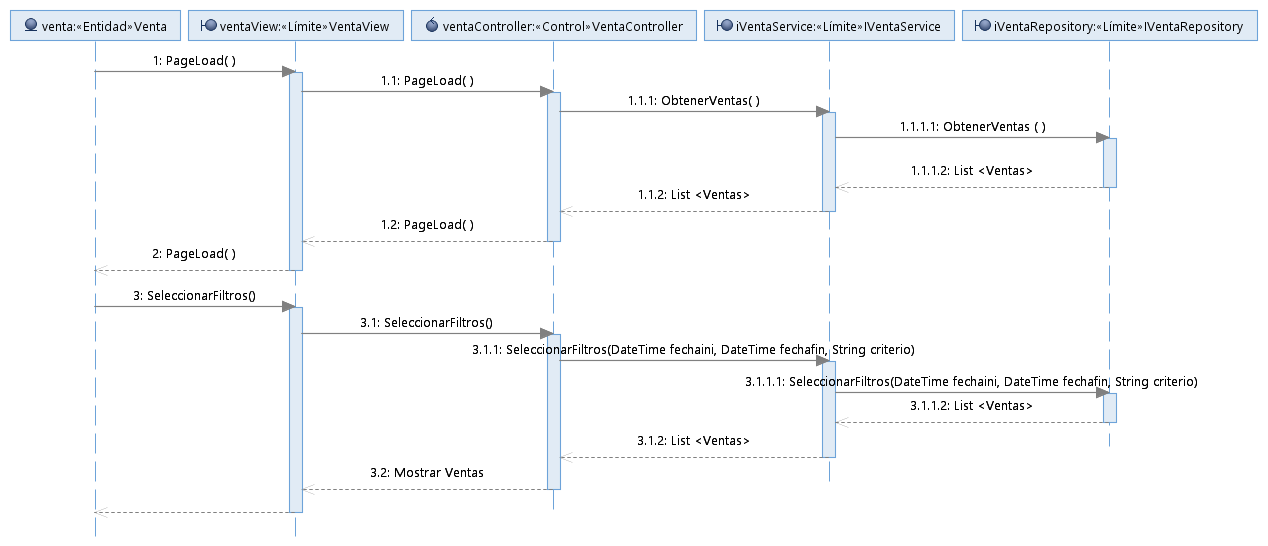
Interfaz Venta Repository. Interfaz con métodos de venta.

Venta Repository. Métodos implementados de la Interfaz Repository.

Venta Validator. Metodos para validar venta.

Entidad Venta. Clase con atributos de venta.

**Escenario Básico**



**Eliminar Venta**

**Vista de los participantes**

Venta Vista. UI del módulo venta.

Venta Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Venta Service. Interfaz con métodos de venta.

Venta Service. Métodos implementados de la Interfaz service.

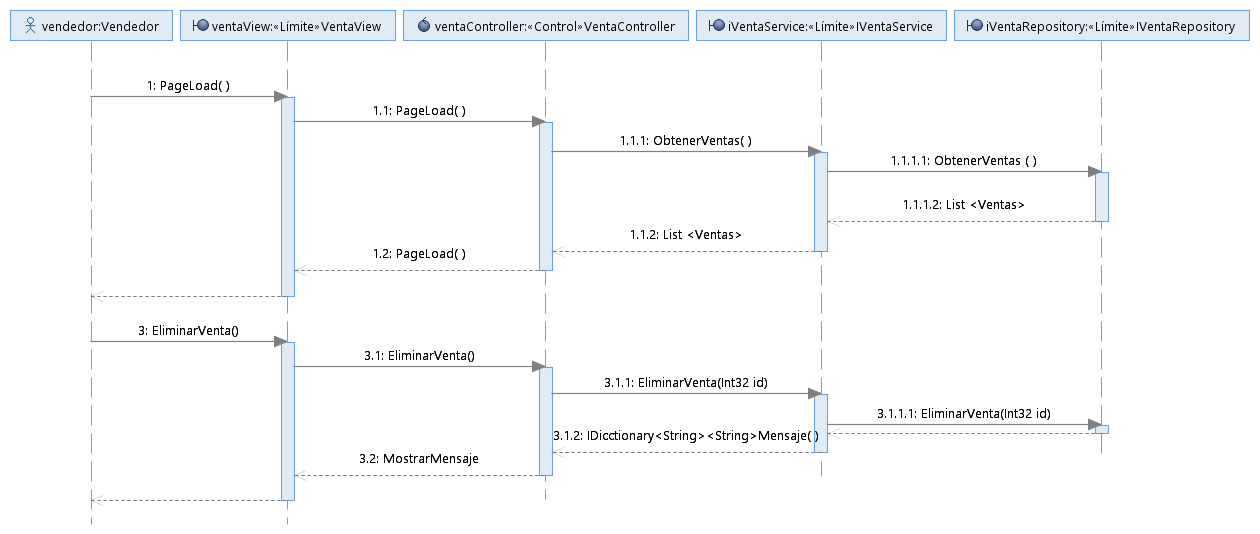
Interfaz Venta Repository. Interfaz con métodos de venta.

Venta Repository. Métodos implementados de la Interfaz Repository.

Venta Validator. Metodos para validar venta.

Entidad Venta. Clase con atributos de venta.

**Escenario Básico**



**Registrar Reserva**

**Vista de los participantes**

Reserva Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Reserva Service. Interfaz con métodos de reserva.

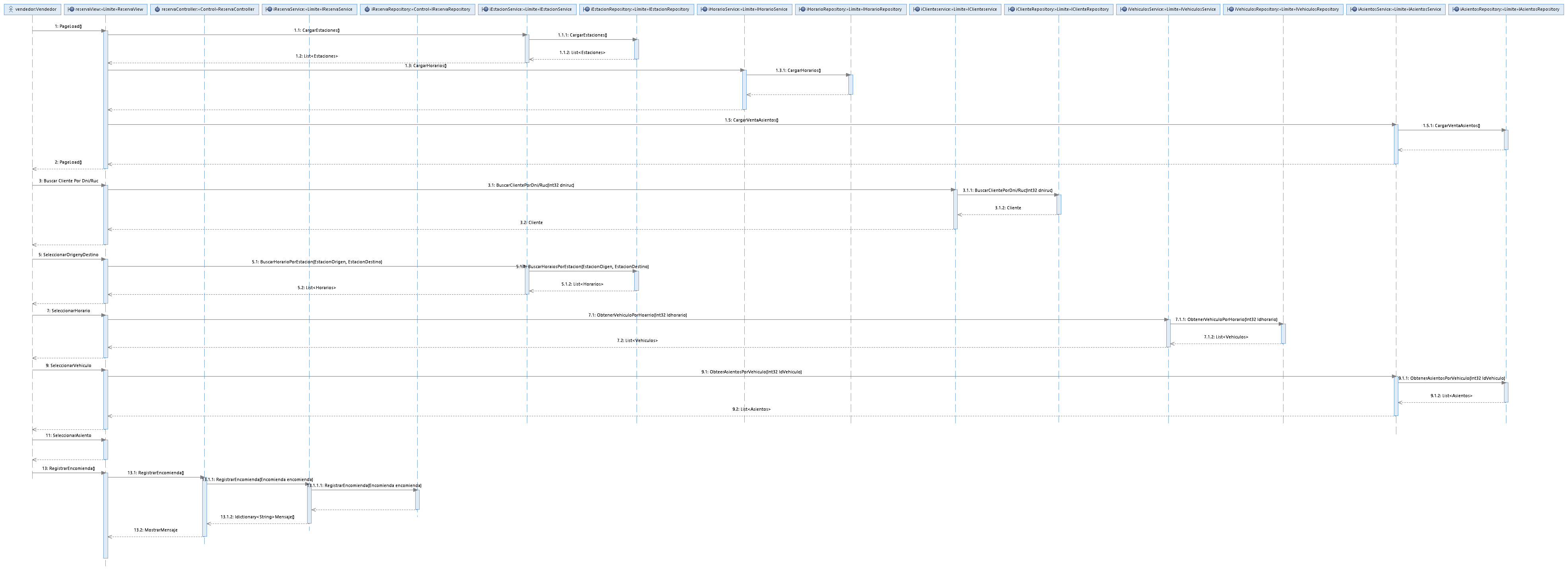
Reserva Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de reserva.

Reserva Repository. Métodos implementados de la Interfaz Repository.

Reserva Validator. Metodos para validar reserva.

**Escenario Básico**



**Buscar Reserva**

**Vista de los participantes**

Reserva Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Reserva Service. Interfaz con métodos de reserva.

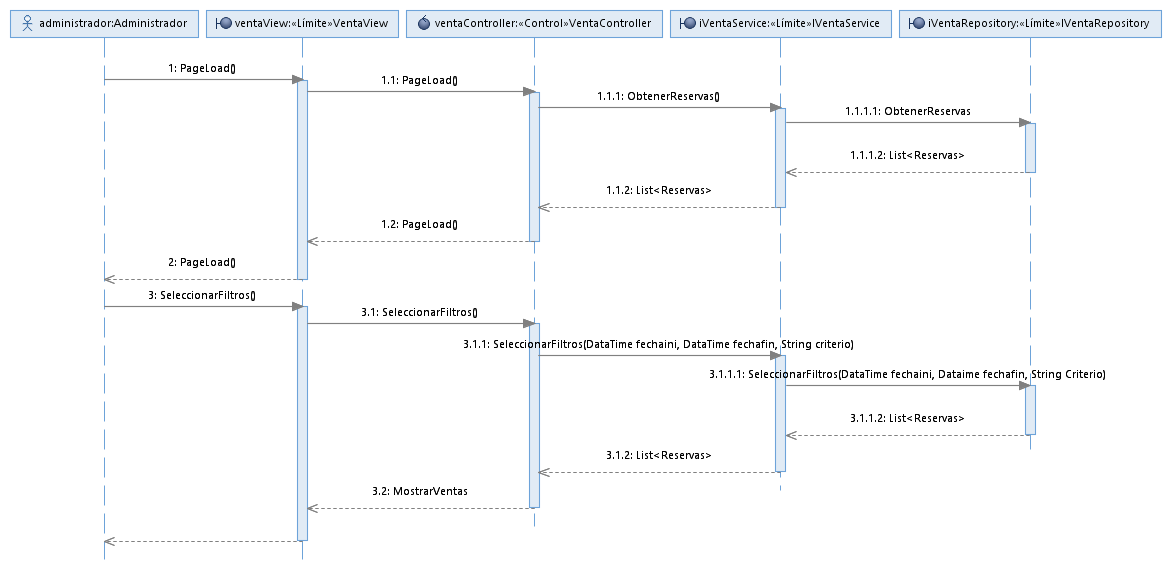
Reserva Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de reserva.

Reserva Repository. Métodos implementados de la Interfaz Repository.

Reserva Validator. Metodos para validar reserva.

**Escenario Básico**

****

**Modificar Reserva**

**Vista de los participantes**

Reserva Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Reserva Service. Interfaz con métodos de reserva.

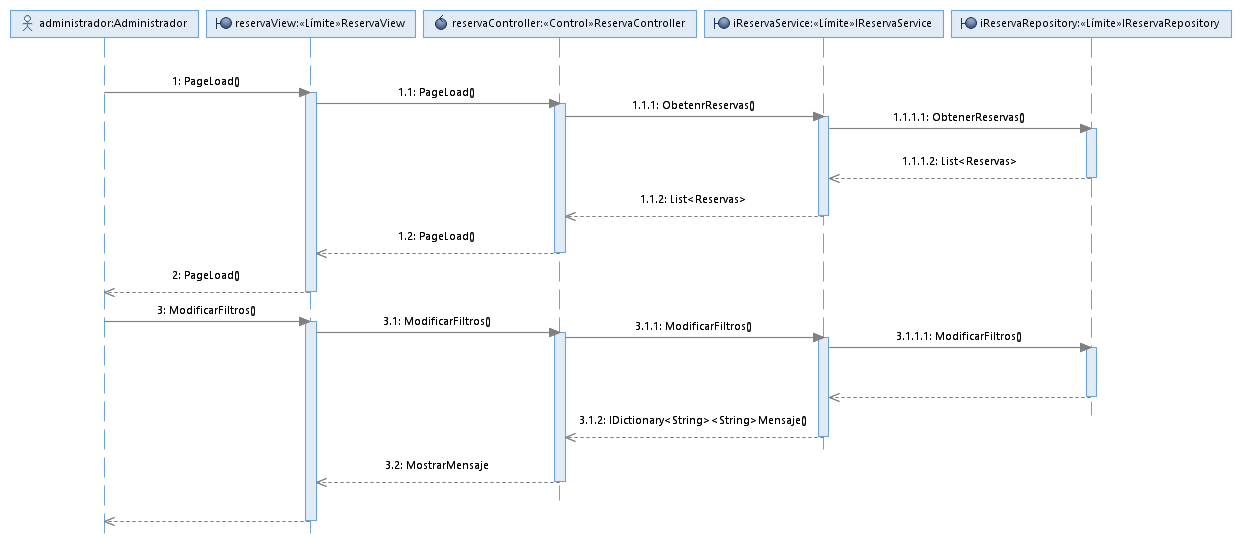
Reserva Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de reserva.

Reserva Repository. Métodos implementados de la Interfaz Repository.

Reserva Validator. Metodos para validar reserva.

**Escenario Básico**

****

**Eliminar Resera**

**Vista de los participantes**

Reserva Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Reserva Service. Interfaz con métodos de reserva.

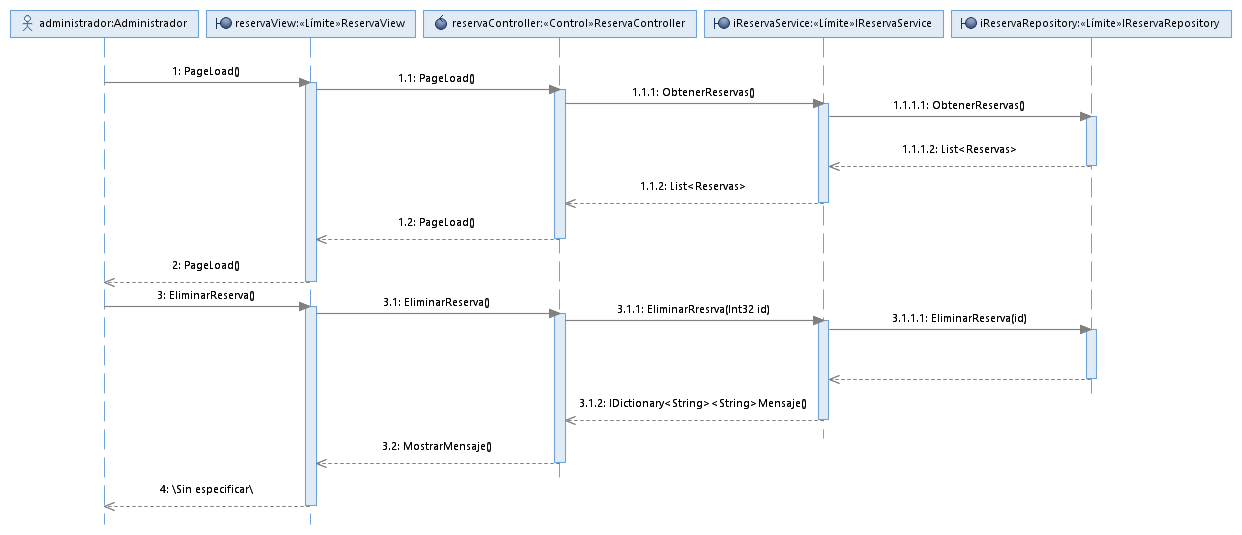
Reserva Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de reserva.

Reserva Repository. Métodos implementados de la Interfaz Repository.

Reserva Validator. Metodos para validar reserva.

**Escenario Básico**

****

**Registrar Encomienda**

**Vista de los participantes**

Encomienda Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Encomienda Service. Interfaz con métodos de Encomienda.

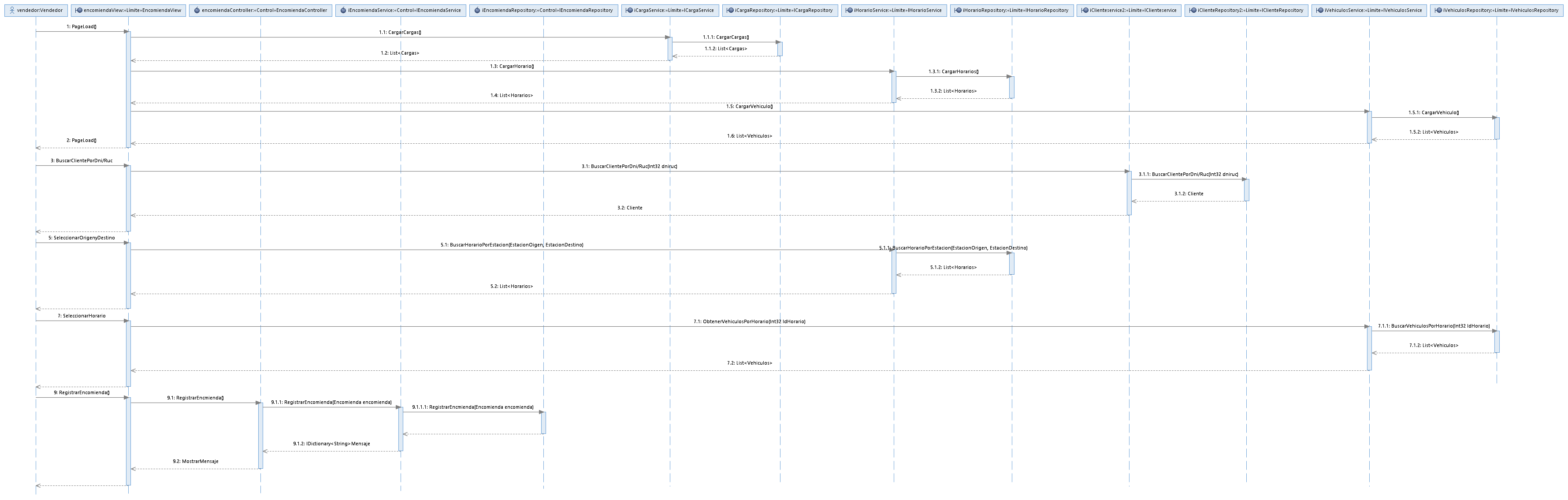
Encomienda Service. Métodos implementados de la Interfaz service.

Interfaz Encomienda Repository. Interfaz con métodos de Encomienda.

Encomienda Repository. Métodos implementados de la Interfaz Repository.

Encomienda Validator. Metodos para validar Encomienda.

**Escenario Básico**

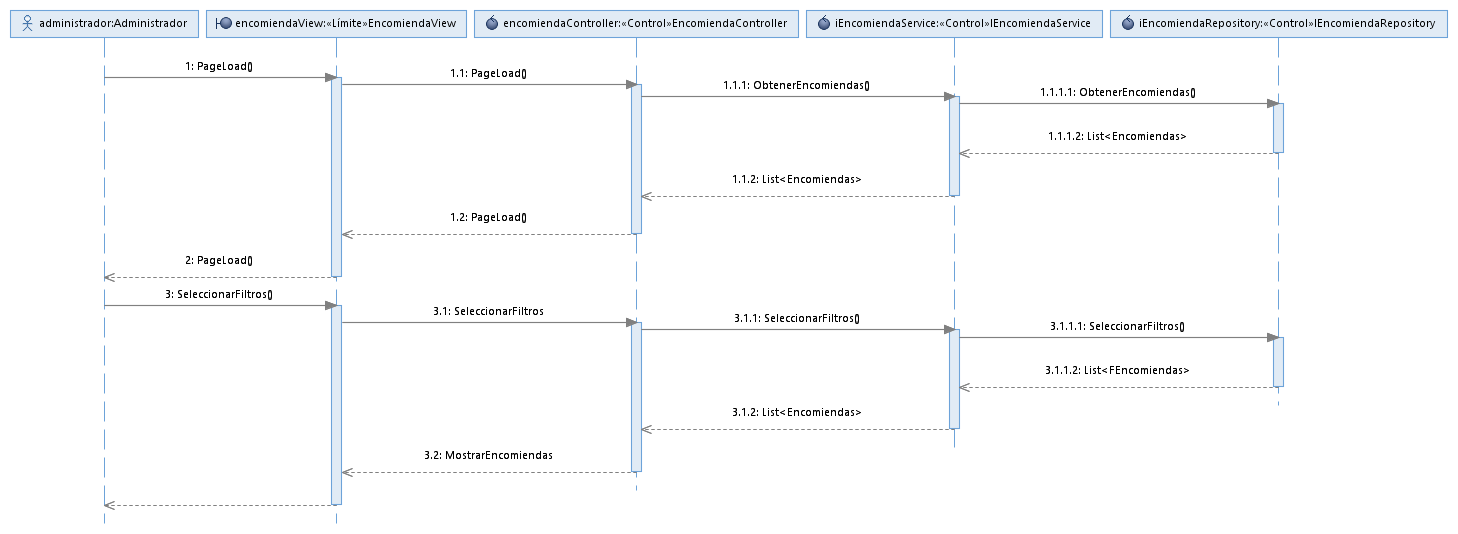
**Buscar Encomienda**

**Vista de los participantes**

Encomienda Controller. Clase que se encarga de enviar y recibir datos a la Vista.Interfaz Encomienda Service. Interfaz con métodos de Encomienda.Encomienda Service. Métodos implementados de la Interfaz service.Interfaz Encomienda Repository. Interfaz con métodos de Encomienda.

Encomienda Repository. Métodos implementados de la Interfaz Repository.Encomienda Validator. Metodos para validar Encomienda.

**Escenario Básico**

**Modificar Encomienda**

**Vista de los participantes**

Encomienda Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Encomienda Service. Interfaz con métodos de Encomienda.

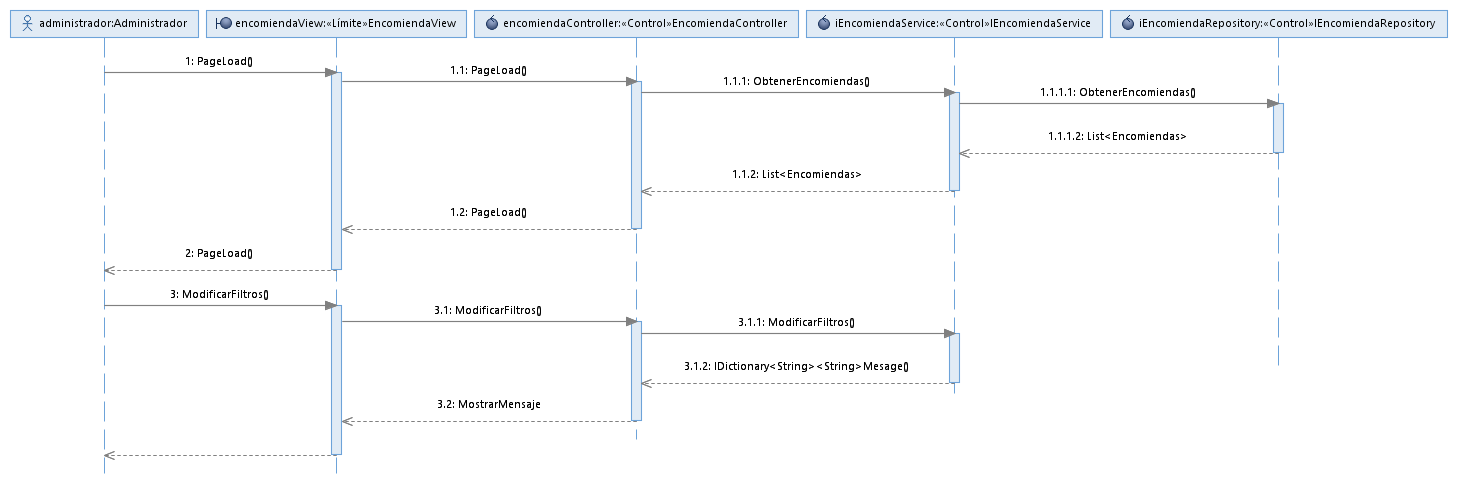
Encomienda Service. Métodos implementados de la Interfaz service.

Interfaz Encomienda Repository. Interfaz con métodos de Encomienda.

Encomienda Repository. Métodos implementados de la Interfaz Repository.

Encomienda Validator. Metodos para validar Encomienda.

**Escenario Básico**

****

**Eliminar Encomienda**

**Vista de los participantes**

Encomienda Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Encomienda Service. Interfaz con métodos de Encomienda.

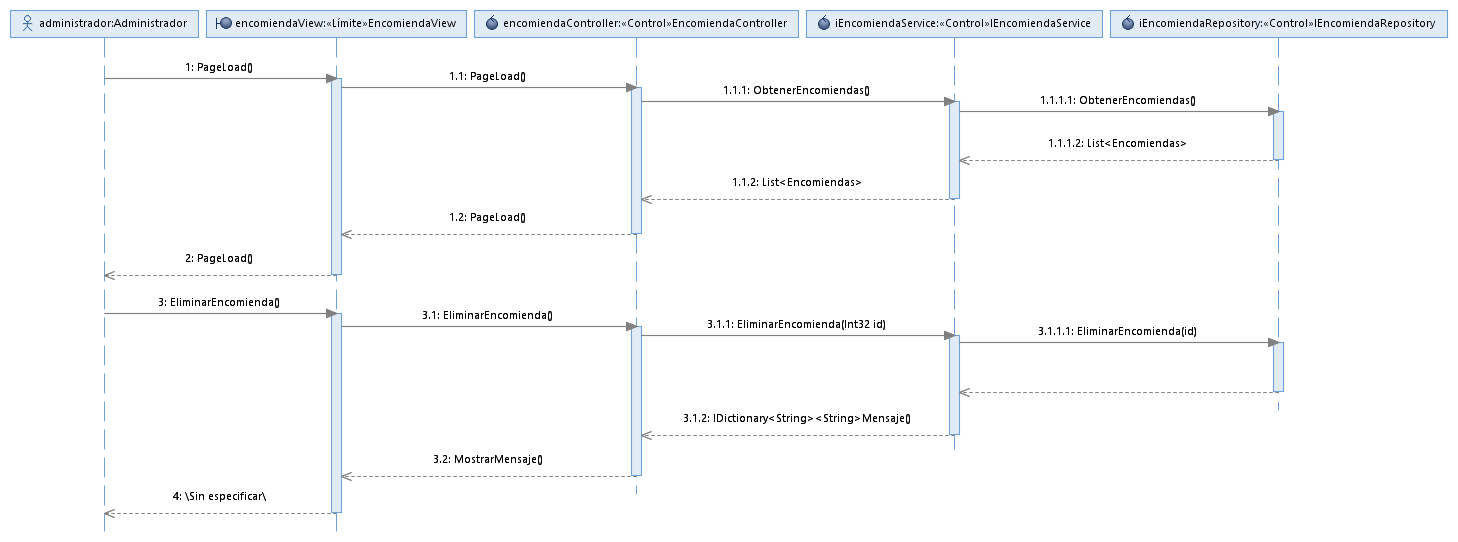
Encomienda Service. Métodos implementados de la Interfaz service.

Interfaz Encomienda Repository. Interfaz con métodos de Encomienda.

Encomienda Repository. Métodos implementados de la Interfaz Repository.

Encomienda Validator. Metodos para validar Encomienda.

**Escenario Básico**

****

**Registrar Usuario**

**Vista de los participantes**

Usuario Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Usuario Service. Interfaz con métodos de Usuario.

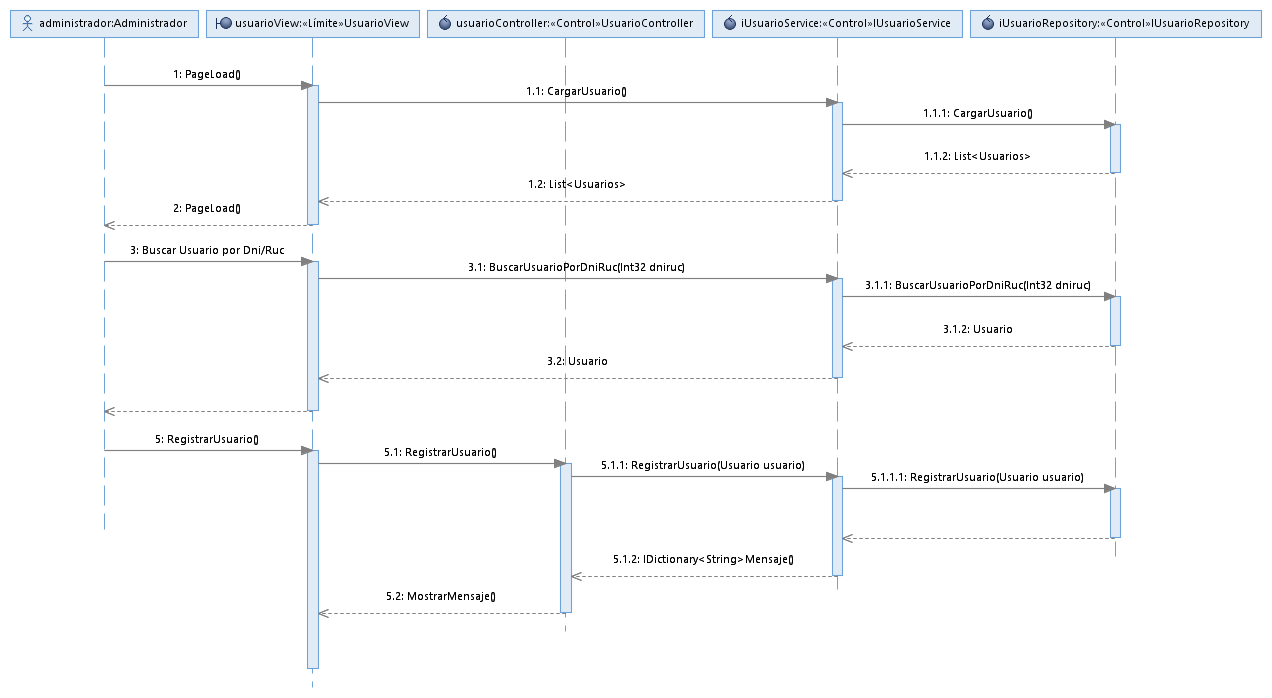
Usuario Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Usuario.

Usuario Repository. Métodos implementados de la Interfaz Repository.

Usuario Validator. Metodos para validar Usuario.

**Escenario Básico**



**Buscar Usuario**

**Vista de los participantes**

Usuario Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Usuario Service. Interfaz con métodos de Usuario.

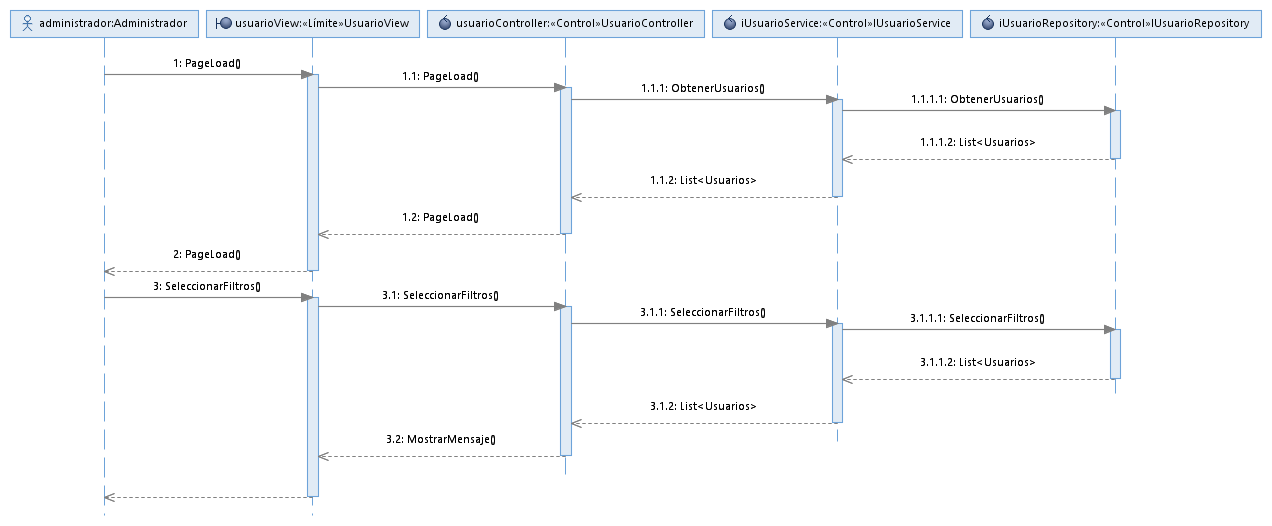
Usuario Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Usuario.

Usuario Repository. Métodos implementados de la Interfaz Repository.

Usuario Validator. Metodos para validar Usuario.

**Escenario Básico**

**Modificar Usuario**

**Vista de los participantes**

Usuario Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Usuario Service. Interfaz con métodos de Usuario.

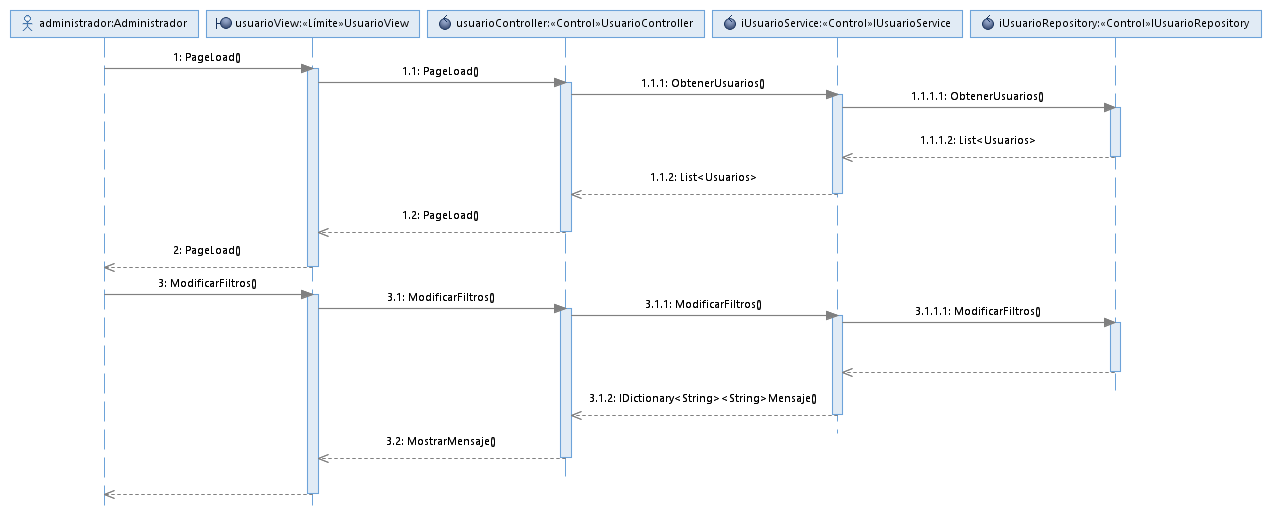
Usuario Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Usuario.

Usuario Repository. Métodos implementados de la Interfaz Repository.

Usuario Validator. Metodos para validar Usuario.

**Escenario Básico**



**Eliminar Usuario**

**Vista de los participantes**

Usuario Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Usuario Service. Interfaz con métodos de Usuario.

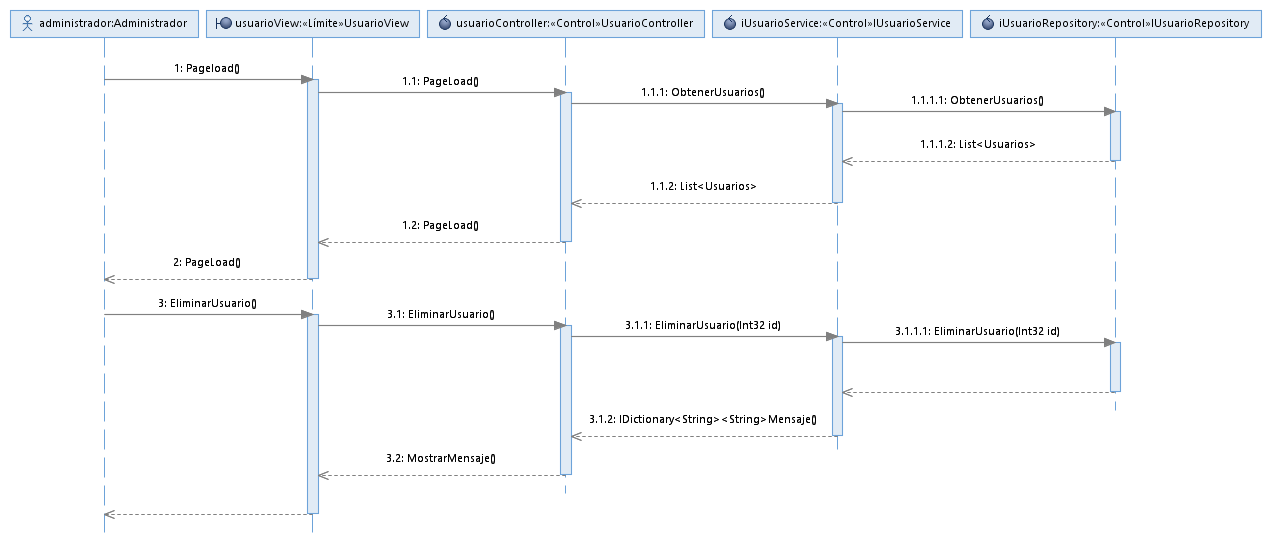
Usuario Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Usuario.

Usuario Repository. Métodos implementados de la Interfaz Repository.

Usuario Validator. Metodos para validar Usuario.

**Escenario Básico**



**Modificar Usuario**

**Vista de los participantes**

Usuario Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Usuario Service. Interfaz con métodos de Usuario.

Usuario Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Usuario.

Usuario Repository. Métodos implementados de la Interfaz Repository.

Usuario Validator. Metodos para validar Usuario.

**Escenario Básico**

**Registrar Cliente**

**Vista de los participantes**

Cliente Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Cliente Service. Interfaz con métodos de Cliente.

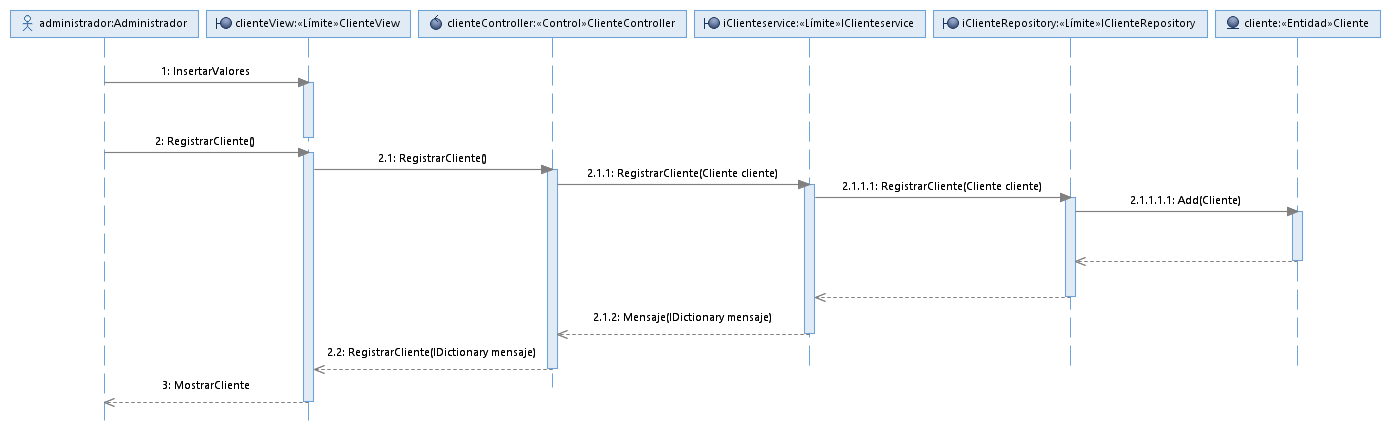
Cliente Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Cliente.

Cliente Repository. Métodos implementados de la Interfaz Repository.

Reserva Validator. Metodos para validar reserva.

**Escenario Básico**



**Registrar Carga**

**Vista de los participantes**

Carga Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Carga Service. Interfaz con métodos de Carga.

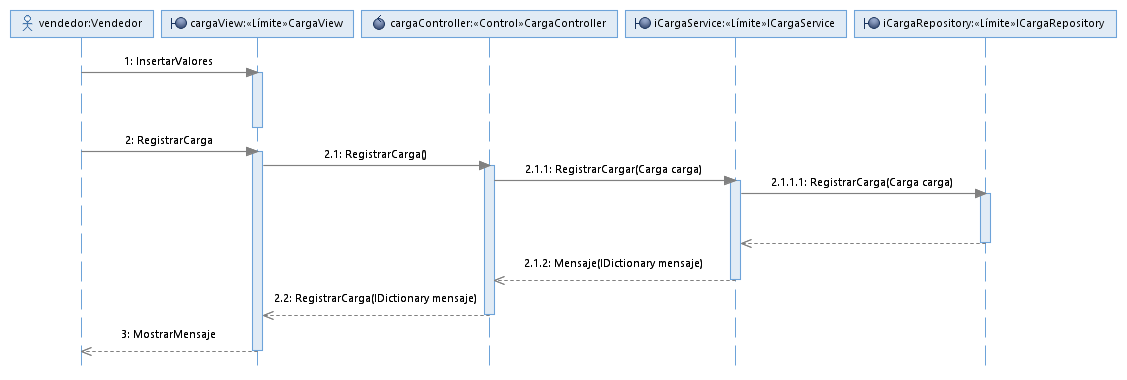
Carga Service. Métodos implementados de la Interfaz service.

Interfaz Carga Repository. Interfaz con métodos de Carga.

Carga Repository. Métodos implementados de la Interfaz Repository.

Carga Validator. Metodos para validar Carga.

**Escenario Básico**



**Buscar Carga**

**Vista de los participantes**

Carga Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Carga Service. Interfaz con métodos de Carga.

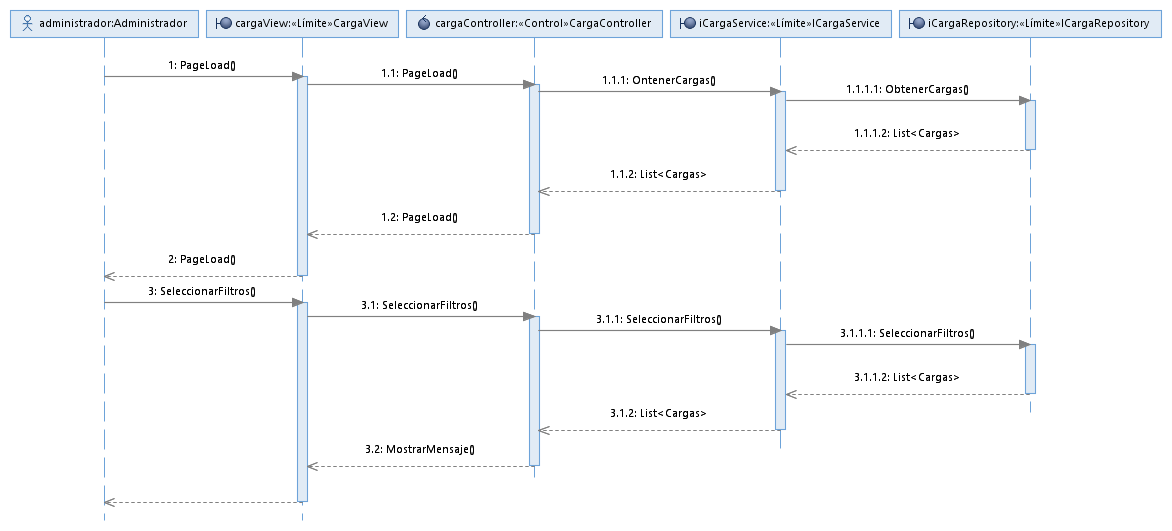
Carga Service. Métodos implementados de la Interfaz service.

Interfaz Carga Repository. Interfaz con métodos de Carga.

Carga Repository. Métodos implementados de la Interfaz Repository.

Carga Validator. Metodos para validar Carga.

**Escenario Básico**

****

**Modificar Carga**

**Vista de los participantes**

Carga Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Carga Service. Interfaz con métodos de Carga.

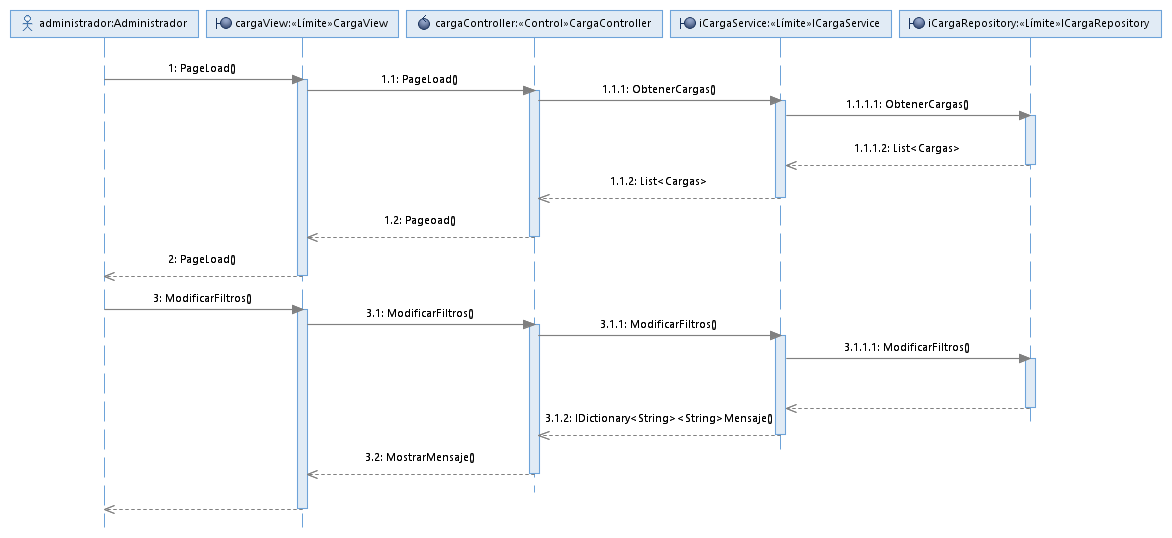
Carga Service. Métodos implementados de la Interfaz service.

Interfaz Carga Repository. Interfaz con métodos de Carga.

Carga Repository. Métodos implementados de la Interfaz Repository.

Carga Validator. Metodos para validar Carga.

**Escenario Básico**

****

**Registrar Empleados**

**Vista de los participantes**

Empleado Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Empleado Service. Interfaz con métodos de Empleado.

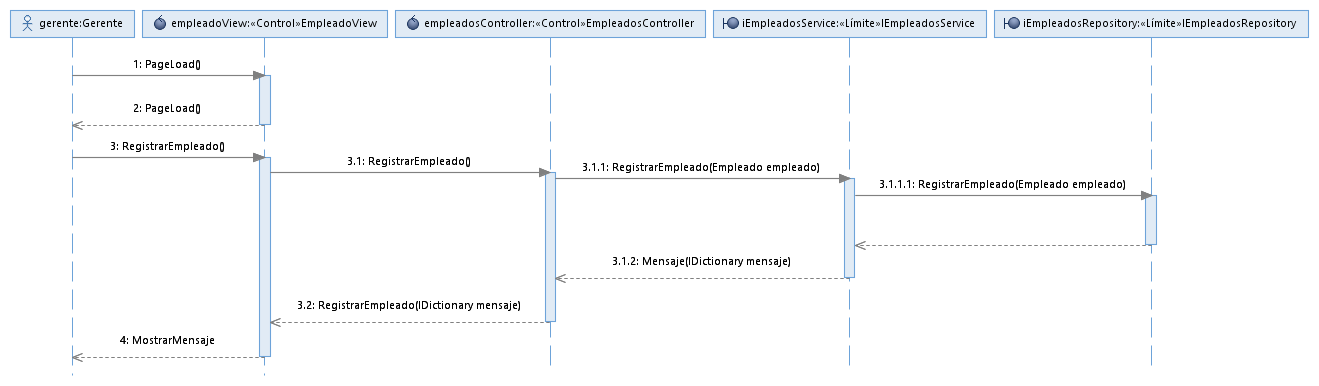
Empleado Service. Métodos implementados de la Interfaz service.

Interfaz Empleado Repository. Interfaz con métodos de Empleado.

Empleado Repository. Métodos implementados de la Interfaz Repository.

Empleado Validator. Metodos para validar Empleado.

**Escenario Básico**



**Registrar Vehículos**

**Vista de los participantes**

Vehículo Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Vehículo Service. Interfaz con métodos de Vehículo.

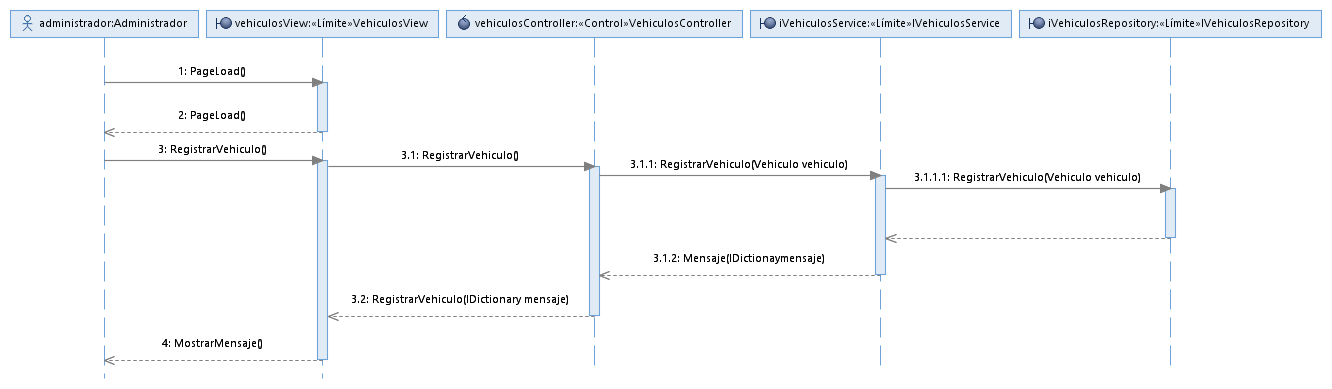
Vehículo Service. Métodos implementados de la Interfaz service.

Interfaz Vehículo Repository. Interfaz con métodos de Vehículo.

Vehículo Repository. Métodos implementados de la Interfaz Repository.

Vehículo Validator. Metodos para validar Vehículo.

**Escenario Básico**



**Buscar Vehículos**

**Vista de los participantes**

Vehículo Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Vehículo Service. Interfaz con métodos de Vehículo.

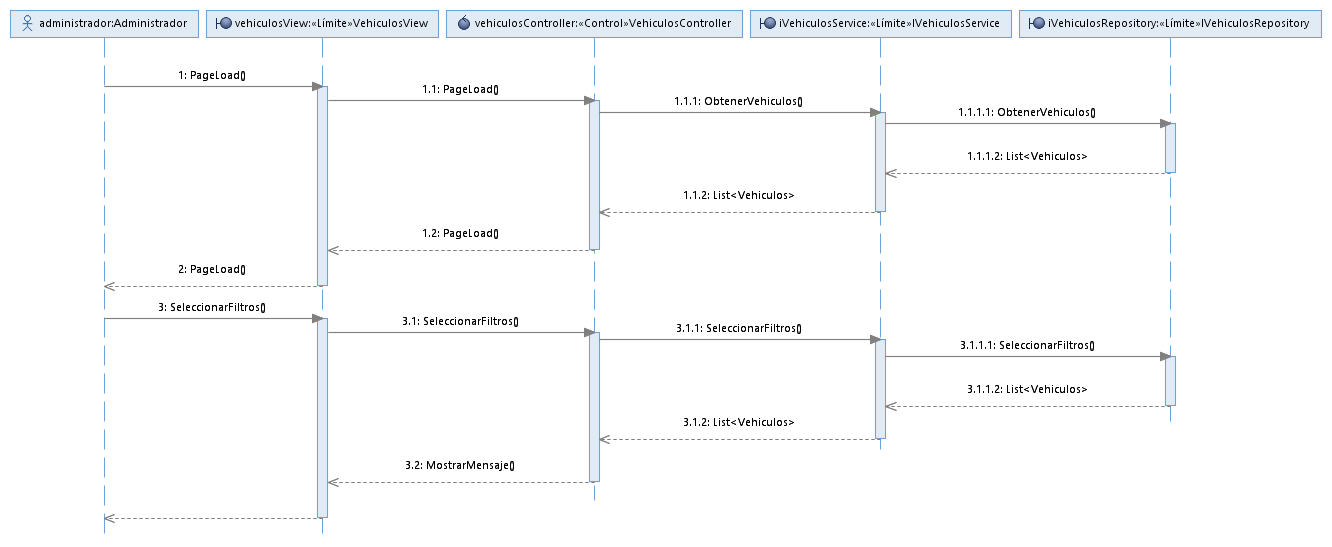
Vehículo Service. Métodos implementados de la Interfaz service.

Interfaz Vehículo Repository. Interfaz con métodos de Vehículo.

Vehículo Repository. Métodos implementados de la Interfaz Repository.

Vehículo Validator. Metodos para validar Vehículo.

**Escenario Básico**

****

**Registrar Horarios**

**Vista de los participantes**

Horario Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Horario Service. Interfaz con métodos de Horario.

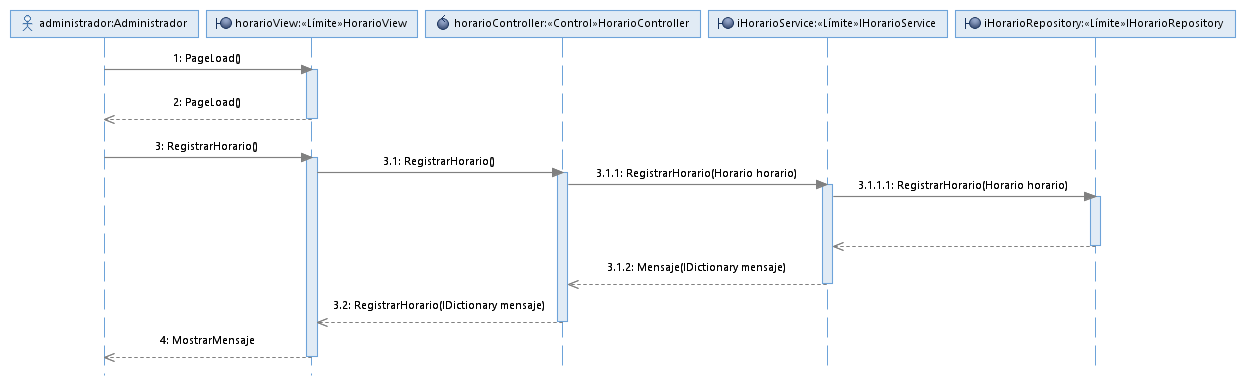
Horario Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Horario.

Horario Repository. Métodos implementados de la Interfaz Repository.

Horario Validator. Metodos para validar Horario.

**Escenario Básico**

****

**Registrar Asientos**

**Vista de los participantes**

Asiento Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Asiento Service. Interfaz con métodos de Asiento.

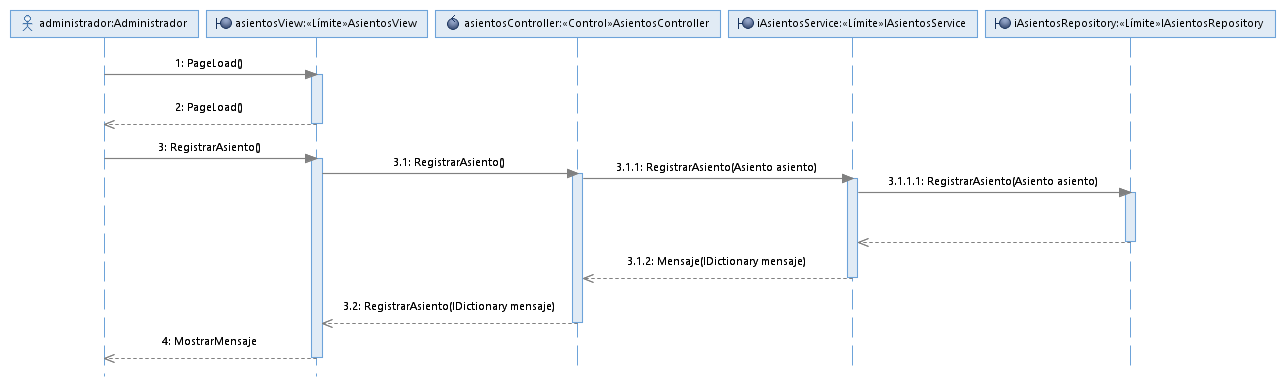
Asiento Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Asiento.

Asiento Repository. Métodos implementados de la Interfaz Repository.

Asiento Validator. Metodos para validar Asiento.

**Escenario Básico**

****

**Buscar Asientos**

**Vista de los participantes**

Asiento Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Asiento Service. Interfaz con métodos de Asiento.

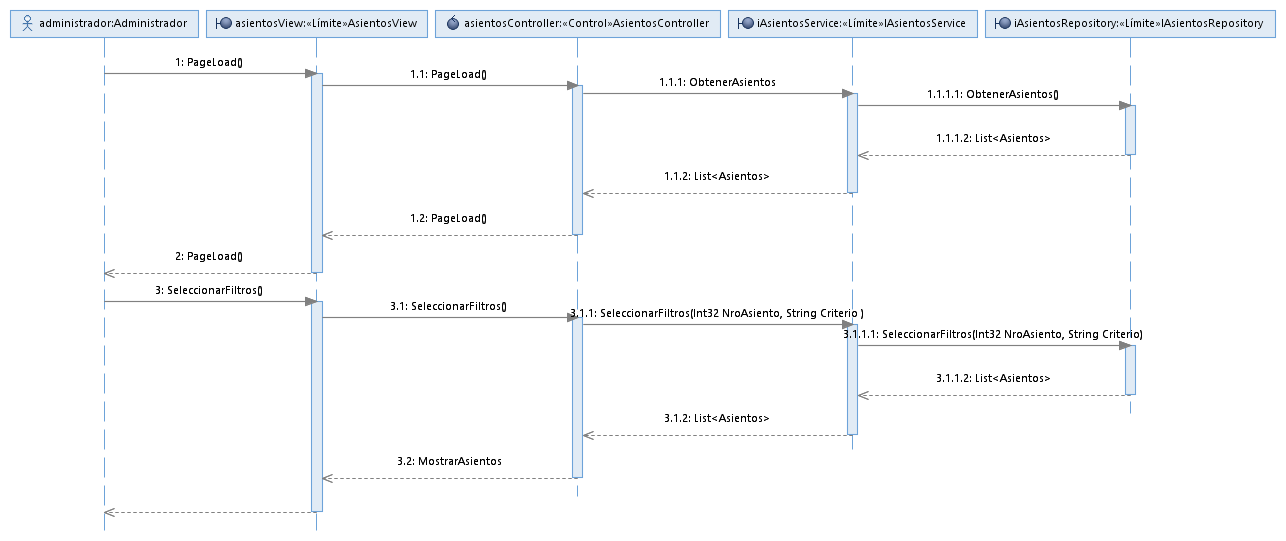
Asiento Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Asiento.

Asiento Repository. Métodos implementados de la Interfaz Repository.

Asiento Validator. Metodos para validar Asiento.

**Escenario Básico**

****

**Registrar Estaciones**

**Vista de los participantes**

Estaciones Controller. Clase que se encarga de enviar y recibir datos a la Vista.

Interfaz Estaciones Service. Interfaz con métodos de Estaciones.

Estaciones Service. Métodos implementados de la Interfaz service.

Interfaz Reserva Repository. Interfaz con métodos de Estaciones.

Estaciones Repository. Métodos implementados de la Interfaz Repository.

Estaciones Validator. Metodos para validar Estaciones.

**Escenario Básico**

