Arquitectura de Software - Obligatorio

Arquitectura de software (Universidad ORT Uruguay)

UNIVERSIDAD ORT URUGUAY

Facultad de Ingeniería

EnviosYa

DESCRIPCIÓN DE ARQUITECTURA

Arquitectura de Software

PROFESOR: Gastón Mousqués

Junio 2017

ÍNDICE

1. Introducción

1.1 Propósito

El propósito del presente documento es proveer una especificación completa de la arquitectura del sistema desarrollado para el área de operaciones de la empresa EnviosYa, en lo sucesivo se lo denominará Sistema EnviosYa.

1.2 Estructura

El presente documento está estructurado en secciones donde cada una de ellas se centra en un aspecto puntual del sistema describiendo sus características y las decisiones, con sus respectivas justificaciones, que llevaron al sistema a presentar la arquitectura que posee en la actualidad.

En primera instancia se describe el propósito general del sistema, incluyendo un detalle de los requerimientos funcionales y no funcionales del mismo, de forma que el lector pueda familiarizarse con las características que se buscaban en el sistema y, por ende, motivaron la arquitectura resultante.

Luego, se presenta la documentación de la arquitectura. Esta sección, mediante el uso de diferentes vistas, busca explicar al lector cómo se da soporte a los requerimientos especificados en la sección anterior.

La siguiente sección presenta el conjunto de directrices definidos por el grupo de trabajo, en base a estándares de la industria, para asegurar una buena calidad del código que da forma al sistema.

También se incluye una sección que indica que dependencias existen entre el sistema y sistemas desarrollados por terceros.

Finalmente, se incluye un manual de instalación con pasos para facilitar el despliegue del sistema, de forma exitosa, en el ambiente de producción.

2. Antecedentes

2.1 Propósito del sistema

El sistema a desarrollar abarca el área de operaciones de la empresa EnviosYa. Para ello el mismo se encarga de permitir la manipulación de toda la información referente a los clientes, cadetes y envíos de paquetes. Además, brinda las operaciones necesarias para la calificación de cadetes y el servicio de envíos en sí mismo, así como la posibilidad de administrar esta información.

Los actores del sistema, a quienes nos referiremos de forma general como usuarios, tienen los siguientes roles:

- Cliente: usuario que envía un paquete o es destinatario de un envío.
- Cadete: usuario que se encarga del transporte de paquetes.
- Administrador: usuario que se encarga del control y la administración de los datos que ingresan al sistema.

2.2 Requerimientos significativos de Arquitectura

A continuación se detallan las restricciones, así como también, los requerimientos funcionales y no funcionales más significativos del sistema. Los mismos definen las operaciones que los usuarios deben ser capaces de realizar utilizando el sistema y las características que el mismo debe tener para satisfacer ciertos atributos de calidad previamente identificados.

2.2.1 Restricciones

- El sistema debe desarrollarse en JEE.
- El sistema debe exponer una API REST para ser consumida por las aplicaciones “cliente” (aplicaciones móviles, clientes web, etc.).
- El sistema debe estar modularizado en los siguientes sub sistemas, donde cada uno debe cumplir con las características indicadas:

Módulo	Responsabilidad
Shipments	Implementa las reglas de negocio para el registro de solicitudes de envío de los clientes, medios de pago, preferencias. La entidad principal es Shipment. La latencia promedio de las operaciones que expone este módulo es de 100 milisegundos.
Clients	Implementa las reglas de negocio para gestión de los clientes que solicitan envíos, sus datos personales, ubicaciones, etc. La entidad principal es Client. La latencia promedio de las operaciones que expone este módulo es de 150 milisegundos.
Cadets	Implementa las reglas de negocio para gestión de los cadetes que realizan los envíos, sus datos personales, los vehículos que usan, etc. La entidad principal es Cadet. La latencia promedio de las operaciones que expone este módulo es de 70 milisegundos.
Notifications	Implementa y provee a los demás módulos formas de comunicación interna y externa a través de diferentes medios (mensajería, sms, email, telefónica, etc.). Las entidades principales son Target y Message. La latencia promedio de las operaciones que expone este módulo es de 70 milisegundos.
Reviews	Implementa la gestión de las calificaciones que hacen los clientes sobre los envíos y los cadetes. La entidad principal es Review.

2.2.2 Resumen de Requerimientos Funcionales

ID Requerimiento	Descripción	Módulo	Actor
RF 1 - Mantenimiento de Cadetes y Vehículos	El sistema deberá permitir registrar la información de los cadetes (CI, Nombre, Apellido, Correo Electrónico), sus Vehículos (Matrícula, Descripción) y asociarlos. También se deberá poder modificar y borrar a los cadetes y sus vehículos. Las cédulas de los cadetes no pueden ser repetidas y se deberá validar que el correo electrónico sea válido.	Cadets	Usuario
RF 2 - Mantenimiento de Clientes	El sistema deberá permitir registrar la información de los clientes (CI, Nombre, Apellido, Correo Electrónico, información sobre medios de pago). También se deberá poder modificar los datos y borrar a los clientes. Las cédulas no pueden ser repetidas, y se debe validar que el correo electrónico sea válido.	Clients	Usuario
RF 3 - Registrar una solicitud de envío	El sistema deberá permitir que los clientes puedan solicitar envíos a destinatarios. Para ello deberán poder crear un Envío (ID, descripción, Ubicación del Cliente que Envía, Ubicación Cliente Destinatario, Forma de pago, comisión, Foto Paquete y otros datos que sean pertinentes). El sistema debe verificar que el cliente y destinatario sean usuarios registrados y responder con los datos de los 4 cadetes más cercanos que puedan recoger el envío. El cliente selecciona el cadete que desea y el sistema le asigna el envío notificándole los datos del envío que debe realizar.	Shipments	Usuario
RF 4 - Registrar recepción de envío	El destinatario confirma la recepción del envío y el sistema notifica a ambos usuarios los datos del envío, de su recepción y la información sobre el cadete que realizó el traslado. La notificación debe ser por correo electrónico, y el correo debe incluir un link para que ambos usuarios puedan calificar el servicio y al cadete que realizó el traslado.	Shipments	Usuario
RF 5 - Calificar el servicio y el Cadete	Utilizando el link recibido por correo ambos usuarios deben poder calificar al servicio y al cadete. Para ambos casos el usuario puede valorar utilizando una escala de 0 a 5 estrellas y agregar un comentario.	Reviews	Usuario

RF 6 - Listar los envíos pendientes de calificación de un cliente	El sistema obtiene del módulo Shipment la lista de envíos del cliente, ordenados cronológicamente de forma descendente. El sistema obtiene del módulo Reviews la lista de calificaciones en estado “approved” para esos envíos. El sistema compara los resultados anteriores y retorna al usuario la lista de envíos pendientes de calificación.	Reviews	Usuario
RF 7 - Listar todas las calificaciones de un Cadete	El sistema obtiene del módulo Reviews la lista de calificaciones en estado “approved” para el Cadete, ordenadas cronológicamente de forma descendente y la retorna al usuario.	Reviews	Usuario
RF 8 - Rechazar manualmente una calificación inapropiada a un Cadete	El administrador obtiene la lista de calificaciones de un Cadete (RF7). Selecciona una calificación y la opción “rechazar”. El sistema marca la calificación con estado “rejected”.	Reviews	Administrador
RF 9 - Consultar las calificaciones no aprobadas	El administrador accede a la opción de consultar calificaciones (no visible para otros usuarios). Se le presenta la posibilidad de filtrar su consulta por estado e intervalo de fechas de registro de las calificaciones. El sistema obtiene del módulo de Reviews la lista de calificaciones, aplicando los filtros seleccionados y la retorna al usuario.	Reviews	Administrador

2.2.3 Resumen de Requerimientos No Funcionales

ID Requerimiento No Funcional	RF asociados	Atributo de calidad asociado	Descripción
RNF 1 - Configuración del proceso de creación de una Review	RF5	Modificabilidad, Disponibilidad	La estructura del proceso debe permitir agregar, quitar o intercambiar pasos con el menor impacto posible tanto en tiempo de desarrollo como de despliegue; así como soportar el reintento de los pasos que no lleguen a cumplirse en caso de que el proceso se vea interrumpido por cualquier motivo (por ejemplo fallas propias o de sistemas externos con los que se interopera).
RNF 2 - Tiempo de respuesta de creación de una Review	RF5	Performance	El tiempo de respuesta promedio de todos los módulos de back-end considerado aceptable para todas las peticiones que se realizan desde las aplicaciones “cliente” (apps móviles, clientes web) es de 200ms.

			En particular el proceso de creación de una Review no debe superar los 350ms.
RNF 3 - Usuarios registrados	Todos	Seguridad	Todas las funcionalidades descritas sólo pueden ser utilizadas por usuarios debidamente autenticados. Por razones estratégicas del negocio, los usuarios solo pueden registrarse en el sistema utilizando sus cuentas de Facebook, por lo tanto la autenticación de los usuarios se debe realizar contra esos proveedores. Es decir que para acceder a su cuenta en EnviosYa, el usuario debe loguearse a través de Facebook.
RNF 4 - Gestión de errores y fallas	Todos	Disponibilidad	El sistema debe proveer suficiente información, de alguna forma, que permita conocer el detalle de las tareas que realiza. En particular, en el caso de ocurrir una falla o cualquier tipo de error, es imprescindible que el sistema provea toda la información necesaria que permita a los administradores hacer un diagnóstico rápido y preciso sobre las causas.
RNF 5 - Recuperación y Fallas	Todos	Disponibilidad	Sería deseable que la información producida por el sistema pueda ser reutilizada de forma automática en aquellos casos en que es posible reconstruir alguna tarea que no haya finalizado debidamente (por error, falla o intervención de un administrador).
RNF 6 - Independencia de “consultas” y “escrituras”	Todos	Modificabilidad	Teniendo en cuenta que se trata de un sistema que soporta una gran demanda de los usuarios, donde la relación entre las operaciones de “consulta de datos” e “ingreso de datos” son del orden de 10 a 1 (es decir, por cada ingreso de datos se realizan 10 consultas), se desea construir la aplicación de forma tal que las modificaciones que afecten las funcionalidades de ingreso de datos (como la creación de una Review) no impliquen la necesidad de re-desplegar los componentes que implementan las funcionalidades de consultas. Es decir, es deseable que los componentes que implementan operaciones de ingreso de datos puedan gestionarse y desplegarse de forma independiente de los

			componentes que implementan operaciones de consulta de datos.
RNF 7 - Protección de los datos de los clientes.	Todos	Seguridad	Los datos de los clientes deben protegerse de accesos no autorizados. Se debe hacer especial énfasis en proteger la información de los medios de pago asociados a los clientes.
RNF 8 - Información de auditoría.	Todos	Seguridad	El sistema debe registrar información que permita realizar auditorías de acceso de forma de identificar los accesos autorizados y no autorizados al sistema.
RNF 9 - Tiempo de respuesta de creación de una solicitud de envío	RF3	Performance	El tiempo de respuesta promedio de todos los módulos de back-end considerado aceptable para todas las peticiones que se realizan desde las aplicaciones “cliente” (apps móviles, clientes web) es de 200ms. En particular el proceso de creación de una solicitud no debe superar los 200ms para aquellos usuarios que hayan tenido alguna actividad en el sistema en los últimos 15 días. Para aquellos usuarios que no hayan tenido actividad por más de 15 días el tiempo de respuesta no debe superar los 500ms.
RNF 10 - Modificación del cálculo de costos de los envíos.	RF3	Modificabilidad	A futuro se debe poder modificar el mecanismo de cálculo de costo del envío con el menor impacto posible en los módulos del sistema y con el redespliegue del menor número de componentes posibles.

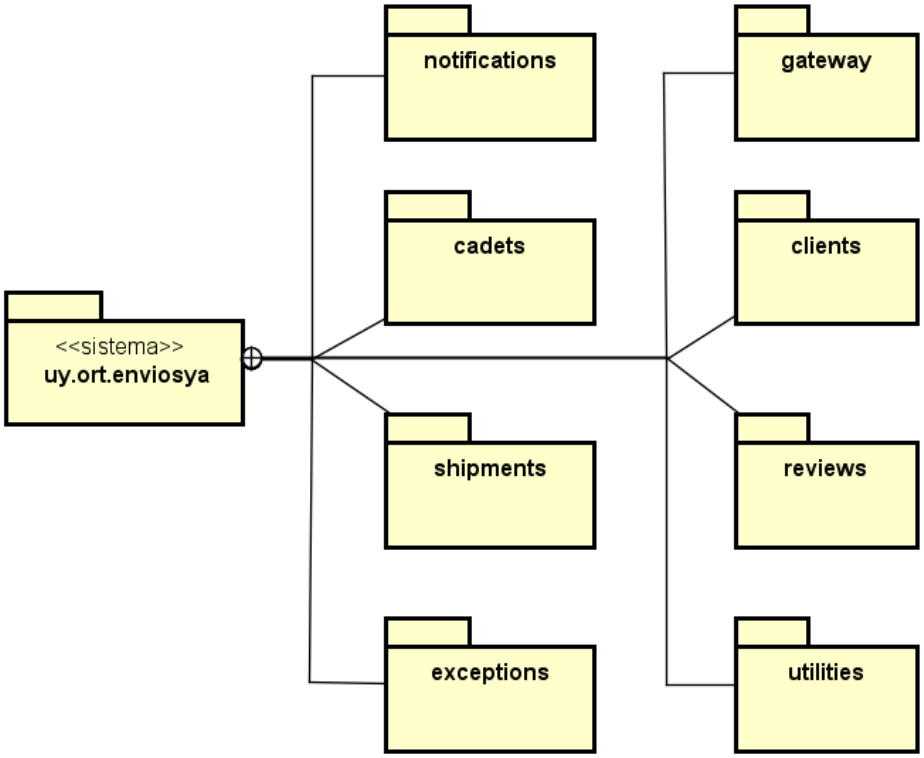
3. Documentación de la arquitectura

3.1 Vistas de Módulos

3.1.1 Vista de Descomposición

3.1.1.1 Sistema

Representación primaria



Catálogo de elementos

Elemento	Responsabilidades
Notifications	Provee a los otros módulos con mecanismos de comunicación para informar a los clientes del sistema a través de diferentes medios.
Cadets	Implementa las reglas de negocio para la gestión de los cadetes que realizan los envíos.
Shipments	Implementa las reglas de negocio para la gestión de los envíos.
Exceptions	Contiene las excepciones propias del sistema que se utilizan en varios de los otros módulos.
Gateway	Expone los recursos del sistema a los clientes y facilita el manejo de las sesiones de los usuarios comportándose como una puerta de entrada al sistema.
Clients	Implementa las reglas de negocio para la gestión de los clientes que solicitan o son destinatarios de los envíos.
Reviews	Implementa la gestión de las calificaciones que hacen los clientes sobre el servicio y los cadetes.

Utilities	Encapsula funcionalidad común utilizada por diferentes módulos para el manejo de fechas, direcciones de correo electrónico y encriptación de datos.

Justificación

- Restricciones

  Parte de la separación en módulos ilustrada en esta vista estaba definida como una de las restricciones del sistema (véase Restricciones). En las mismas se indicaba el sistema debía de estar dividido en los siguientes subsistemas:

  - Shipments
  - Clients
  - Cadets
  - Notifications
  - Reviews

  Donde cada uno de ellos expone interfaces específicas de forma de poder comunicarse entre sí a través de llamadas remotas y, además, gestiona su propio repositorio de datos.

- Seguridad

  El módulo Gateway funciona como el único punto de entrada al sistema y luego se encarga de realizar las llamadas a los demás módulos para responder las consultas de los clientes. Previo a derivar las consultas este módulo se encarga de validar que el cliente este autorizado, para ello gestiona las sesiones de los clientes permitiendo que los de demás modulo no manejen sesiones. Nos permite centrar los requerimientos de seguridad en un solo módulo, aunque lleva a un aumento en el tiempo de respuesta ya que deben redirigirse las consultas.

- Disponiblidad

  El módulo Exceptions agrupa todas las excepciones propias del sistema que son utilizadas por varios de los otros módulos. Tener excepciones propias facilita la gestión de errores ya que rastrear de donde proviene el error se torna más fácil. De esta forma contribuimos a cumplir con el RNF4.

  De esta forma definimos

  - EnviosYaException: agrupa todas las posibles excepciones propias del sistema.
  - PersistenceException: utilizada en los módulos de persistencia.
  - DoesNotExistException: utilizada en los módulos de persistencia cuando el dato requerido no existe.
  - ServiceException: utilizada en los módulos que contienen la lógica de negocio.
  - ValidationException: utilizada en los módulos que validan los datos recibidos por el sistema.

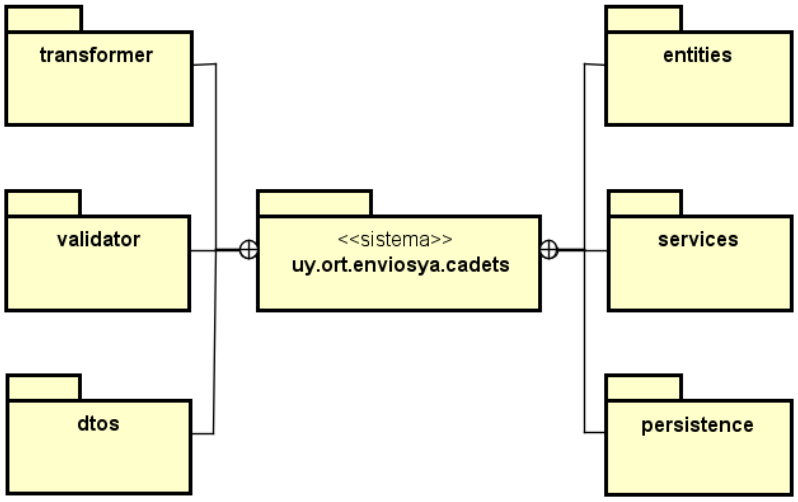
- Modificabilidad

  El módulo Utilities abstrae funcionalidades genéricas que se utilizan en varios lugares del sistema como es la validación de emails, los mecanismos de encriptaciones y funcionalidades para las fechas. Nos permiten reutilizar el código y vuelven el sistema más mantenible, ya que si se debe hacer una modificación en las mismas hay que modificar únicamente este módulo.

A continuación se incluyen vistas que presentan la descomposición de cada uno de los módulos identificados, a excepción de Exceptions ya que no está estructurado en sub módulos.

3.1.1.2 Cadets

Representación primaria



Catálogo de elementos

Elemento	Responsabilidades
Entities	Contiene las entidades del sub sistema que permiten representar toda la información necesaria para la gestión de los cadetes. Las mismas están compuestas por la entidad principal, Cadet, definida en las restricciones y otra entidad para representar los vehículos, Vehicle.
Services	Maneja la lógica del negocio y es la puerta de entrada al sub sistema.
Persistence	Gestiona el repositorio de datos del sub sistema.
Transformer	Encapsula la transformación de DTOs a entidades y viceversa.
Validator	Encapsula la validación de los DTOs recibidos por el sub sistema. Valida, entre otras cosas, que las cédulas no sean repetidas y que el correo electrónico sea válido, cumpliendo así con lo requerido como parte del requerimiento funcional de este sub sistema.
Dtos	Contiene los objetos que se utilizan para transportar datos entre este sub sistema y los otros sub sistemas. Los mismos son una versión de las dos entidades definidas por el sub sistema: vehículos y cadetes.

Justificación

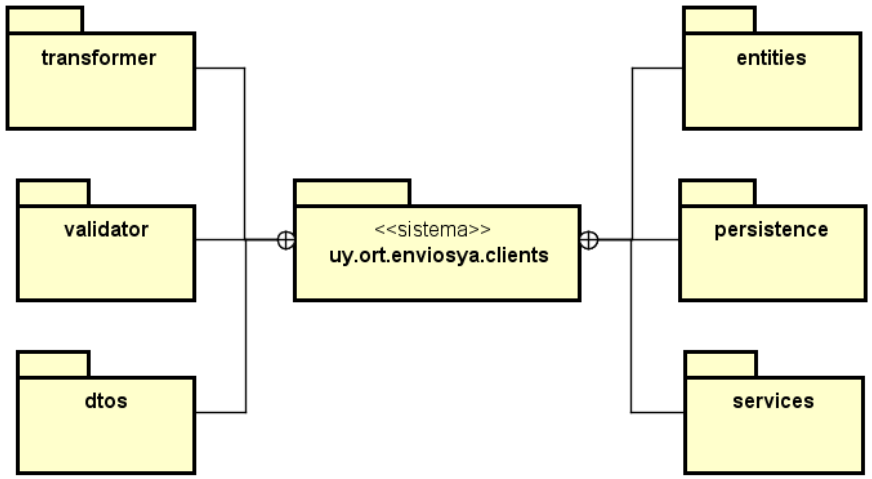
- Modificabilidad

  La separación en módulos de este sub sistema se realiza buscando que cada módulo del mismo cumpla con una responsabilidad específica, de esta forma reducimos la probabilidad de que un cambio afecte varios módulos.

  Dado que la comunicación entre este módulo y los demás se realiza de forma remota consideramos necesario exponer DTOs de forma que los otros módulos no deban acoplarse a las entidades definidas por este sub sistema.

3.1.1.3 Clients

Representación primaria



Catálogo de elementos

Elementos	Responsabilidades
Entities	Contiene las entidades del sub sistema que permiten representar toda la información necesaria para la gestión de los clientes. Las mismas están compuestas por la entidad principal, Client, definida en las restricciones y otra entidad para representar los medios de pago, PaymentMethod.
Services	Maneja la lógica del negocio y es la puerta de entrada al sub sistema.
Persistence	Gestiona el repositorio de datos del sub sistema.
Transformer	Encapsula la transformación de DTOs a entidades y viceversa.
Validator	Encapsula la validación de los DTOs recibidos por el sub sistema. Valida, entre otras cosas, que las cédulas no sean repetidas y que el correo electrónico sea válido, cumpliendo así con lo requerido como parte del requerimiento funcional de este sub sistema.
Dtos	Contiene los objetos que se utilizan para transportar datos entre este sub sistema y los otros sub sistemas. Los mismos son una versión de las dos entidades definidas por el sub sistema: clientes y métodos de pago.

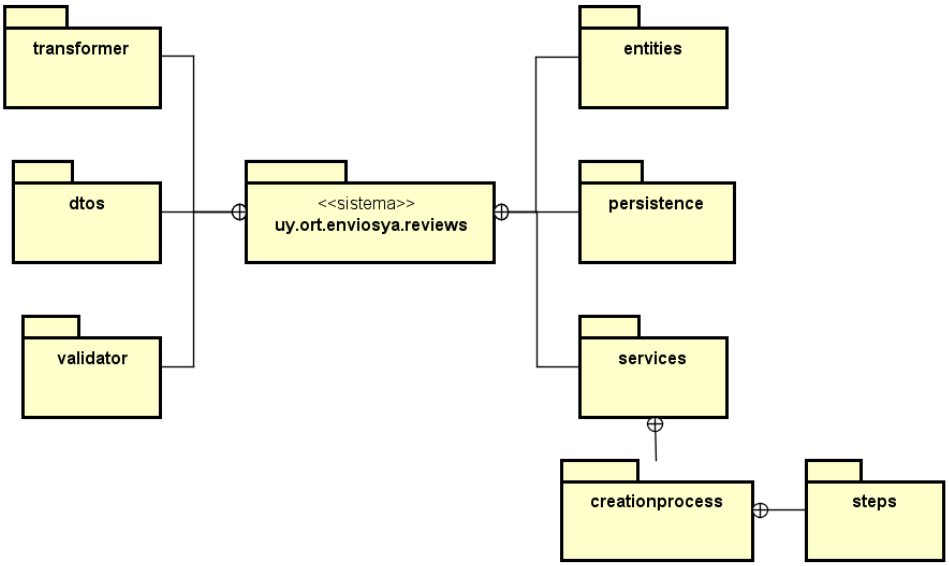
Justificación

Como puede la forma en que está estructurada este sub sistema es la misma que la vista en Cadets. Esto es así porque la funcionalidad provista por el mismo es idéntica, lo único que cambia es que la información gestionada refiere a los clientes del sistema en lugar de a los cadetes. Básicamente ambos se encargan del alta, baja y modificación de una entidad y sus atributos.

Al estructurar ambos sistemas de la misma forma mantenemos un patrón que facilita a los desarrolladores la identificación de las diferentes partes que componen el sub sistema y permite que el conocimiento adquirido para uno de ellos pueda ser utilizado en el otro.

3.1.1.4 Reviews

Representación primaria



Catálogo de elementos

Elementos	Responsabilidades
Entities	Contiene las entidades del sub sistema que permiten representar toda la información necesaria para la gestión de las calificaciones que hacen los clientes sobre los envíos y los cadetes. Las mismas están compuestas por la entidad principal, Review, definida en las restricciones y otra entidad Data, que encapsula la información contenida en la calificación al servicio y el cadete (el puntaje y el comentario).
Services	Maneja la lógica del negocio y es la puerta de entrada al sub sistema.
CreationProcess	Contiene funcionalidades genéricas que son utilizadas por todos los pasos de creación de una reseña, por ejemplo el resultado de un paso, y permite comenzar el proceso.
Steps	Agrupa, dentro del módulo services, los servicios que representan los diferentes pasos en el proceso de creación de una reseña.
Persistence	Gestiona el repositorio de datos del sub sistema.
Transformer	Encapsula la transformación de DTOs a entidades y viceversa.
Validator	Encapsula la validación de los DTOs recibidos por el sub sistema. Valida, entre otras cosas, que el usuario pueda valorar el servicio utilizando una escala de 0 a 5, cumpliendo así con lo requerido como parte del RF5.
Dtos	Contiene los objetos que se utilizan para transportar datos entre este sub sistema y los otros sub sistemas. Los mismos son una versión de las dos entidades definidas por el sub sistema: reseña y datos. Además contiene los enumeradores definidos para representar ciertos valores como el estado de la reseña y el sentimiento identificado, estos son utilizados tanto por las entidades como por los Dtos.

Justificación

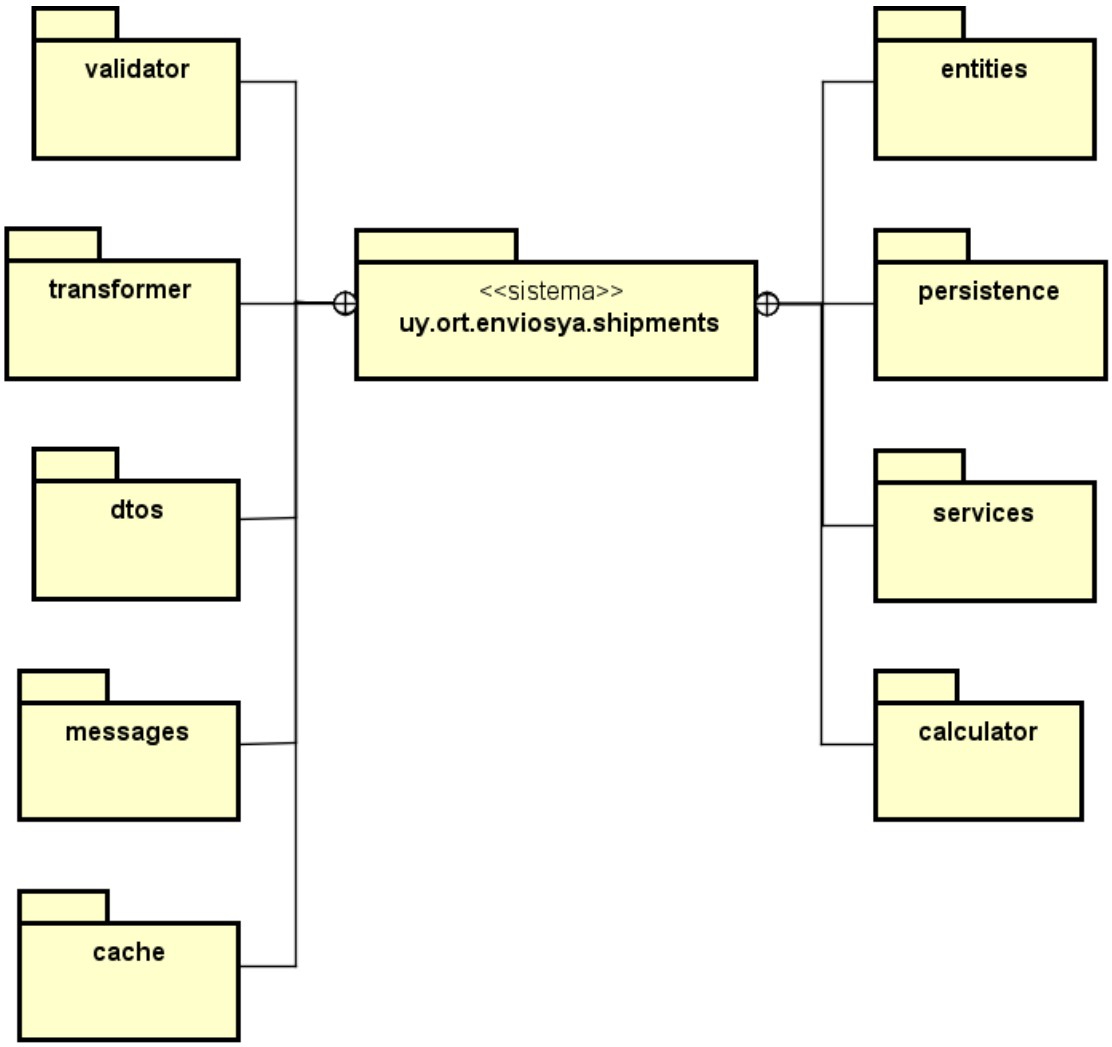
Nuevamente se repite la estructura vista en los sub sistemas anteriores, Cadets y Clients. Esto es así porque este sub sistema también se encarga de exponer funcionalidades para el alta, baja y modificación de una de las entidades del sistema, en este caso las calificaciones. Al repetir esta estructura en todos los sub sistemas que encargados de este tipo de acciones resulta más simple comprender como funciona cada sub sistema.

- Modificabilidad

  Dado que el proceso de creación de una reseña es un proceso complejo que debe ser fácil de modificar decidimos encapsularlo en un sub módulo de forma de que todos los cambios que el mismo deba sufrir se concentren únicamente en este módulo.

3.1.1.5 Shipments

Representación primaria



Catálogo de elementos

Elementos	Responsabilidades
Entities	Contiene las entidades del sub sistema que permiten representar toda la información necesaria para la gestión de los envíos. Las mismas están compuestas por la entidad principal, Shipment, definida en las restricciones, Location que guarda la información que refiere a la ubicación donde deberá ser retirado y enviado un

	paquete, y Client que guarda los datos del cliente que interesan para poder realizar el envío (ubicación y forma de pago seleccionada).
Services	Maneja la lógica del negocio y es la puerta de entrada al sub sistema.
Calculator	Encapsula el mecanismo utilizado para el cálculo del costo del envío. Incluye además un Dto para facilitar la comunicación con el módulo Services y una excepción propia para diferenciar los errores que puedan provenir del mecanismo para el cálculo del costo implementado. La implementación realizada, utilizando un servicio provisto por un tercero enviando la imagen del paquete, define además un conjunto de objetos para facilitar el envío de datos a la API utilizada y el parseo de las respuestas.
Persistence	Gestiona el repositorio de datos del sub sistema.
Transformer	Encapsula la transformación de DTOs a entidades y viceversa.
Validator	Encapsula la validación de los DTOs recibidos por el sub sistema.
Dtos	Contiene los objetos que se utilizan para transportar datos entre este sub sistema y los otros sub sistemas. Los mismos son una versión dos de las entidades definidas por el sub sistema: envío y ubicación. Además define objetos para facilitar la creación de un nuevo envío, tanto para que el usuario ingrese el pedido como para responderle con la información de los cadetes. Finalmente, contiene un enumerador definido para representar quien es el encargado de pagar la comisión utilizado tanto por las entidades como por los Dtos.
Messages	Se encarga comunica con el subsistema Notifications para indicar el envío de notificaciones a usuarios u otros subsistemas.
Cache	Contiene un cache de datos utilizados por el sub sistema que provienen de otros sub sistemas. Con este módulo buscamos disminuir el tiempo que le lleva al sistema procesar un nuevo envio para aquellos usuarios que hayan tenido actividad por más de 15 días, como se pide en el RNF10.

Justificación

Vuelve a repetirse la estructura de módulos base utilizada a lo largo del sistema en aquellos subsistemas encargados del alta, baja y modificación de una entidad, siendo en este caso la misma los envíos. Los beneficios de repetir esta estructura en todos los sub sistemas que exponen este tipo de funcionalidades ya fueron explicados. Aparecen además dos nuevos módulos, donde cada uno de ellos tiene una responsabilidad específica.

- Modificabilidad

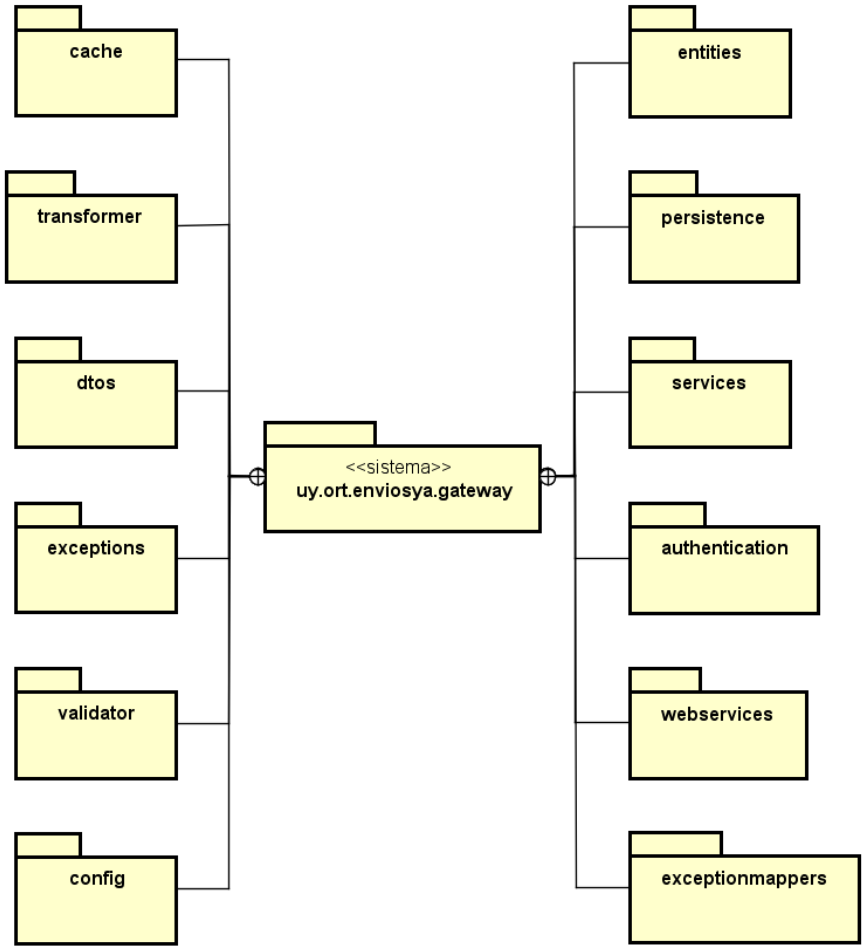
  Calculator encapsula el cálculo del costo del envío, de esta forma si el proceso de cálculo cambia impacta únicamente en este sub modulo cumpliendo, en parte, lo planteado por el RNF 10.

- Performance

  Cache almacena datos referentes a entidades gestionadas por otros sub sistemas de forma de que puedan ser accedidos rápidamente y a un menor costo cuando el sub sistema lo requiera.

3.1.1.6 Gateway

Representación primaria



Catálogo de elementos

Elementos	Responsabilidades
Entities	Contiene las entidades del sub sistema que permiten representar toda la información necesaria para la gestión de las sesiones de los usuarios del sistema. Las mismas almacenan la información de los administradores, las sesiones y los tipos de usuarios.
Services	Maneja la lógica del negocio y es la puerta de entrada al sub sistema.
Persistence	Gestiona el repositorio de datos del sub sistema.
Transformer	Encapsula la transformación de DTOs a entidades y viceversa.
Validator	Encapsula la validación de los DTOs recibidos por el sub sistema.
Dtos	Contiene los objetos que se utilizan para transportar datos entre este sub sistema y los otros sub sistemas.
Cache	Contiene un cache de datos utilizados por el sub sistema que provienen de otros sub sistemas.

Exceptions	Contiene excepciones propias del sub sistema, las mismas refieren a errores con la sesión del usuario.
Config	Contiene la configuración de la aplicación, puntualmente la referente a los recursos REST que expone este sub sistema.
Authentication	Encapsula los mecanismos utilizados para autenticar y autorizar a los usuarios que realizan consultas a los recursos expuestos. Para ello define una serie de filtros que se ejecutan antes de que la consulta sea procesada. Contiene además funcionalidades como la lectura del token en el encabezado de las consultas y la validación del token contra la API provista por Facebook, el proveedor contra quien se implementó el mecanismo de autenticación.
Exception Mappers	Este paquete contiene clases que realizan el mapeo de las excepciones del lenguaje a los mensajes de error apropiados para ser enviados como respuesta en los servicios (mediante HTTP). Esto incluye el código de error que corresponda más un mensaje adecuado al error.
Web Services	Contiene los recursos REST que expone el sistema a los clientes. Los mismos son la puerta de entrada a todo el sistema.

Justificación

Este es el último de los sub sistemas del sistema que presenta la estructura definida para la gestión de una entidad, en este caso dicha entidad son las sesiones de los usuarios. Los beneficios de mantener una única estructura en estos casos ya fueron comentados.

Los demás módulos que componen este sub sistema están separados de la forma presentada según las responsabilidades de cada uno. En general, ya que como mencionamos este sub sistema es la puerta de entrada para todo el sistema, buscan permitir el acceso al sistema de forma segura y manejar las respuestas que el sistema da a los clientes que lo utilizan.

- Disponibilidad

  El módulo exception mappers se encarga de manejar las respuestas que da el sistema en caso de que la llamada a uno de los recursos de la API no sea exitosa. Se busca que las mismas sean claras y permitan a los clientes que consumen estos servicios entender el error.

- Seguridad

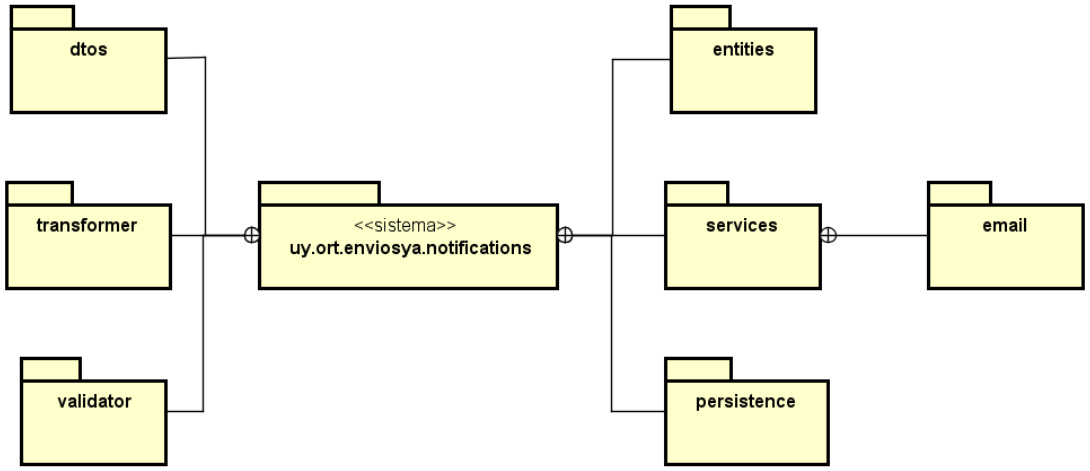
  El módulo authentication funciona como un filtro que permite identificar si un usuario puede tener acceso al recurso requerido, en cuyo caso el proceso continua, o no, de forma de poder devolver un error indicando que no se cuenta con los permisos necesarios. De esta forma nos aseguramos que las funcionalidades solo puedan ser utilizadas por usuarios registrados como se pide en el RNF3.

- Performance

  El módulo Cache surge debido a que la información de los cadetes y clientes es gestionada por otros sub sistemas. Al crear un cache disminuimos el costo de consultar esta información, mejorando así la performance de este sub sistema en particular ya que es el que brinda acceso a todo el sistema y por ello podría convertirse en un cuello de botella.

3.1.1.7 Notifications

Representación primaria



Catálogo de elementos

Elementos	Responsabilidades
Services	Contiene la lógica que se encarga de recibir los mensajes enviados por los otros sub sistemas y decidir qué acción realizar en respuesta.
Entities	Contiene las entidades del sub sistema que permiten representar toda la información necesaria para la gestión de las notificaciones que el sistema envía al cliente. Las mismas están compuestas por las entidades principales, Message y Target, definidas en las restricciones.
Email	Encapsula el envío de emails conteniendo distinta información que debe ser notificada a los distintos actores.
Persistence	Gestiona el repositorio de datos del sub sistema.
Transformer	Encapsula la transformación de DTOs a entidades y viceversa.
Validator	Encapsula la validación de los DTOs recibidos por el sub sistema.
Dtos	Expone objetos que son utilizados por los otros sub sistemas para enviar mensajes que puedan ser procesados en este sub sistema. Además de exponer una versión de las entidades, expone un conjunto de objetos para representar el contenido de distintos tipos de mensajes.

Justificación

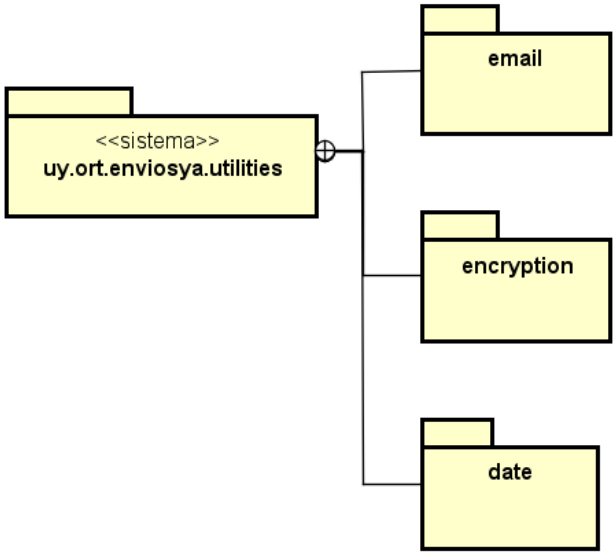
Dado que los demás sub sistemas se comunican de manera remota con este y deben enviar conjuntos de datos conteniendo información variada resulta necesario exponer DTOs de forma de poder encapsular esta información para que pueda ser enviada y recibida en conjunto. Los mismos se encuentran en el módulo Dtos.

- Modificabilidad

  El módulo de servicios se encarga de recibir todos los mensajes que se envían a esta sub sistema y dependiendo del target de los mismos lo deriva a sus sub módulos para poder así comunicar esta información a los clientes a través del medio de comunicación escogido. En este caso únicamente contamos con comunicación a través de email, encapsulada en el módulo homónimo, pero sería posible extender el sistema agregando otros sub módulos que implementen otros mecanismos de comunicación.

3.1.1.8 Utilities

Representación primaria



Catálogo de elementos

Elementos	Responsabilidades
Email	Implementa lo necesario para poder realizar la validación de emails.
Encryption	Contiene los mecanismos utilizados para la encriptación de datos que son utilizados en el sistema. Los mismos son necesarios para proteger los datos sensibles de los clientes, como se indica en el RNF7.
Date	Contiene los mecanismos necesarios para obtener la fecha actual y una fecha determinada a partir de la actual.

Justificación

- Modificabilidad

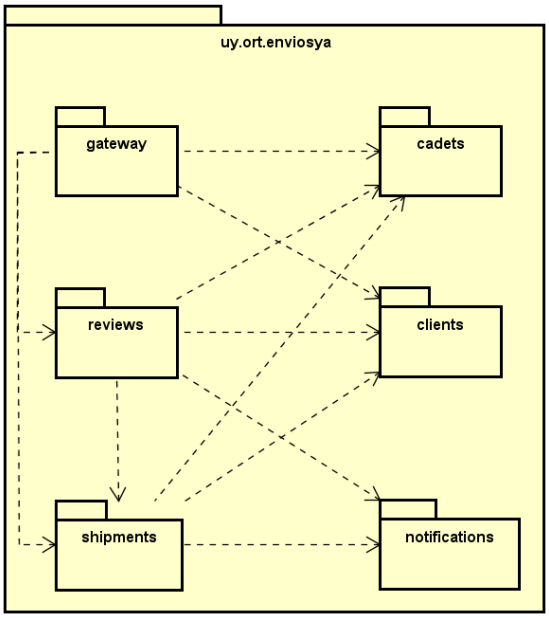
  La separación en módulos dentro de este sub sistema responde a las responsabilidades de cada una de las partes. De esta forma si se debe implementar una nueva utilidad, por ejemplo para la validación de números de teléfono o cédulas para utilizar en Clients y Cadets, la misma seria desarrollada en un nuevo módulo. Así conseguimos que el eje de cambio de cada módulo sea único y que la necesidad de desarrollar nuevas utilidades no afecte los módulos pre-existentes.

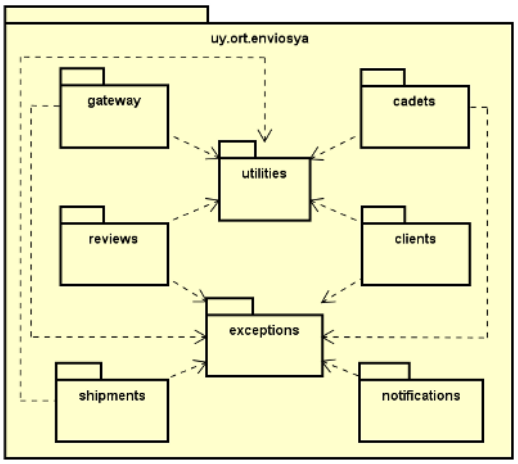
3.1.2 Vista de Uso

3.1.2.1 Sistema

Representación primaria

Las dependencias que se muestran en estos diagramas se dan en simultáneo dentro de la solución. Lo separamos para poder analizar mejor las dependencias del primer paquete ya que son claves para entender las relaciones entre los sub sistemas de la aplicación.





Catálogo de elementos

El catálogo de elementos es idéntico al presentado en la vista de descomposición (véase Sistema), por lo cual no será repetido en este punto.

Justificación

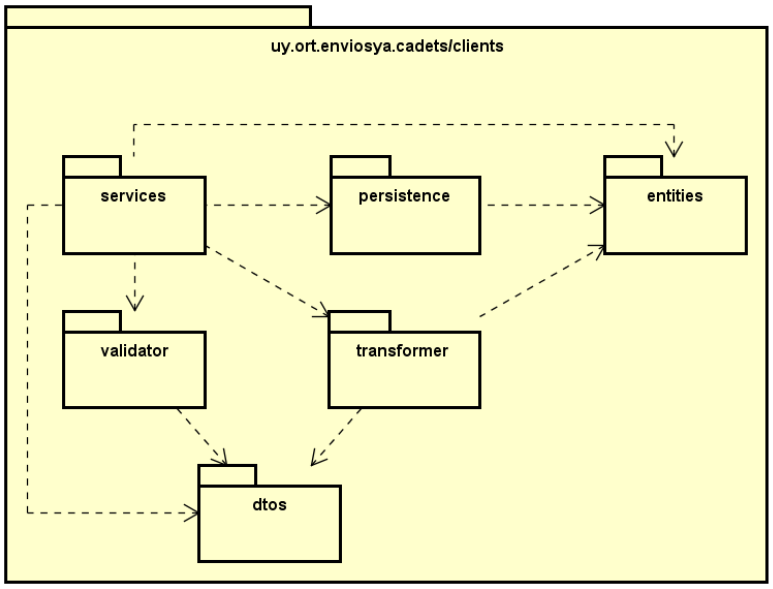
- Modificabilidad

  En el diagrama inferior se muestran las dependencias de los sub sistemas con los módulos Utilities y Exceptions, como vemos todos dependen de las excepciones y todos menos Notifications dependen de las utilidades. Esto nos lleva a pensar que haber encapsulado las utilidades y las excepciones en un módulo tuvo sentido, de otra forma el código estaría repetido en todos los otros módulos.

  En cuanto a las dependencias mostradas por el diagrama a la superior, es decir aquellas que realmente muestran la comunicación entre las partes centrales de nuestro sistema, queremos destacar que se cumple el principio de dependencias acíclicas. Gracias a esto el impacto del cambio será menor.

3.1.2.2 Cadets/Clients

Representación primaria



Catálogo de elementos

El catálogo de elementos es idéntico al presentado en la vista de descomposición (véase Cadets o Clients), por lo cual no será repetido en este punto.

Justificación

Como identificamos en la vista de los sub sistemas Cadets y Clients estan compuesto por los mismos módulos. Esto es así porque su única responsabilidad es encargarse de permitir registrar, modificar y borrar información relacionada a una de las entidades del sistema. No solo están compuestos por los mismos módulos, sino que la relación de uso que se da entre ellos es la misma (por los mismos motivos). Es por ello que se decide juntar ambas vistas en una sola para evitar ser repetitivo.

- Modificabilidad

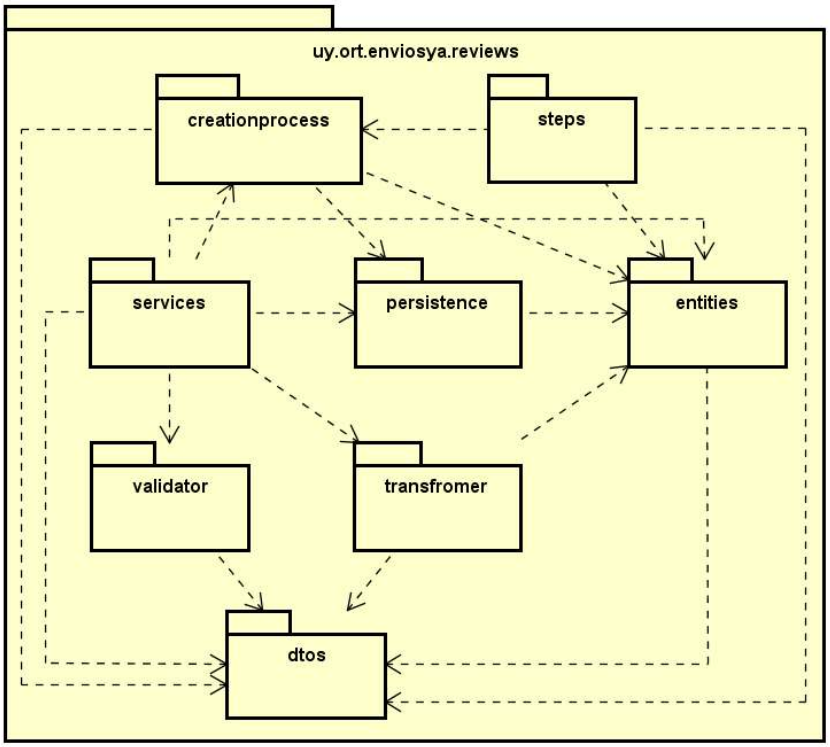
  El módulo Services se manejan tanto dtos como sus correspondientes entidades, en este se realiza el proceso de transformación de uno en otro, utilizando las funcionalidades provistas por el módulo Transformer. De esta forma conseguimos que la persistencia desconozca como se reciben los datos enviados por el cliente.

- Performance

  Las validaciones de la correctitud de los datos se realizan sobre los dtos (véase que Validator solo depende de Dtos), de esta forma conseguimos que la validación sea un filtro, colocándolo como el paso inicial a realizarse en las llamadas a la lógica de negocios. Si los datos ingresados no están completos o su formato es incorrecto inmediatamente se devuelve un error al usuario, evitando la realización de un proceso de transformación que podría ser costoso.

3.1.2.3 Reviews

Representación primaria



Catálogo de elementos

El catálogo de elementos es idéntico al presentado en la vista de descomposición (véase Reviews), por lo cual no será repetido en este punto.

Justificación

Es sumamente fácil notar que el diagrama presentado aquí es muy similar al expuesto en la sección anterior, esto es así porque este sub sistema también se encarga de la gestión de una de las entidades del sistema. Si bien provee una mayor cantidad de funcionalidades, puntualmente para diferentes tipos de listados de reseñas los mismos no se reflejan en el diagrama de uso ya que están encapsulados en el módulo sistemas.

- Disponiblidad

  Para poder llevar un registro de los pasos realizados y los resultados de los mismos es que CreationProcess se comunica con la persistencia de esta forma tenemos información suficiente como para poder reintentarlos en caso de que el proceso se vea interrumpido como se pide en el RNF1 y RNF5.

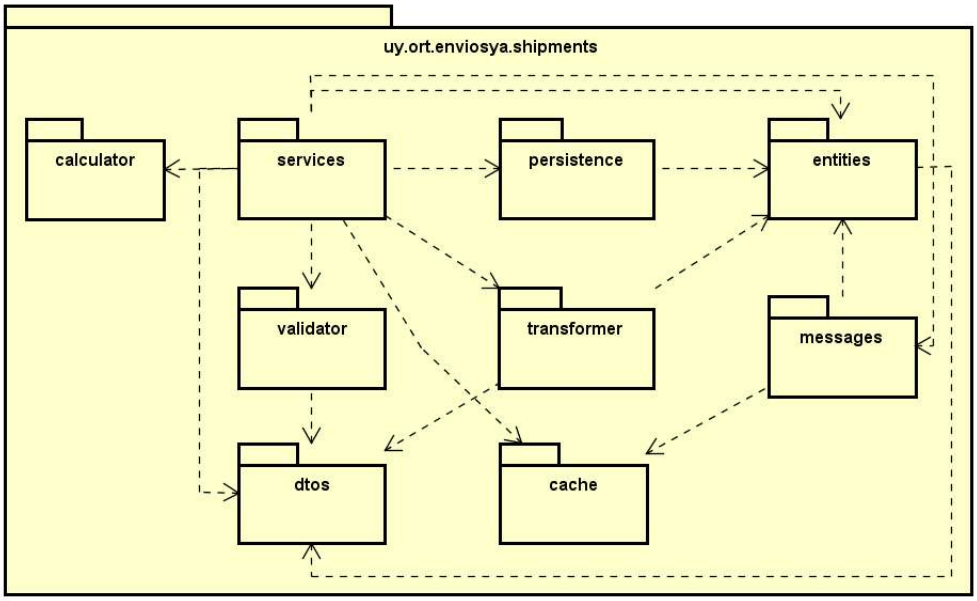
- Modificabilidad

  Existe una dependencia entre Entities y Dtos ya que ciertos atributos, definidos como enumeradores, son utilizados en ambos módulos. Se eligió colocarlos en Dtos, de forma de no acoplar otros sub sistemas a Entities y, al mismo tiempo, no tener código duplicado.

  Steps no tiene una dependencia hacia la persistencia, sino que de esto se encarga únicamente el módulo CreationProcess. De esta forma si se quisiera modificar la información guardad para los pasos debería de modificarse únicamente este módulo.

3.1.2.4 Shipments

Representación primaria



Catálogo de elementos

El catálogo de elementos es idéntico al presentado en la vista de descomposición (véase Shipments), por lo cual no será repetido en este punto.

Justificación

Es fácil ver en el diagrama que la estructura presente en la representación provista en la vista anterior para Clients y Cadets está presente también en este módulo. Si quitáramos los módulos Calculator, Messages y Cache estaríamos frente a el mismo diagrama. Esto, como explicamos en la vista de descomposición se debe a que este módulo también realiza el ABM de una entidad. Todo lo mencionado para esta estructura en la vista anterior resulta válido para esta vista, así como también los motivos para tener una dependencia entre Entities y Dtos que fueron explicados en la vista de uso de Reviews.

- Performance

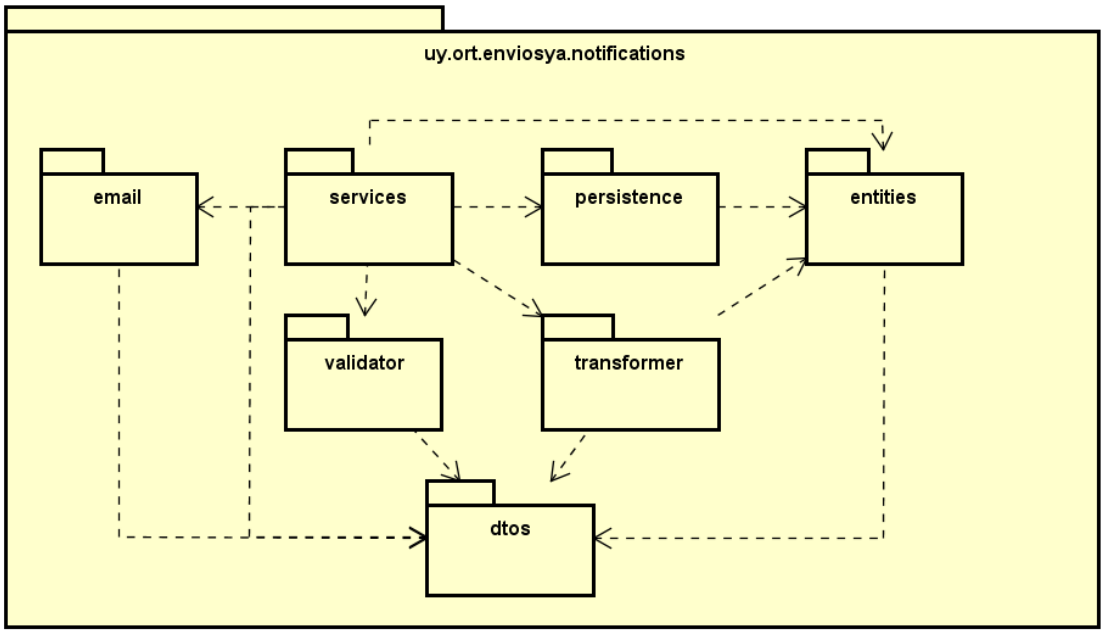
  El módulo Cache busca disminuir el tiempo que le lleva al sistema procesar un nuevo envío para aquellos usuarios que hayan tenido actividad por más de 15 días, como se pide en el RNF10. No es el único mecanismo utilizado, ya que en este cache solo figuran los usuarios que han enviado o a los que se les ha enviado un paquete en los últimos 15 días, pero un usuario puede haber estado activo de otras formas. Es utilizado por Services y Messages, para realizar las validaciones pedidas por el RF3 (verificar que el cliente y destinatario sean usuarios registrados) y obtener los datos del cadete al cual notificar.

- Modificabilidad

  Messages se encarga de construir los mensajes que se envían desde el módulo Shipments a los clientes de la aplicación. No se encarga de enviar el mensaje a los clientes, solo construye un mensaje con todos los datos necesarios y se comunica con el sub sistema Notifications para que este si lo envié a los clientes. De esta forma la única responsabilidad del sub sistema Notifications es ser el intermediario encargado de implementar mecanismos de comunicación.

3.1.2.5 Notifications

Representación primaria



Catálogo de elementos

El catálogo de elementos es idéntico al presentado en la vista de descomposición (véase Notifications), por lo cual no será repetido en este punto.

Justificación

Nuevamente se nos presenta en el diagrama la estructura base utilizada a lo largo del sistema para el ABM de entidades, también se ve la relación entre Entities y Dtos que surge de tener los enumeradores disponibles para el resto del sistema evitando que los otros sub sistemas se acoplen a la entidades de Notifications y repetir código. Como esto ya fue explicado en detalle en las secciones anteriores pasaremos a explicar las dependencias con el módulo Email.

- Modificabilidad

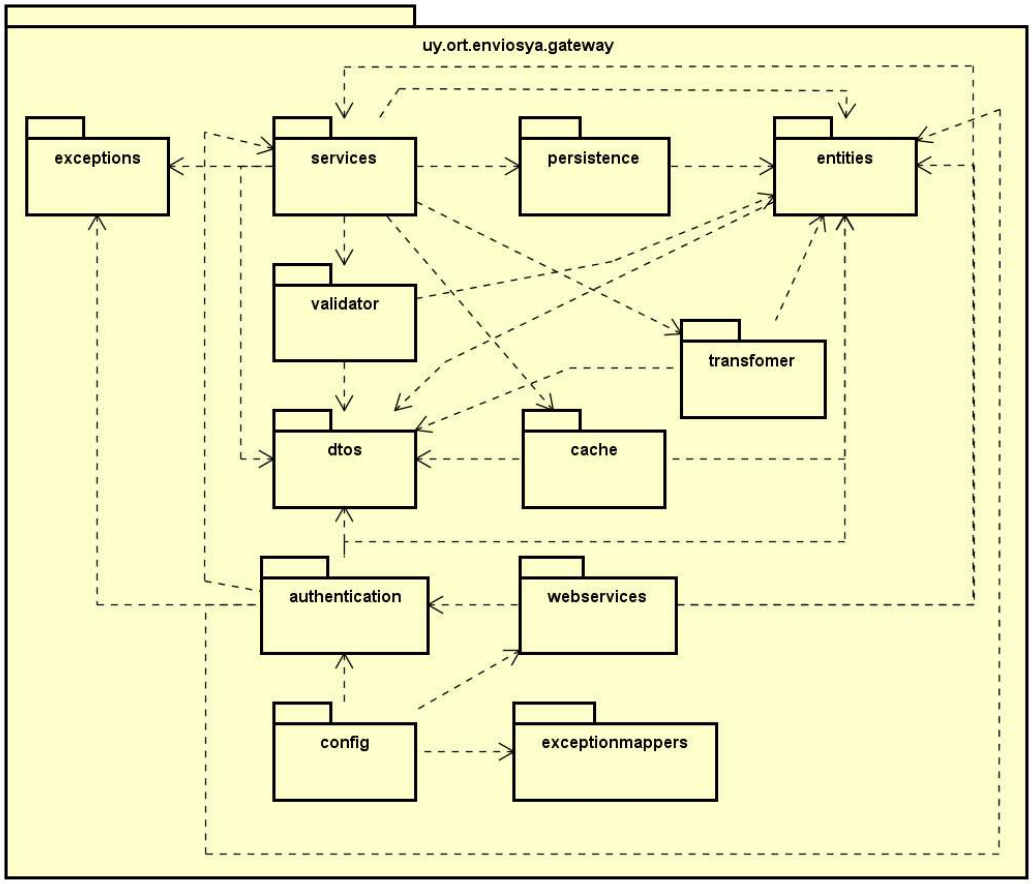
  El proceso de enviar un email es iniciado desde Services (véase la dirección de la relación entre Services y Email) además Email maneja únicamente DTOs por lo que no realiza modificaciones a las entidades guardadas en la base de datos. De esta forma, en el futuro, si se agregan otros medios de transporte de mensajes la lógica que nos lleva a decidir cuál o cuáles utilizar y la actualización de la base de datos para indicar si el envío fue exitoso se encontrará en Services. 4

- Disponibilidad

  Se utiliza el módulo de Persistence para registrar en una base de datos si el envío de los mensajes a sus correspondientes destinatarios fue exitoso o no, de esta forma sería posible reconstruir el envío en caso de un fallo como se pide en el RNF 5.

3.1.2.6 Gateway

Representación primaria



Catálogo de elementos

El catálogo de elementos es idéntico al presentado en la vista de descomposición (véase Gateway), por lo cual no será repetido en este punto.

Justificación

Este es el sub sistema con mayor complejidad a nivel de dependencias entre módulos, pero a pesar de ello es posible identificar la estructura utilizada para la gestión de una entidad del sistema que contiene campos enumerables como lo es la sesión de un usuario.

El proceso de autenticación se analizara en detalle con un diagrama de comportamiento y es el que lleva a la existencia de la mayoría de las dependencias entre los módulos que no refieren a la gestión de entidades. Por lo que no realizaremos un mayor análisis en esta sección.

3.2 Vistas de Componentes y conectores

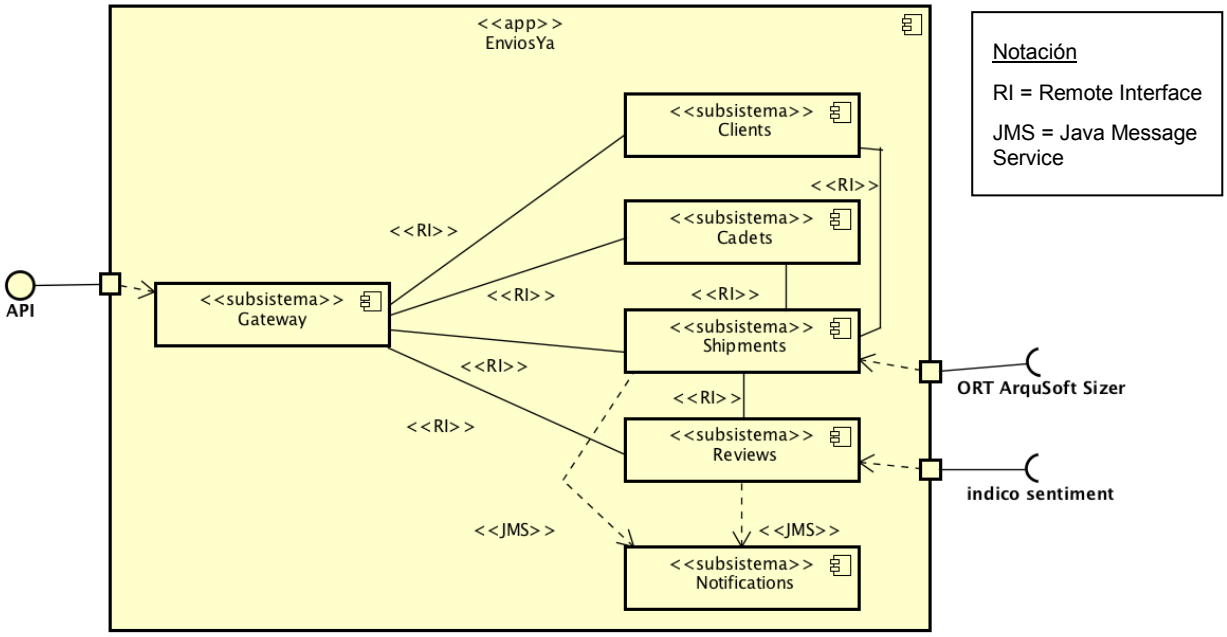
En la presente sección se explicará al lector el funcionamiento del sistema en tiempo de ejecución mediante el uso de vistas de componentes y conectores.

Para mostrar el sistema con mayor claridad utilizamos un diagrama de contexto y uno por cada subsistema que existe en nuestra solución. Cada uno tiene un catálogo de elementos e interfaces asociado. Las interfaces locales no fueron incluidas por considerar que no aportan información relevante.

Además las interfaces requeridas que son provistas por otros subsistemas no estarán incluidas en el listado de interfaces del subsistema que las requiere sino únicamente en el que las provee.

3.2.1 Diagrama de contexto

Representación primaria



Catálogo de elementos

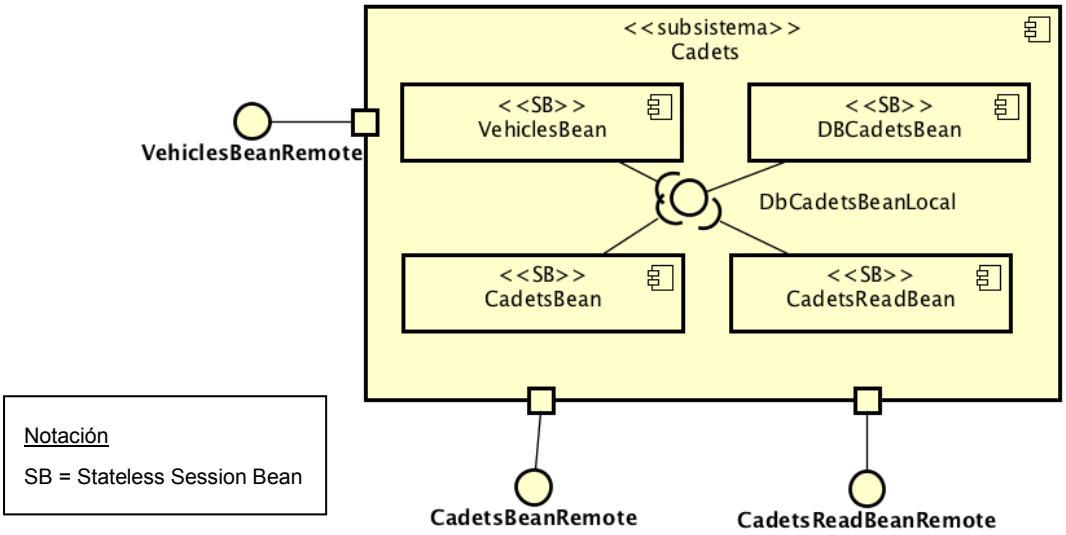
Componentes	Tipo	Descripción
Gateway	Subsistema	Expone una API REST para la aplicación EnviosYa y maneja las sesiones de los usuarios.
Clients	Subsistema	Alta, baja y modificación de clientes y alta de sus métodos de pago.
Cadets	Subsistema	Alta, baja y modificación de cadetes y sus vehículos.
Shipments	Subsistema	Manejo de envíos.
Reviews	Subsistema	Manejo de reseñas.
Notifications	Subsistema	Procesamiento de mensajes.

Interfaces

Interfaz	Descripción
API	API REST que permite accede a las funcionalidades expuestas por la aplicación para el manejo de envíos, clientes, cadetes reseñas.
ORT ArqSoft Sizer	API externa que consume la aplicación para el cálculo las dimensiones de un paquete a partir de la imagen del mismo.
Indico Sentiment	API externa que consume el sistema para determinar si un comentario es positivo, negativo y neutro.

3.2.2 Cadets

Representación primaria



Catálogo de elementos

Componente	Tipo	Descripción
DbCadetsBean	SB	Se encarga de la comunicación con la base de datos a través de JPA para los cadetes.
CadetsBean	SB	Realiza el alta, baja y modificación de un cadete.
CadetsReadBean	SB	Realiza acciones de lectura sobre los cadetes como buscar cadetes por identificador o email. También permite obtenerlo los cuatro cadetes más cercanos y validar que exista un cadete con un identificador dado.
VehiclesBean	SB	Realiza el alta, baja y modificación de los vehículos de los cadetes.

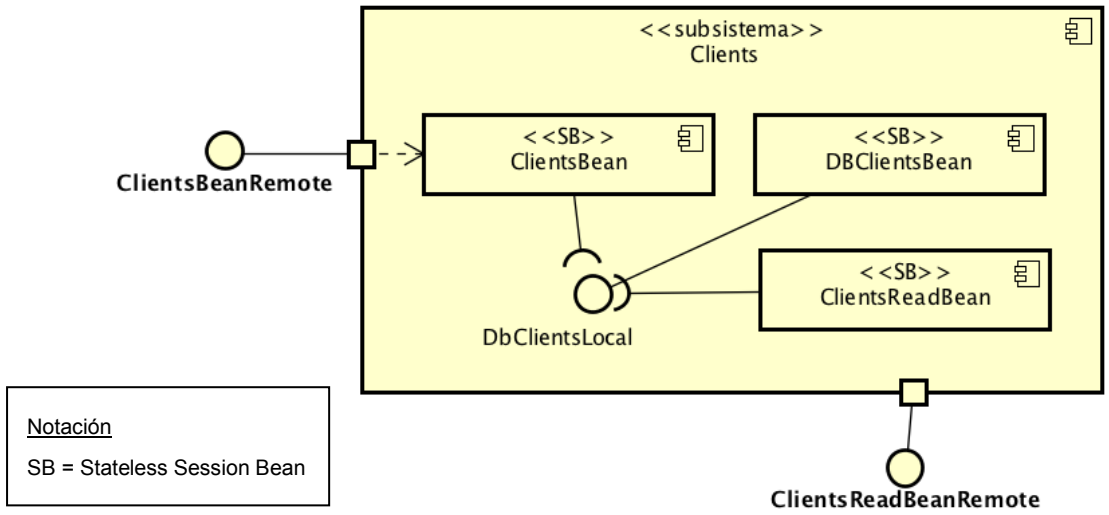
Interfaces

Interfaz	Descripción
CadetsBeanRemote	Provee una interfaz remota que es utilizada por el recurso de los cadetes (CadetsResource) para realizar el alta, baja y modificación de los mismos.

CadetsReadBeanRemote	Provee una interfaz remota que es utilizada por el recurso de los cadetes (CadetsResource) para obtener cadetes dados distintos criterios.
VehiclesBeanRemote	Provee una interfaz remota utilizada por el CadetsResource para realizar el alta, baja y modificación de los mismos.

3.2.3 Clients

Representación primaria



Catálogo de elementos

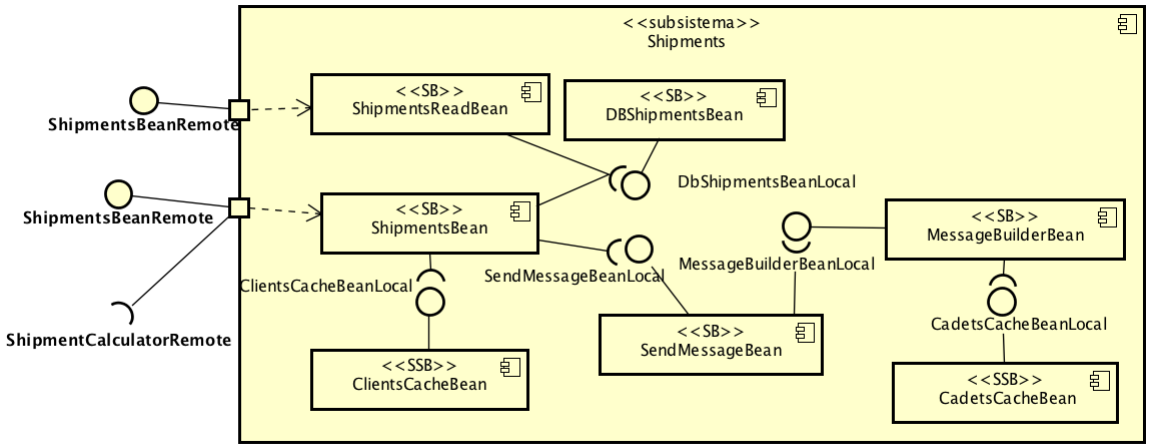
Componente	Tipo	Descripción
DbClientsBean	SB	Se encarga de la comunicación con la base de datos a través de JPA para los clientes.
ClientBean	SB	Realiza el alta, baja y modificación de los clientes.
ClientsReadBean	SB	Implementa las funcionalidades de buscar clientes por identificador o email. También permite obtener un cliente dado su identificador.

Interfaces

Interfaz	Descripción
ClientsBeanRemote	Provee una interfaz remota que es utilizada por el recurso de los clientes (ClientsResource) para realizar el alta, baja y modificación de los mismos.
ClientsReadBeanRemote	Provee una interfaz remota para obtener clientes dados distintos criterios.

3.2.4 Shipments

Representación primaria



Notación

SB = Stateless Session Bean

SSB = Singleton Session Bean

Catálogo de elementos

Componente	Tipo	Descripción
DbShipmentsBean	SB	Se encarga de la comunicación con la base de datos a través de JPA para los envíos.
ShipmentsReadBean	SB	Devuelve uno o varios pedidos según distintos criterios de consulta.
ShipmentsBean	SB	Sus principales funcionalidades son agregar pedidos, registrar la recepción de los mismos y calcular su costo.
ClientsCacheBean	SSB	Implementa un cache en memoria que contiene una lista de clientes para poder accederla de forma menos costosa.
CadetsCacheBean	SSB	Implementa un cache en memoria que contiene una lista de cadetes para poder accederla de manera menos costosa.
MessageBuilderBean	SB	Se encarga de crear los mensajes que serán enviados a los cadetes, cuando se les asigne un envío, y a los clientes para que recuerden realizar la reseña.
SendMessageBean	SB	Se comunica con el subsistema Notifications para enviar los mensajes.

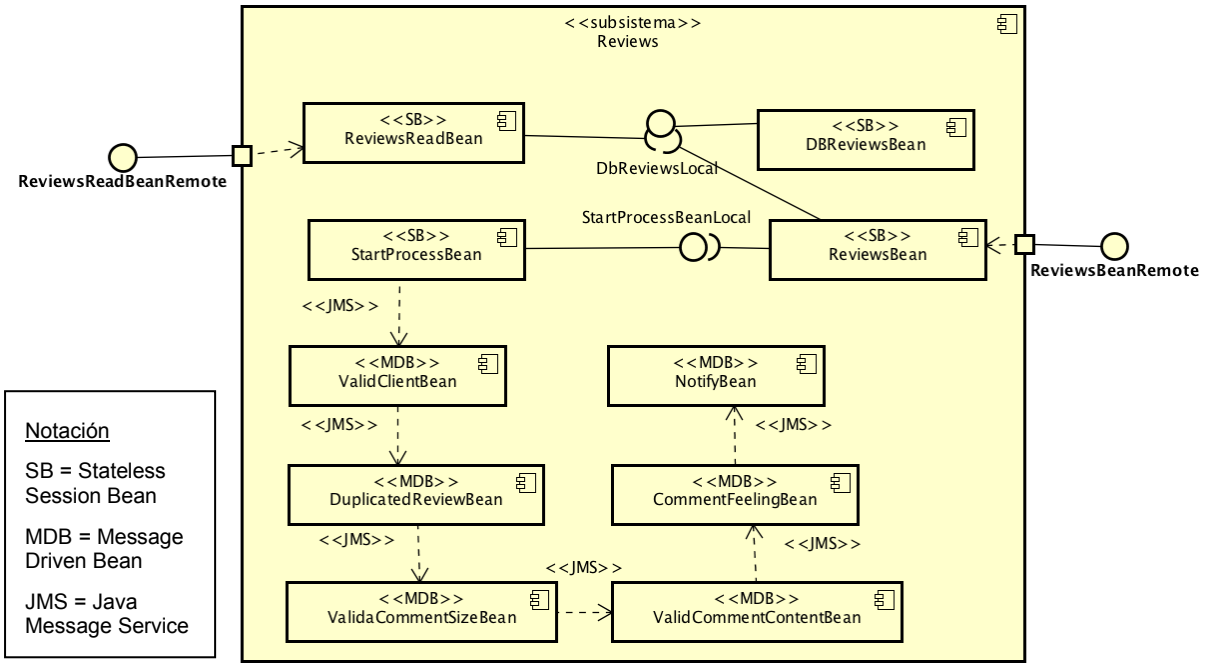
Interfaces

Interfaz	Descripción
ShipmentsBeanRemote	Provee una interfaz remota que es utilizada por el recurso de los envíos (ShipmentsResource) para acceder a la lógica de negocios para esta entidad.

ShipmentsReadBeanRemote	Provee una interfaz remota que es utilizada por el recurso de los envíos (ShipmentsResource) para obtener envíos dados distintos criterios.
ShipmentsCalculatorRemote	Usa una interfaz para acceder a las funcionalidades del cálculo del costo del envío.

3.2.5 Reviews

Representación primaria



Catálogo de elementos

Componente	Tipo	Descripción
ReviewsReadBean	SB	Devuelve uno o varias reseñas según distintos criterios.
ReviewsBean	SB	Permite agregar una reseña, iniciando el proceso de validación en pasos, y cambiar el estado de una reseña a rechazada.
DbReviewsBean	SB	Se encarga de la comunicación con la base de datos a través de JPA para las reseñas.
StartProcessBean	SB	Comienza el proceso de validación de una reseña.
ValidateCommentSizeBean	MDB	Valida que la cantidad de palabras incluidas en el comentario de la reseña sea mayor a la mínima definida por el sistema. Para ello lee este valor desde un archivo de configuración
DuplicateReviewBean	MDB	Valida que no exista una reseña para el envío, para ello consulta la base de datos a través de DbReviewsBean.

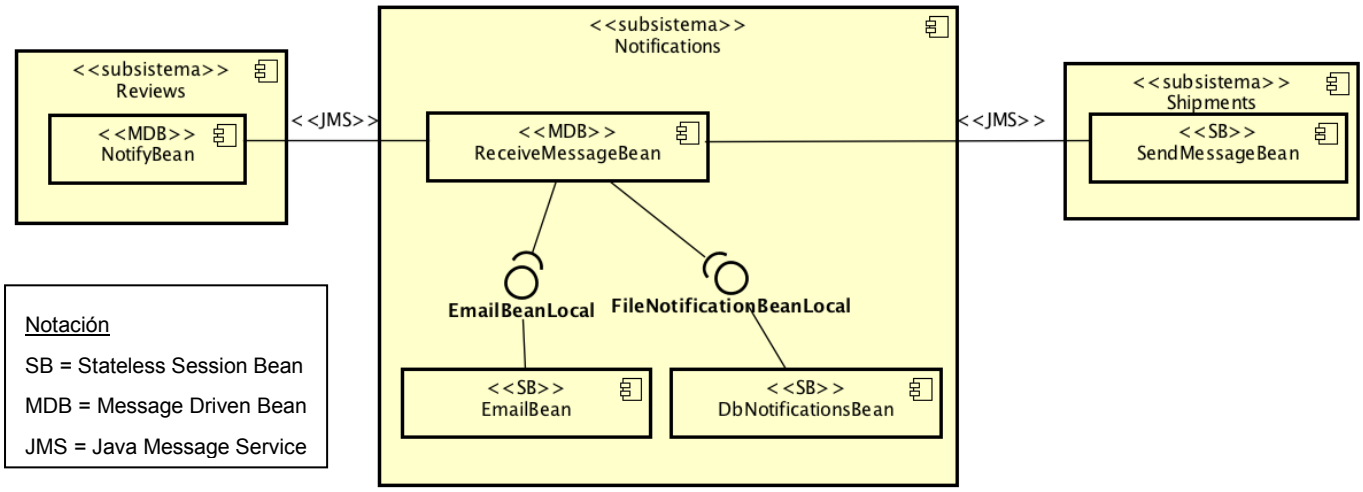
ValidClientBean	MDB	Valida que el usuario que está haciendo la reseña sea el que envió el envío o el destinatario.
ValidCommentContentBean	MDB	Valida que el comentario de la reseña no contenga palabras que se encuentran en la lista negra definida en el sistema. Para ello lee esta lista de un archivo de configuración.
CommentFeelingBean	MDB	Utiliza una API externa para analizar si el comentario de la reseña es positivo, negativo o neutro y actualiza la reseña en la base de datos agregando esta información a través de DbReviewsBean.
NotifyBean	MDB	Envía un mensaje al subsistema Notifications, a través de una cola de mensajes, para disparar procesos en otras partes del sistema.

Interfaces

Interfaz	Descripción
ReviewsBeanRemote	Provee una interfaz remota que es utilizada por el recurso de las reseñas (ReviewsResource) para realizar el alta de reseñas y modificar su estado a rechazada.
ReviewsReadBeanRemote	Provee una interfaz remota para obtener reseñas dados distintos criterios.

3.2.6 Notifications

Representación primaria



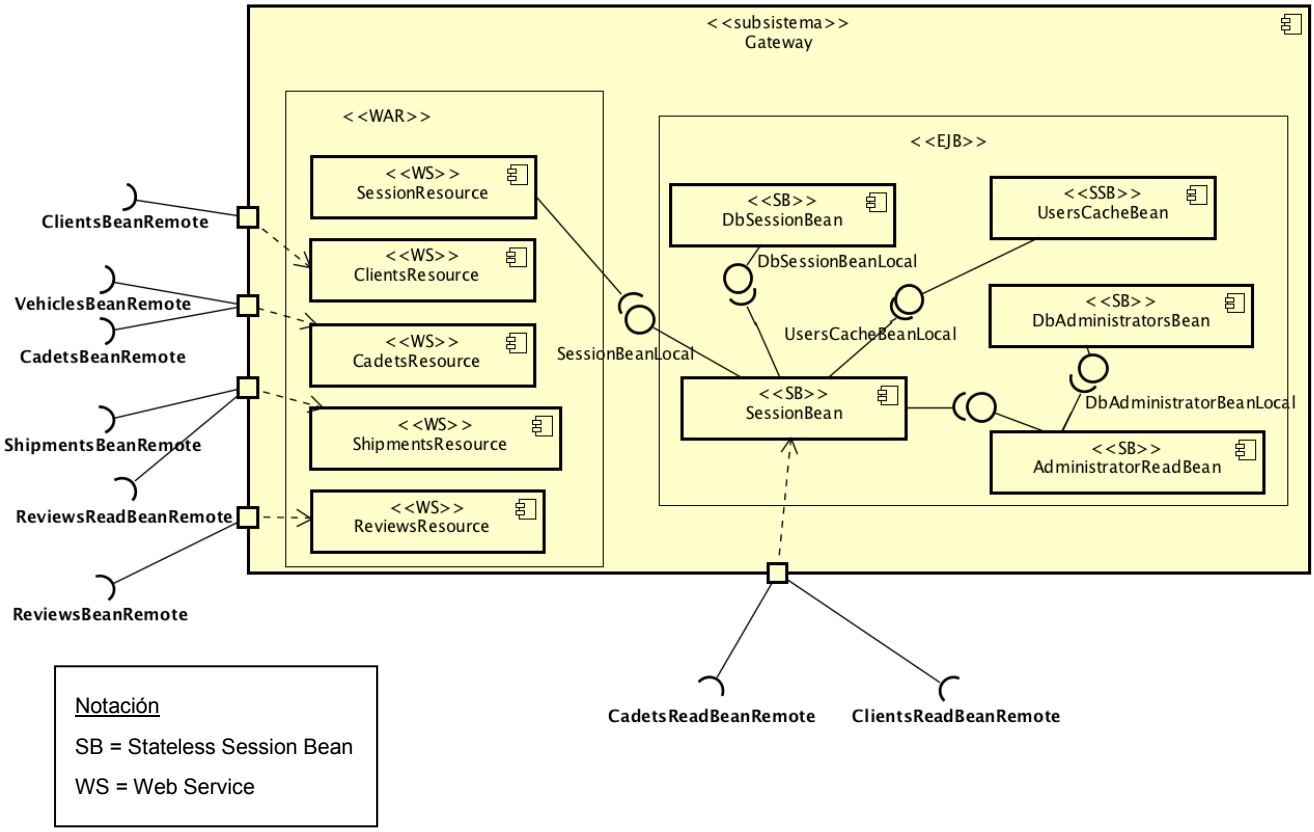
Catálogo de elementos

Componente/conector	Tipo	Descripción
ReceiveMessageBean	MDB	Recibe los mensajes de la cola de mensajes e inicia el proceso de procesamiento del mismo identificando a través de qué medio de comunicación debe ser enviado.

EmailBean	SB	Se encarga de enviar los mensajes recibidos por el subsistema a sus correspondientes destinatarios a través de una cuenta de correo electrónico. Para ello utiliza la tecnología JavaMail asociada a una dirección de correo electrónico de Gmail.
DbNotificationsBean	SB	Se encarga de la comunicación con la base de datos a través de JPA para las notificaciones.
EnviosYaQueue	JMS	Cola de mensajes encargada de recibir todos los mensajes dirigidos a este subsistema.

3.2.7 Gateway

Representación primaria



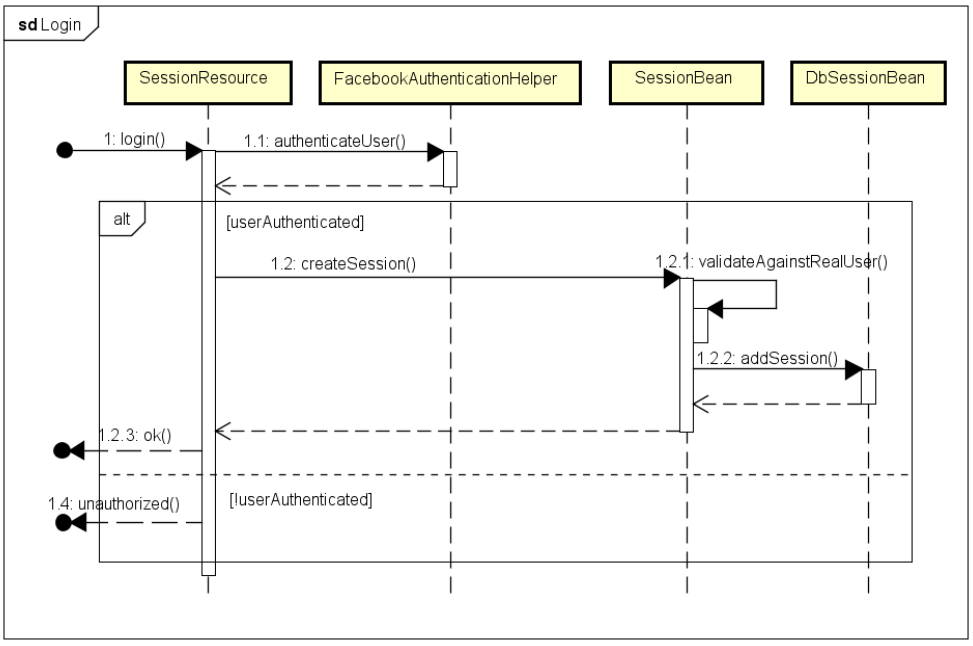
Catálogo de elementos

Componente	Tipo	Descripción
SessionResource	WS	Expone el servicio para manejar las sesiones en el sistema. El login crea una nueva sesión que tiene el email del usuario, el token que se genera por la autenticación por Facebook y el tiempo de expiración la guarda en la base de datos. El logout elimina la sesión de la base de datos.
CadetsResource	WS	Expone los servicios que permiten realizar el alta, baja y modificación de los cadetes y sus vehículos.

ClientsResource	WS	Expone los servicios para el manejo de clientes que permite agregar y modificar clientes.
ReviewsResource	WS	Expone los servicios de las reseñas de los cadetes. Permite obtener las reseñas para un cadete dado, por rango de fechas o estado de la reseña. También permite a los administradores rechazar una reseña.
DbSessionBean	SB	Se encarga de la comunicación con la base de datos a través de JPA para las sesiones.
SessionBean	SB	Se encarga de agregar una nueva sesión y eliminarla cuando se hace el logout. También permite identificar el rol de un usuario para analizar si tiene permisos suficientes para realizar la funcionalidad pedida. Otro chequeo que se hace es validar contra el usuario real para verificar que los datos de la sesión tengan sentido.
DbAdministratorsBean	SB	Se encarga de la comunicación con la base de datos a través de JPA para los administradores.
AdministratorReadBean	SB	Dado un correo electrónico devuelve al administrador correspondiente si existe.
UserCacheBean	SSB	Se encarga de agregar y obtener usuarios del cache que se lleva en la memoria de este subsistema.

3.2.8 Diagramas de comportamiento

3.2.8.1 Login



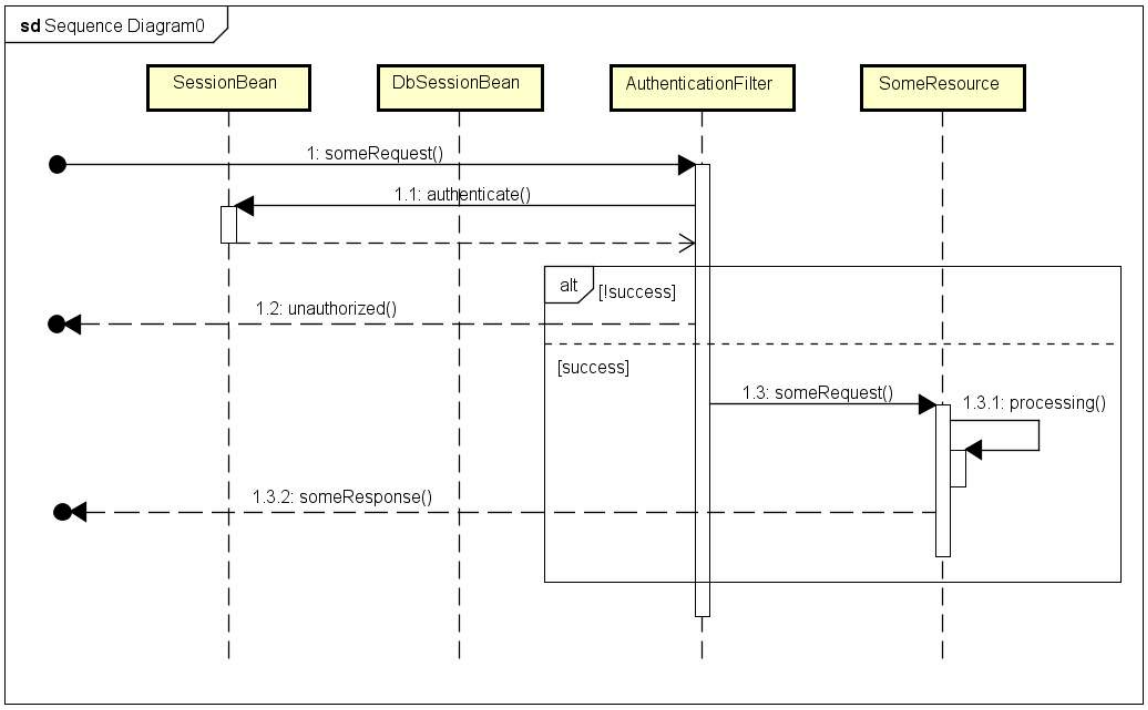
En este diagrama se muestra el proceso de login del sistema:

1. Comienza con una llamada del usuario al resource SessionBean y tiene dos posibles respuestas.
   1.1. Se valida contra la API de Facebook que el token recibido haya sido generado para nuestro sistema. Según la respuesta de la API el flujo se divide en dos.
   1.2. Condición: token válido

Se dispara el proceso de creación de la sesión para poder almacenar en la base de datos el token con su correspondiente usuario y utilizarlo posteriormente para autenticarlo. Para ello verificamos que el usuario esté registrado en el sistema.
Finalmente se responde al usuario.

1.3. Condición: token inválido
Se responde al usuario indicando que no tiene autorización.

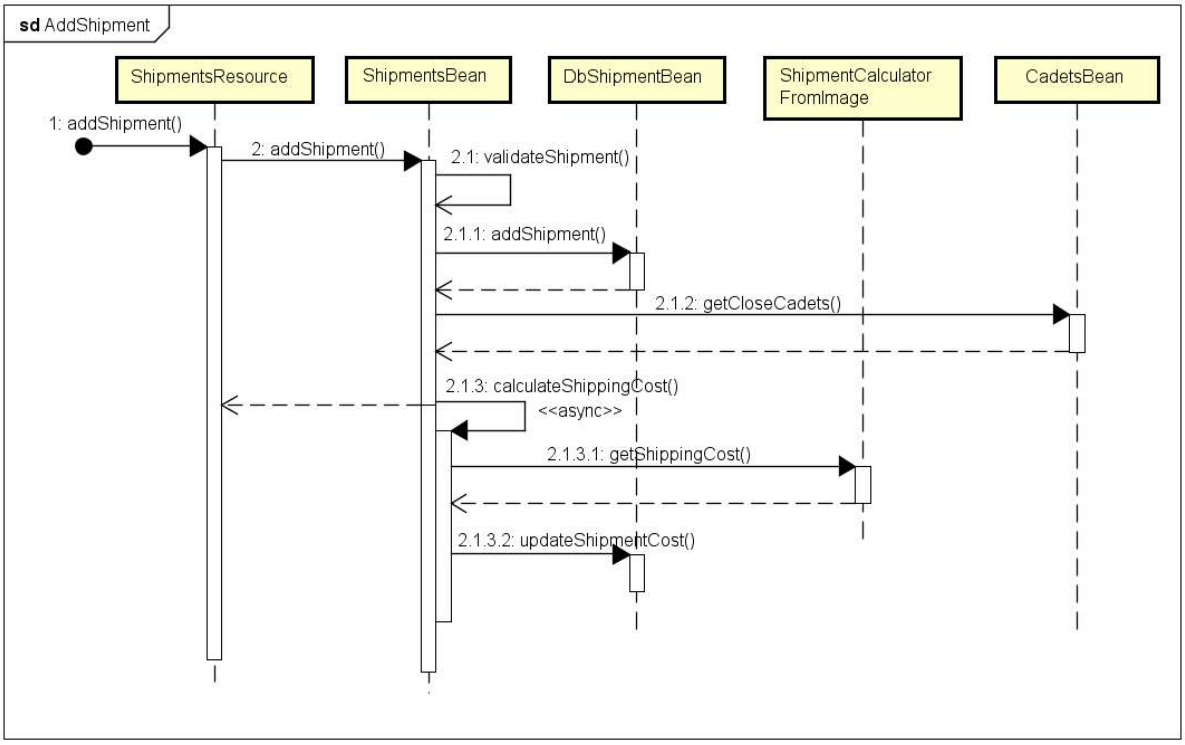
3.2.8.2 Autenticación en una consulta



Este diagrama muestra el proceso de autenticación que se realiza en cada una de las consultas a la API, el proceso es el mismo para todos los tipos de usuarios:

1. Comienza el proceso con una llamada del usuario a alguno de los recursos. Esta llamada es interceptada por un filtro de autenticación.
   1.1. Verifica que el token recibido en el encabezado corresponda al token brindado por uno de los usuarios loggeados. Según la respuesta el flujo se divide en dos.
   1.2. Condición: token inválido
   Se responde al usuario indicando que no tiene autorización.
   1.3. Condición: token válido
   La consulta pasa el filtro y llega al recurso al que originalmente estaba dirigido que procesara el pedido y retornara una respuesta al usuario.

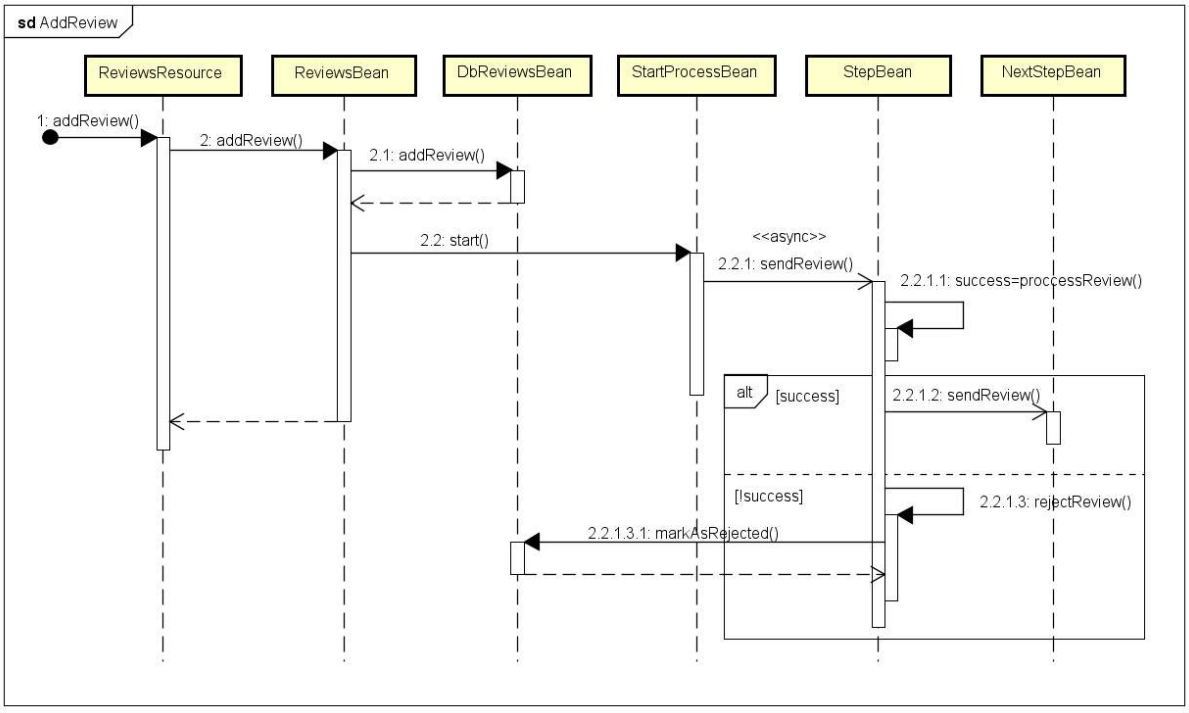
3.2.8.3 Agregar un nuevo envío



Lo interesante del diagrama anterior es notar como la respuesta es devuelta antes de que el proceso del cálculo del costo del envío termine. Para ello la llamada 2.1.3 es asincrónica y termina actualizando la base de datos con el costo del envío para que pueda ser consultado en otro momento por el usuario.

De esta forma mejoramos la performance y evitamos depender del tiempo de respuesta de una API de terceros (como la utilizada para calcular el costo actualmente).

3.2.8.4 Agregar una reseña



La complejidad del proceso de creación de una reseña comienza en el paso 2.2 del diagrama, en este caso representamos dos pasos genéricos:

2.2. Se inicia el proceso, se lee el nombre de la cola de mensajes correspondiente al paso 1 del archivo de configuración.

2.2.1. Se envía la reseña a la cola de mensajes del primer paso.

2.2.1.1. El paso recibe el mensaje y lo procesa. Según el resultado de este procesamiento el flujo se divide en dos.

2.2.1.2. Condición: procesamiento exitoso

Se envía la reseña a la cola de mensajes que lee el siguiente paso, el nombre también se obtiene de un archivo de configuración, y se repiten los pasos desde el 2.2.1.1.

2.2.1.3. Condición: procesamiento fallido o problema con la reseña

Se actualiza el estado de la reseña a *REJECTED* y se detiene el proceso. También es posible continuar al paso siguiente con la reseña en este estado.

3.2.9 Justificación

Performance

Utilizamos distintas técnicas para favorecer la performance de nuestra solución. En primer lugar, decidimos utilizar interfaces para la comunicación entre distintos subsistemas. Tuvimos en cuenta la posibilidad de elegir comunicarnos a través de Web Services, pero preferimos elegir la herramienta provista por Java EE para obtener una leve mejora en el tiempo de comunicación entre los subsistemas. Al hacer esto nos acoplamos a la tecnología, por lo que nuestros subsistemas solo podrán comunicarse con otros desarrollados en Java EE.

Para optimizar los tiempos de respuesta creación de solicitud de un envío (RNF9) implementamos clientes en el subsistema, está desarrollado de forma de que almacene la información únicamente por 15 días. Además, implementamos un cache de cadetes que nos permite disminuir los tiempos de creación de un mensaje de aviso al cadete al momento de ser asignado al envío. De esta manera, en muchos casos, evitamos consultar esta información realizando llamados a los subsistemas Clients y Cadets mejorando así la performance.

En la creación de una solicitud de envío consumimos una API externa para calcular el costo del paquete, la llamada a la misma se hace de forma asincrónica ya que no podemos controlar ese tiempo de respuesta. (RNF9)

También utilizamos de mecanismos de asincronismo para realizar el proceso de validación de una reseña, de esta forma cuando el usuario crea una reseña podemos responderle rápidamente y dejar que el proceso complejo se ejecute paralelo. (RNF2)

Modificabilidad

Para que el proceso de creación de una reseña sea modificable nos basamos en el patrón Pipes and Filters, donde cada paso que hay que validar de la reseña es un filter y estos se comunican entre sí mediante colas de mensajes, que serían los pipes. Además, se extrajo la configuración que refiere a la identificación del paso siguiente a un archivo de configuración (indicando el nombre de la cola de mensajes del cual el mismo va a leer). De esta manera se logró que agregar un nuevo filtro al proceso de creación sea simple, solamente se tiene que agregar un Message Driven Bean que implemente el procesamiento requerido, y luego, mediante un archivo de configuración insertarlo en la cadena donde corresponda. También, si lo que se desea es eliminar un paso o modificar el orden solamente debemos modificar el archivo de configuración. Aprovechando la existencia del archivo de configuración colocamos allí variables como la lista negra de palabras y la cantidad de palabras mínimas permitiendo que los pasos en sí mismos sean configurables fácilmente. (RNF1)

Para lograr la independencia de la lectura y la escritura, dentro de los servicios de cada proyecto, separamos las acciones de lectura y escritura en distintos beans. En los diagramas se puede ver como los subsistemas exponen una interfaz de lectura y otra de escritura. (RNF6)

Para poder modificar el mecanismo de cálculo de costo del envío con el menor impacto posible en los módulos del sistema y con el redespliegue del menor número de componentes posible utilizamos la táctica de diferir enlaces. Se creó una interfaz remota y un componente nuevo que implementa la interfaz remota realizando la llamada a la API indicada. De esta manera para cambiar el mecanismo del cálculo simplemente se tiene modificar este componente que implementa la interfaz remota y desplegarlo. (RNF10)

Disponibilidad

Para gestionar los errores y fallas del sistema se implementó un mecanismo log junto con excepciones propias. Al utilizar excepciones propias los errores que se envían son sencillos de entender y descriptivos, además se devuelven como un objeto Json de forma de que puedan ser manejables programáticamente por los consumidores de la API.

Para el mecanismo de logging se utilizaron las librerías SLF4J y Log4j. La primera funciona como una fachada y nos permite modificar el mecanismo real de log modificando únicamente la librería de log, de esta forma podríamos pasar a utilizar fácilmente java.util.logging. Elegimos utilizar Log4j porque nos permite loggear diferentes mensajes en varios archivos de configuración de manera simultánea, esto nos fue útil para dividir el logging de auditoria de los errores que se producen en el subsistema Gateway Cada entrada guarda la clase donde ocurrió el error junto con un mensaje descriptivo y, de ser necesario, la excepción generada. (RNF4)

Al tener un log detallado y descriptivo de los errores del sistema, es fácil poder luego recuperarse de las fallas, ya que el estado del sistema se puede visualizar claramente en cada momento. Además puntualmente si analizamos el proceso de validación de una reseña podemos ver que al guardar el resultado de cada uno de los pasos, acompañados del mensaje de error si corresponde, en la base de datos podríamos reconstruir este proceso en caso de falla o intervención. Algo similar nos permite la información guardada para los mensajes enviados por el subsistema Notifications, entre los datos que incluimos guardamos si el mensaje fue enviado de manera exitosa o fallo de forma de poder reenviarlo en caso de ser necesario. (RNF5)

Seguridad

Para favorecer la seguridad del sistema decidimos usar autenticación a través de Facebook que utiliza OAuth 2.0. Para poder logearse al sistema el usuario debe enviar un Access Token obtenido a través de una llamada a la API de Facebook, el sistema realiza otra llamada a una API de Facebook para validar que el token recibido corresponda a uno generado para nuestro sistema. Para no comprometer la performance de las demás funcionalidades del sistema, una vez que el usuario ya está logeado al sistema, el token se guarda encriptado en la base de datos y en un cache de usuarios, donde estará disponible por 15 días, y se utiliza para validar que el usuario tenga los permisos necesarios para realizar el resto de las llamadas. (RNF3)

Para proteger los datos de los clientes se encriptó la información más sensible de los mismos. El token de autenticación se guarda en hasheado en la base de datos, de manera de que si se compromete la seguridad de la misma y se puede acceder al token el atacante no podrá utilizarlo para realizar llamadas a la API ya que sería doblemente hasheado. La información del método de pago del cliente también se guarda encriptada de manera de proteger la confidencialidad del cliente y al igual que en el caso anterior, un ataque en la base de datos no comprometa la información de por ejemplo, la tarjeta de crédito del cliente. En este caso la información se encripta de forma tal que pueda ser desencriptada por el sistema si es necesario utilizarla (RNF7)

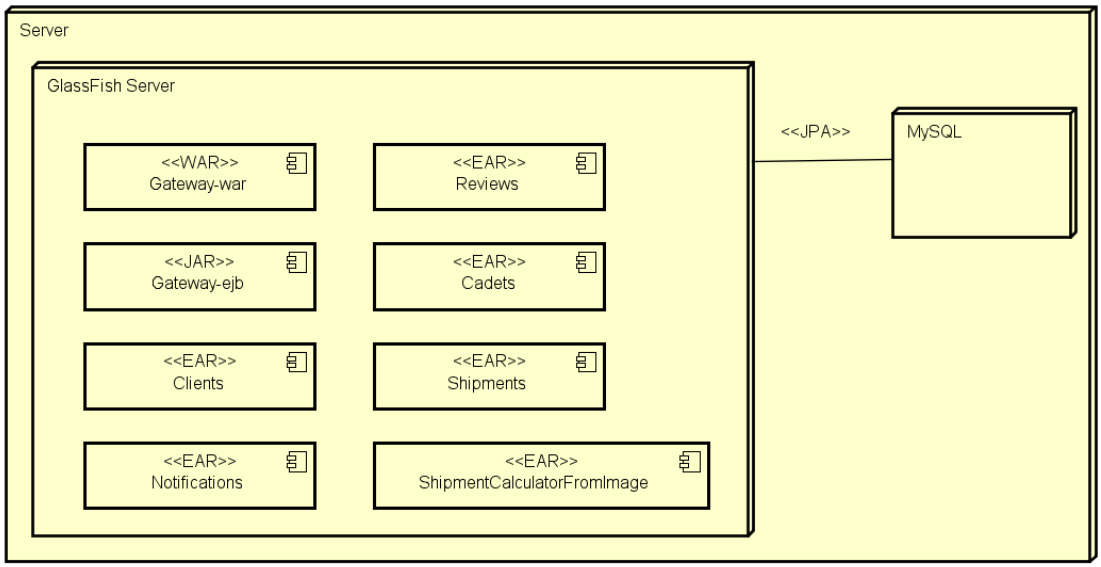
De forma de tener información de auditoria, es decir, los accesos autorizados y no autorizados del sistema, se guarda un log independiente. Cada intento de acceso del sistema crea una nueva entrada en el log. Si el acceso fue autorizado se guarda el tipo de usuario y su correo electrónico, mientras que si fue identificado un acceso no autorizado se registra esta información. (RNF8)

Para proteger la información enviada al sistema desde el cliente se generó un certificado SSL que se agregó al servidor y obliga a que las comunicaciones a través de la API se realicen utilizando HTTPS, si el usuario realiza una consulta a HTTP la misma se redirige a la versión segura.

3.3 Vistas de Asignación

3.3.1 Vista de Despliegue

Representación primaria



Catálogo de elementos

Ya que actualmente el sistema está desplegado en un único servidor conteniendo un servidor GlassFish, con el contenido de todos los artefactos desplegados en un único nodo, y MySQL consideramos que no es necesario incluir una explicación detallada en esta sección.

Únicamente mencionar que ara poder ejecutar la aplicación es necesario que el servidor de despliegue cumpla con los siguientes requisitos:

- GlassFish Server 4.1.1 o superior
- Java EE 7 SDK

Justificación

Con la arquitectura actual, y sin la necesidad de realizar ningún cambio, el artefacto ShipmentCalculatorFromImage podría ser desplegado en otro servidor sin que esto afectara el funcionamiento. Hacer esto podría enlentecer la comunicación con el mismo pero, ya que esta se realiza de manera asincrónica, esto no repercutiría en la respuesta a los clientes del sistema. También es posible realizar una nueva implementación del cálculo del costo del envío y desplegarla de manera independiente modificando así en tiempo de ejecución y sin afectar a los otros subsistemas el mecanismo. (RNF10)

Dado que el intercambio de información es stateless (es decir, no se guarda información de un cliente entre llamadas, se envía todo la información en cada uno de los mensajes intercambiados), es posible escalar horizontalmente a más servidores para balancear la carga.

Para ello deben tenerse en cuenta algunas consideraciones para manejar correctamente el cache y el acceso a las bases de datos. Tendría que analizarse el funcionamiento del cache que ahora se realiza en memoria y debería pasar a ser centralizado si deseamos mantener las mejoras en la performance que se obtienen de utilizarlo. Además, deberíamos separar las bases de datos y hacerlas disponibles para todos los servidores. También deberían configurarse correctamente las colas de mensajes utilizadas para que puedan enviarse mensajes entre servidores.

Los únicos dos artefactos que deberían mantenerse siempre en un mismo servidor (o en varios pero siempre en conjunto) son Gateway-war y Gateway-ejb porque la comunicación entre ellos se realiza a través de interfaces locales.