Universidad ORT Uruguay
Facultad de Ingeniería
Bernard Wand Polak

Arquitectura de Software
Obligatorio

Te Pago Ya

Nicolás Hernández – 198765

Docentes:
Andrés Calviño
Mathias Fonseca

1- Introducción

A continuación en el siguiente documento se describirán las decisiones técnicas, de diseño y decisiones de arquitectura tomadas para el desarrollo del sistema *Te Pago Ya,* mostrándolas, explicándolas y proveyendo también una justificación de porque fueron tomadas a cabo y como las mismas llevaron a la solución actual del problema. Para esto se hará un análisis exhaustivo del sistema anterior y los requerimientos solicitados e identificados por el equipo de desarrollo.

1.1- Propósito

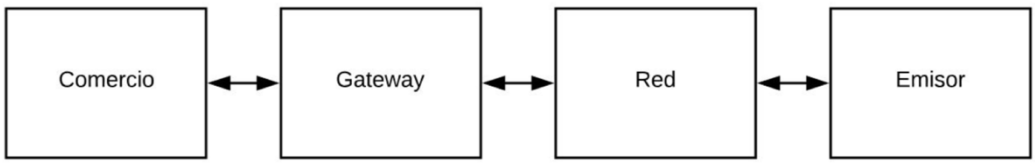
El propósito de este documento entonces será de proveer una especificación para el sistema de pagos *Te Pago Ya*

2- Antecedentes

2.1 - Propósito del sistema

Te Pago Ya es un sistema de pagos el cual viene a tratar de resolver el complicado universo de los pagos online, surge como solución a la urgente necesidad de simplificar lo mas posible la integración de los diferentes actores que se encuentran en este universo (comercios, gateways, emisores de tarjetas, etc.)
Te Pago Ya entonces será una herramienta que actuará como intermediario, resolviendo la mayor parte de los escenarios de integración como pueden ser los diferentes formatos de fechas y números, formatos de representación de datos, etc.

Actualmente las comunicaciones entre los diferentes actores funciona de forma directa, el cliente proveyó el siguiente diagrama a modo de ejemplo:

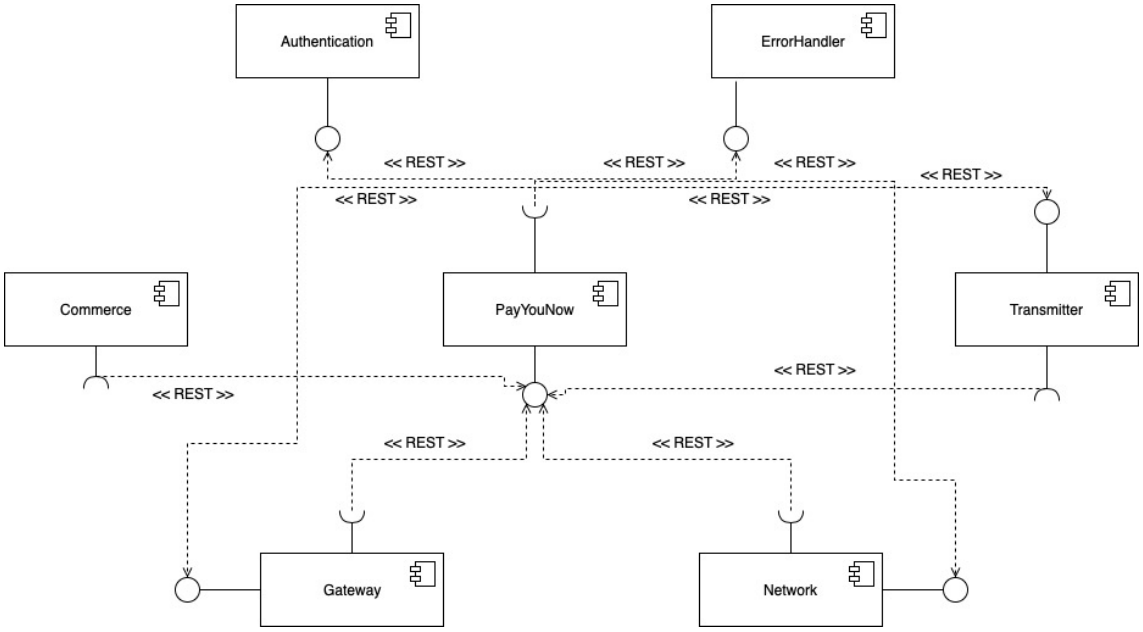


El emisor es el encargado de emitir las tarjetas de crédito
Cuando un comercio quiere realizar ventas con las tarjetas este se comunica con el Gateway, este a su vez necesita conectarse con la Red la cual hace control de fraudes, etc. y esta se comunica con el Emisor para realizar los últimos controles.
Al finalizar el Emisor los controles, este responde a la Red, la cual responde al Gateway y el Gateway al comercio.

Como se puede observar el escenario es bastante complejo y si además se añade que existen muy pocos estándares y reglas de cómo deben comunicarse estos actores se hace muy complicado manejar la integración y mantenibilidad de los sistemas.

Luego de analizar todo este escenario surgió la posibilidad de *Te Pago Ya* que viene a aparecer como una solución en el complicado universo de los pagos online.
Este se encargara de las comunicaciones entre los sistemas. También se podrán registrar nuevos sistemas de forma dinámica solamente proveyendo una interface para poder acceder a sus funcionalidades.

Se plantea como posible solución el siguiente modelo de conexiones entre los distintos módulos del sistema:



Como se puede observar, *Te Pago Ya* actuara como intermediario entre las comunicaciones. Para obtener este modelo fue puesto en consideración el patrón SOA. *Te Pago Ya* permitirá registrar nuevos servicios por medio de un endpoint REST, lo único necesario para hacerlo es enviarle toda la información del servicio nuevo a registrarse, es necesario especificar nombre, tipo de conexión, dirección de localización y las operaciones que expondrá.

Te Pago Ya dará la posibilidad también consumir los servicios especificados siempre y cuando se cumplan determinadas condiciones como son que:

- Se especifique el nombre del proveedor
- Se especifique el nombre de la operación y los parámetros y argumentos necesarios.
- El consumidor se encuentre autenticado debidamente y cuente con los permisos necesarios para realizar la operación.

A partir de esto respetando la correcta aplicación del patrón SOA Te Pago Ya validara, enriquecerá y transmitirá la información que pase atreves de él, usando además mecanismos de soporte como son la autenticación y el manejo de errores.

2.2 - Requerimientos significativos de Arquitectura

Como ya mencionamos anteriormente, los actores de este sistema son Commerce, PayYouNow, Network, Transmitter, Gateway, ErrorHandler y Authentication.

2.2.1 - Resumen de requerimientos funcionales

Id Requerimiento	Descripción	Actor
REQ-1	Compra en comercio, un consumidor podrá realizar una transacción de compra en un comercio, este en función de la categoría del producto a comprar deberá seleccionar el Gateway que procesara la compra y enviarle la solicitud	Commerce, PayYouNow, Authentication, ErrorHandler
REQ-2	Compra en Gateway, el Gateway procesa el pago y decide a que red enviarlo dependiendo de la tarjeta.	Gateway, PayYouNow, Authentication, ErrorHandler
REQ-3	Compra en red, realiza controles de prevención de fraudes contra un limite determinado y envía la transacción hacia el emisor para el paso final. Se debe poder alterar el limite para el control de fraude en tiempo de ejecución.	Network, PayYouNow, Authentication, ErrorHandler
REQ-4	Compra en emisor, debe chequear que la tarjeta sea valida (que no este vencida, ni bloqueada, ni denunciada), ejecutar el algoritmo de luhn y que tenga saldo suficiente para realizar la transacción.	Transmitter, PayYouNow, Authentication, ErrorHandler
REQ-5	Devolución, deberá ser posible la devolución de cualquier transacción que no haya superado el limite de días definido en el emisor.	Commerce, Network, Gateway, Transmitter, PayYouNow, Authentication, ErrorHandler
REQ-6	El usuario puede realizar un desconocimiento de compra en su emisor. El emisor debe informar al comercio para que realice los tramites administrativos.	Commerce, Transmitter, PayYouNow, Authentication, ErrorHandler
REQ-7	El comercio le solicita diariamente al Gateway la transacción de cierre de lotes en donde se informa al máximo detalle los movimientos de dinero. Debe responder en un tiempo promedio menor a 50ms bajo cargas de 5000 solicitudes por minuto	Commerce, Gateway, PayYouNow, Authentication, ErrorHandler

2.2.2 - Resumen de requerimientos no funcionales

Id Requerimiento	Nombre	Atributo de Calidad	Descripción
RNF01	Modificación de las distintas aplicaciones	Modificabilidad	Una aplicación debe poder ser modificada en un futuro sin afectar al resto
RNF02	Integración de aplicaciones	Interoperabilidad	Desarrollar TePagoYa/PayYouNow para resolver la integración e interoperabilidad. Para cada prestador de servicios se debe poder: • Registrar una interfaz de servicio para acceder a la funcionalidad. • Especificar propiedades sobra la interfaz. • Localizar la interfaz de un servicio registrado. • Facilitar la invocación de las funcionalidades
RNF03	Gestión de fallas y errores	Testeabilidad	El sistema debe proveer suficiente información que permita conocer el detalle de las tareas que se realizan.
RNF04	Recuperación de fallas	Disponibilidad	En caso de ocurrir una falla o errores es imprescindible que el sistema provea toda la información.
RNF05	Autenticación	Seguridad	Autenticación, para toda aplicación prestado de servicios que se registre en TePagoYa/PayYouNow

Id Requerimiento	Nombre	Atributo de Calidad	Descripción
			se requiere que todas sus operaciones publicadas sean invocadas únicamente a través de TePagoYa/PayYouNow
RNF06	Estándares de seguridad	Seguridad	Toda aplicación debe cumplir con los estándares de seguridad de PCI al máximo nivel posible, pudiendo demostrar y justificar a cada momento el nivel de cumplimiento.
RNF07	Tiempo de respuesta	Performance	Se debe minimizar el tiempo de respuesta de las peticiones entre aplicaciones

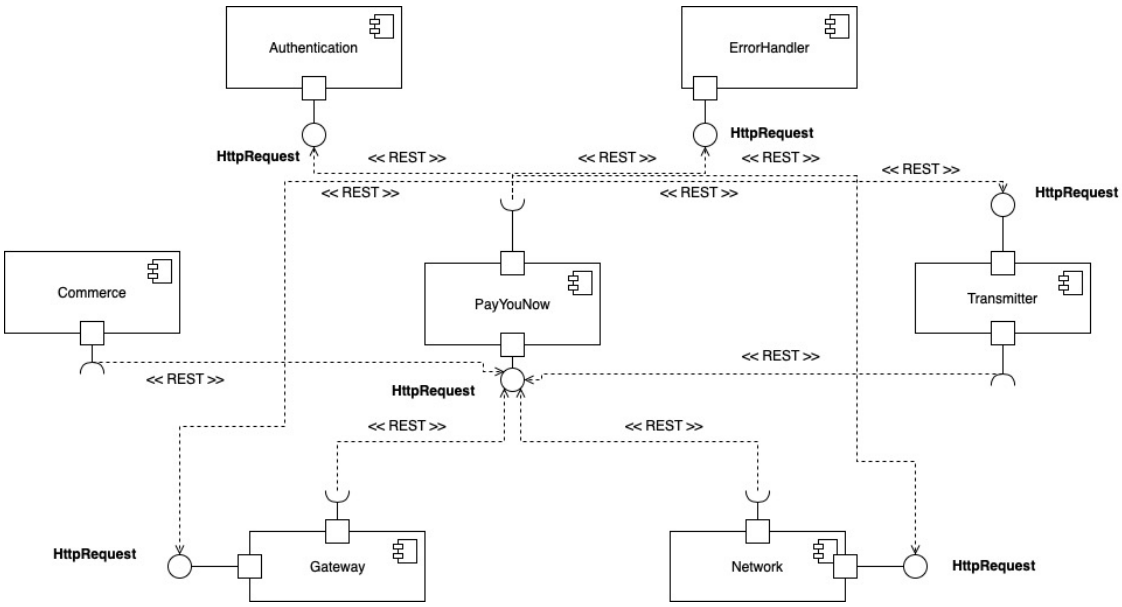
3- Documentación de la arquitectura

En esta sección se presentara la arquitectura del sistema desarrollado como solución a la problemática descripta en las secciones anteriores. Esta representación se llevara a cabo mediante la utilización de diferentes vistas del sistema, las cuales se irán detallando y explicando de manera de poder demostrar los requerimientos que fueron implementados en el sistemas y los atributos de calidad que los mismos contemplan.

Comenzaremos presentando el sistema en general mediante una vista de componentes y conectores, esta vista nos permitirá demostrar el funcionamiento del sistema desarrollador y dar una base de cómo esta compuesto el mismo, finalizaremos este apartado presentando una vista de asignación para demostrar como se distribuye el sistema en general.
Luego de haber entendido el funcionamiento general del sistema se pasara el foco a los módulos del mismo, para lograr darle el detalle y la dimensión adecuada se utilizara la vista de módulos, donde usaremos una vista de usos y una vista de layers las cuales nos ayudara a mostrar las decisiones de diseño que se fueron tomando en cada uno y además se agregara una vista de componentes y conectores.

3.1 – Vista de componentes y conectores general.

3.1.1 – Representación primaria



3.1.2 – Catalogo de elementos

Elemento	Tipo	Responsabilidades
Authentication	Express Server	Es el encargado de manejar la autenticación de los módulos, registra los módulos, provee tokens y los valida.
PayYouNow	Express Server	Es el encargado de la interoperabilidad entre todos los módulos y de las comunicaciones del mismo expone servicios REST en las cuales se pueden registrar servicios o consumir servicios registrados.
Commerce	Express Server	Es el encargado de iniciar el proceso de transacción, determina el Gateway con el cual se procesara la compra.
Gateway	Express Server	Es el encargado de determinar la Red a partir de la tarjeta con la que se realiza la compra, expone servicios REST para el mismo.
Transmitter	Express Server	Es el encargado de validar los últimos pasos de la transacción, que la tarjeta sea valida, que no este bloqueada, que el saldo sea suficiente, además también se encarga de emitir tarjetas. Expone servicios REST para las operaciones nombradas.
Network	Express Server	Es el encargado de llevar a cabo un control de fraude en la tarjeta de la transacción y también de derivar la misma al emisor

Elemento	Tipo	Responsabilidades
		correspondiente, también lo hace mediante un servicio REST.
ErrorHandler	Express Server	Encargado de llevar una trazabilidad de las operaciones realizadas, también provee información en caso de defectos y fallas. Expone un servicio REST que usa un servicio en la WEB para control de fallas.

3.1.3 – Decisiones de diseño

Decisiones generales

Metodología
Se opto por usar como mecanismo de comunicación REST y a partir de este usar los beneficios del mismo como lo es la comunicación por JSON. Gracias a esta decisión ya logramos desacoplar la implementación de los diferentes módulos con sus operaciones provistas, ya que nada mas para acceder a la misma solo necesitamos la uri indicada y los parámetros requeridos, además JSON también nos aporta otra capa mas de desacople donde no juega realmente la tecnología que este detrás de la interfaz REST.

Descomposición por Feature
En una primera descomposición, los módulos definen funcionalidades (Commerce, Network, Transmitter, etc.).
La descomposición por funcionalidad prioriza el Principio de Reúso Común ya que es muy probable que por ejemplo el módulo Commerce quiera reusarse en otro sistema, alcanzando con los módulos contenidos en él y prescindiendo de los demás.

A su vez, cada módulo se descompone según capas como se justificará más adelante, pero en resumen, los módulos que proveen una interfaz REST contendrán un módulo que agrupe todo lo que involucre a y solo a la interfaz con este protocolo, de manera que pueda ser fácilmente intercambiable por otra.

Variables de entorno para configuración
No se creo sub-módulos “Config” en los Módulos sino que se aprovecho de una bondad de NodeJs como son las variables de entorno, estas son las encargadas de hacer la ”configuración” de los módulos, contando cada modulo con un archivo .env que contiene las variables de entorno que se utilizan en el proyecto, siendo un método totalmente desacoplado.
Si bien hoy en día es un mock y retorna configuraciones hardcodeadas, se deberían proveer los mecanismos para acceder a archivos de configuración, que podrían en el futuro reusarse o cambiarse.
Para el manejo interno de estas variables de entorno y poder crear estos archivos .env y que los mismos sean cargados en los proyectos nos valimos de la librería ***dotenv***

Express servers

Todos los módulos funcionan como Express-Server, se eligió Express.js como tecnología para realizar todos los módulos debido que ya se definió que los módulos iban a ser APIS REST siendo Express una herramienta extremadamente potente para realizar APIS además obviamente también se considero una gran cantidad de pros que traía utilizar esta tecnología, siendo algunos los siguientes:

- Fácil de aprender y usar debido a que no usa una sintaxis exótica o una estructura complicada, además de ser 100% JavaScript.
- Altamente flexible para todos los diferentes tipos requerimientos de las APIS
- Soportan middlewares los cuales son fáciles de crear por uno mismo y agregarlos para una gran cantidad de escenarios diferentes, como puede ser por ejemplo que todas las rutas de una REST API tengan seguridad y se pueda validar o invalidar el acceso de una request a las mismas.
- Fácil integración con Third-Parties
- Bien documentado, además de la gran cantidad de tutoriales y ejemplos, lo que es muy bueno para las condiciones del equipo de desarrollo
- Es fácil de configurar y liviano, no afectando la performance del sistema

MongoDB

Como se utiliza JSON y APIS REST se decidió por usar MongoDB como base de datos no relacional y aprovechar su fácil integración con Express.js y que también usa BSON (Binary Serialized JSON) para el almacenamiento de datos. También aprovechamos sus otras ventajas como por ejemplo:

- Schema-less: Como se parsean los datos a JSON, cada aplicación puede mandar diferentes campos comparados a otros y este Schema-less nos ayuda a guardarlos fácilmente
- Insertar y hacer updates es muy performante.
- Fácil instalación, replicación y escalamiento

PayYouNow

El modulo PayYouNow esta capacitado para registrar aplicaciones que pueden ser prestadores de servicios así como también consumidoras de los mismos los cuales se van a comunicar con el en run-time. Para llevar a las operaciones descriptas como pueden ser el registro así como también el consumo de algunas operaciones particulares expone un servicio REST, dicho servicio mapea las peticiones hacia los controladores los cuales se encargan de determinar cual es la operación solicitada y llevarla a cabo. Gracias a REST logramos también conseguir una cierta capacidad de carga importante, pudiendo soportar una gran cantidad de peticiones por minuto, siendo algo crucial en el universo de las compras online.

Manejo de errores

El modulo ErrorHandler se llevo a cabo debido a la necesidad de manejar un cierto mecanismo de trazabilidad de las operaciones llevadas a cabo y también de tener información que sea de utilidad en caso de ocurrir defectos o fallas y lograr aumentar la disponibilidad así como también la testeabilidad.

La decisión de hacer un modulo totalmente separado para esta función surge de lograr un desacoplamiento total de los demás y que los módulos no tuvieran que encargarse cada uno de esta función sino que pudieran delegarla en otro diferente, este modulo solo expone un servicio REST el cual tiene dos POST, uno para loggers y el otro para errores, de esta manera cualquier modulo puede consumir su servicio y hacer uso de el, logrando así también una extensibilidad en la solución ya que si en algún caso se llega a agregar algún actor a la solución, este puede usar fácilmente el servicio REST de manejo de errores.

Autenticación

El modulo Authentication se llevo a cabo para manejar la necesidad de tener ciertas credenciales entre los diferentes módulos, dar un toque de seguridad y proteger a PayYouNow de consumo de servicios indeseados así como también de posibles ataques. Para esto no quedaba otra solución mejor que sacar esta funcionalidad a una entidad superior y apartada de los diferentes módulos, igual que en el caso anterior esto nos da una gran extensibilidad ya que se pueden registrar y autenticar una cantidad virtualmente infinita de módulos.

Este modulo lleva a cabo la tarea de autenticación mediante el uso de JsonWebToken los cuales son un método de autenticación sencillo, seguro y además totalmente desacoplado pudiendo ser utilizado por módulos de todo tipo. Expone un servicio REST con las operaciones de Registrar, Autenticar y Validar.

Como extra podemos añadir que el mismo maneja encriptación en las contraseñas guardadas en la base de datos.

También se quiere añadir que debido a la complejidad y la falta de tiempo no se llevo a cabo la implementación de la política de token de refresco ya que la misma no es tan trivial, sin embargo puede ser fácilmente implementable debido a que JsonWebToken ya provee esta funcionalidad siendo fácilmente extensible la solución en futuras implementaciones.

Commerce, Gateway, Transmitter y Network

Los módulos Commerce, Gateway, Transmitter y Network son módulos los cuales funcionan al estilo de “Mocks” ya que en la realidad el equipo no va a ser el encargado de desarrollar estos diferentes módulos. Sin embargo para esta solución se opto por la decisión de que por mas que fueran mocks lograr que esto fuera lo mas parecido a la realidad. La comunicación entre estos 4 actores y PayYouNow se decisión que fuera REST debido a la naturaleza de la misma, que ya fue descripta en apartados superiores, usando esto se logro desacoplar totalmente los 5 actores fundamentales de la solución y lograr una comunicación sencilla, eficaz y fácilmente modificable.

Flujo comunicación

Al principio en el primer reléase como se puede ver en el repositorio de la solución la comunicación entre los módulos era diferente pensando mas que nada en temas de performance pero sacrificando mucha modificabilidad, la misma la hacia en su totalidad PayYouNow utilizando las respuestas, lo cual eliminaba muchas peticiones en la red y como esta es un recurso caro nos daba un salto interesante en performance. Sin embargo después de tener una charla con el cliente el mismo prefirió sacrificar un poco de performance y ganar en modificabilidad dejando la solución actual de hoy en día.

Como extra general podemos decir que todos los módulos cuentan archivos de configuración los cuales pueden ser fácilmente cambiados dándoles a los módulos un extra en modificabilidad.

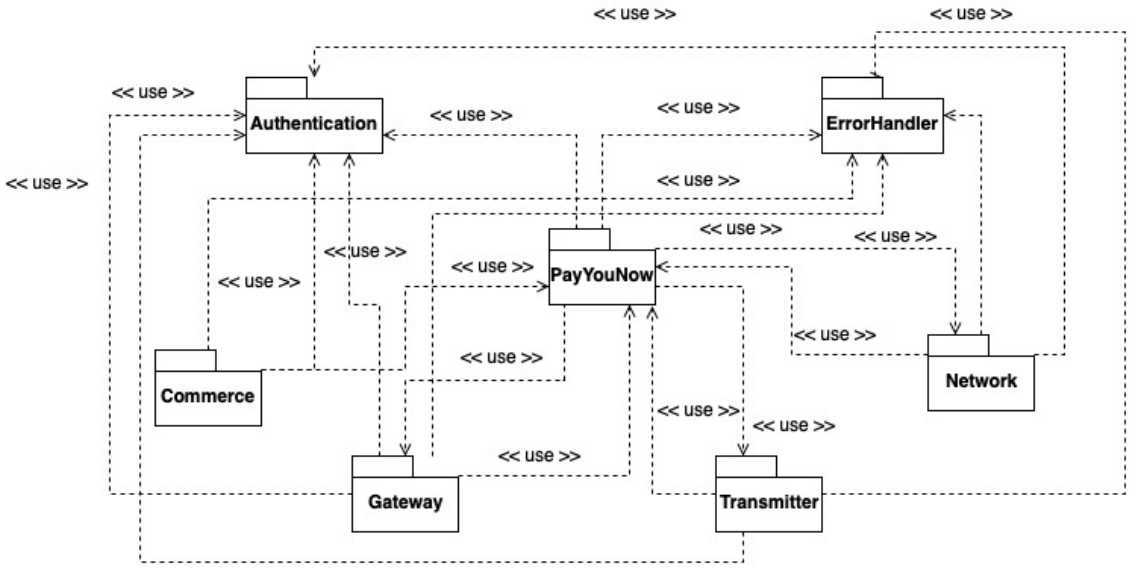
Mejoras a realizar
También en futuras instancias de desarrollo de PayYouNow se le agregaría una queue para manejar las solicitudes REST que recibe con el fin de no bloquear a sus consumidores esperando por el servicio. Lamentablemente, por asuntos de tiempo, esto no llegó a ser implementado para la presente iteración, pero sería una mejora a implementar a futuro.

Otra cosa también es el tema de la seguridad respecto a MongoDB, el equipo de desarrollo tenia varias ideas para aumentar la seguridad y no dejar que nadie inserte cosas que no deberían estar en la DB pero no se llego a implementar, igual se considera que como todos los envíos de datos a las DB pasan por ***Serializers*** y las request por ***Deserializers,*** que suceda esto es bastante improbable pero consideramos que será un gran ‘’Nice to have” en el futuro.

Debido a la falta de tiempo de los integrantes del equipo, no se llegaron a implementar muchas de las ideas para el sistema. Es por esto que se hará especial hincapié en describirlas en este documento, para plasmar las implementaciones a futuro que no fueron posibles ejecutar.

3.2 – Vista de módulos general.

3.2.1 – Representación primaria

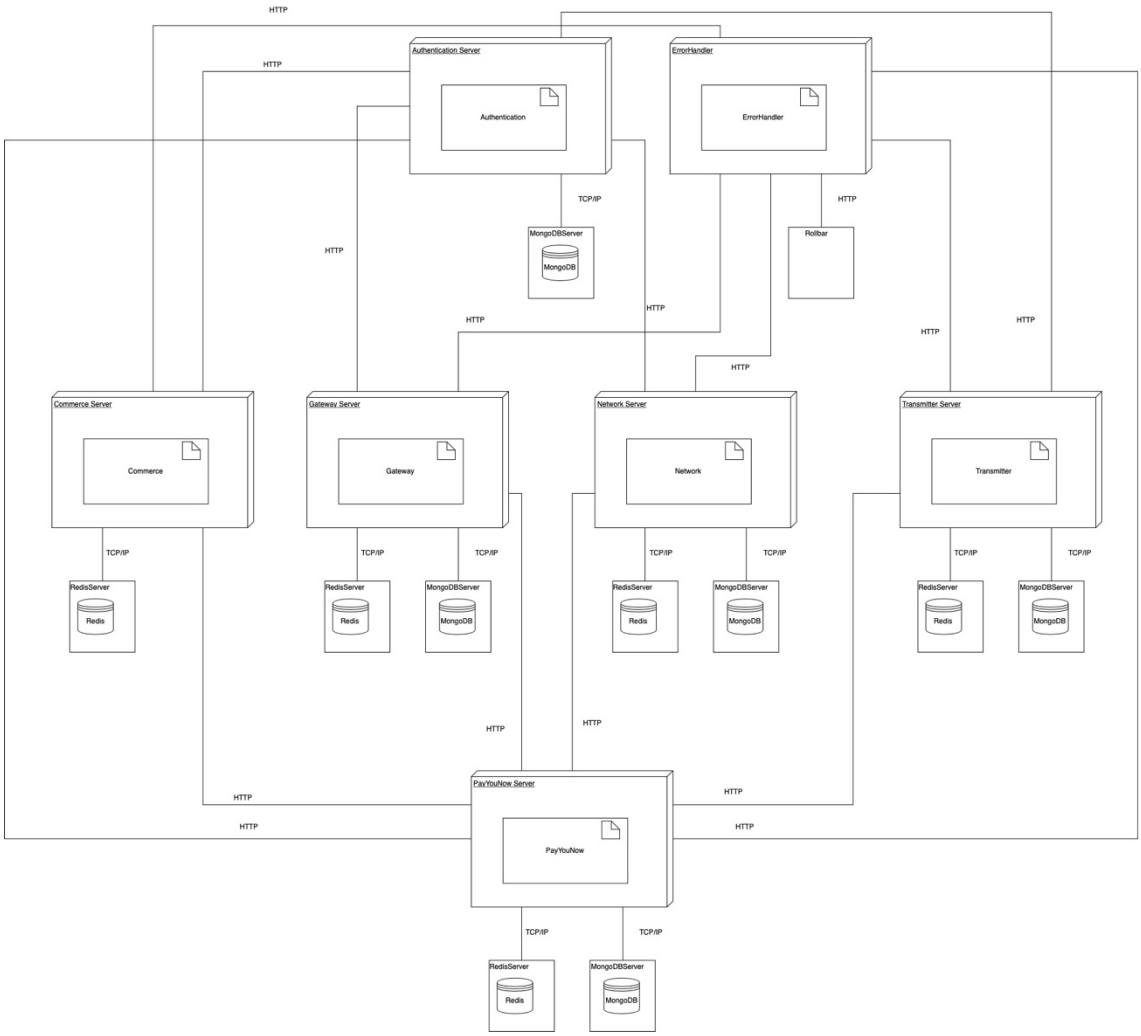


3.2.2 – Catalogo elementos

Elemento	Tipo	Responsabilidades
Authentication	Express Server	Es el encargado de manejar la autenticación de los módulos, registra los módulos, provee tokens y los valida.
PayYouNow	Express Server	Es el encargado de la interoperabilidad entre todos los módulos y de las comunicaciones del mismo expone servicios REST en las cuales se pueden registrar servicios o consumir servicios registrados.
Commerce	Express Server	Es el encargado de iniciar el proceso de transacción, determina el Gateway con el cual se procesara la compra.
Gateway	Express Server	Es el encargado de determinar la Red a partir de la tarjeta con la que se realiza la compra, expone servicios REST para el mismo.
Transmitter	Express Server	Es el encargado de validar los últimos pasos de la transacción, que la tarjeta sea valida, que no este bloqueada, que el saldo sea suficiente, además también se encarga de emitir tarjetas. Expone servicios REST para las operaciones nombradas.
Network	Express Server	Es el encargado de llevar a cabo un control de fraude en la tarjeta de la transacción y también de derivar la misma al emisor correspondiente, también lo hace mediante un servicio REST.
ErrorHandler	Express Server	Encargado de llevar una trazabilidad de las operaciones realizadas, también provee información en caso de defectos y fallas. Expone un servicio REST que usa un servicio en la WEB para control de fallas.

3.3 – Vista de asignación general

3.3.1 – Representación primaria



3.3.2 – Decisiones de diseño

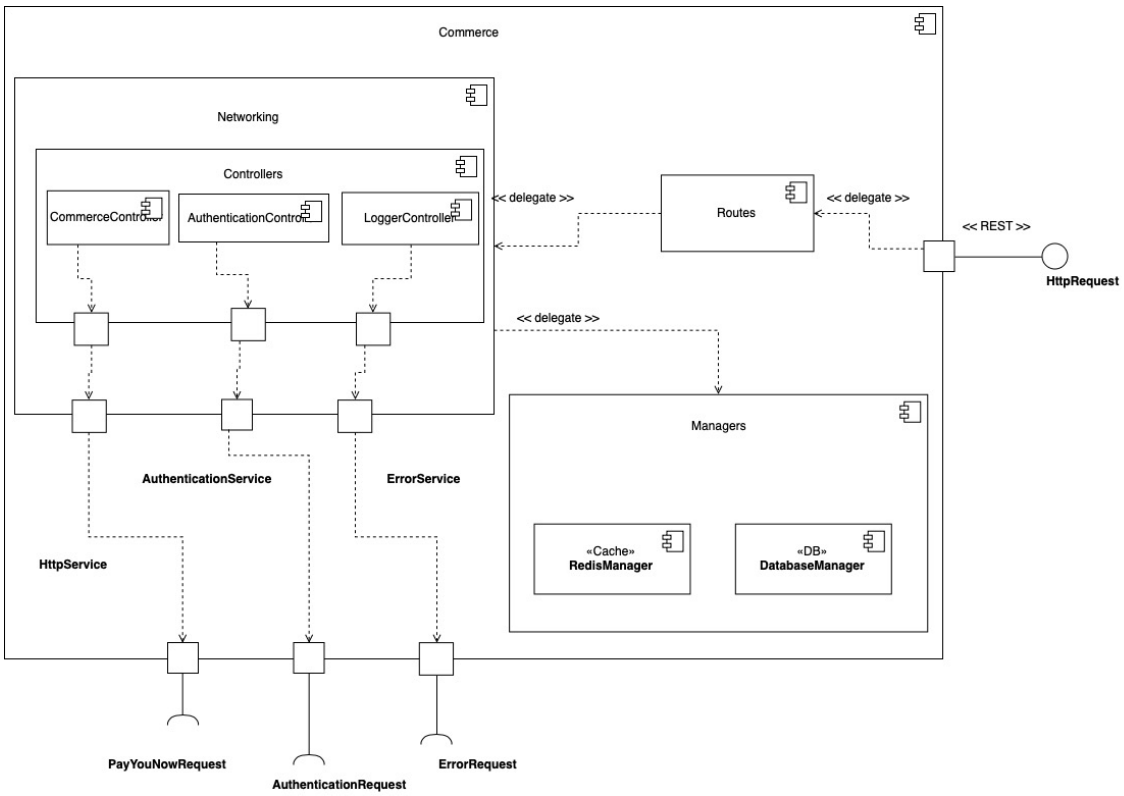
En la vista presentada se puede observar cómo se despliega el sistema que representa la solución actualmente. Como se puede observar las aplicaciones están bien separadas entre si contando cada una con un servidor aparte y que además cada bloque cuenta con su servidor de Redis y su servidor de MongoDB de manera tener mas escalabilidad en los mismos a futuro, además podemos decir que cada modulo es independiente de otros módulos ya que tienen sus propio manejo de datos.

También se puede observar la particularidad de que el manejo de errores se hace a través de la herramienta online Rollbar, pudiéndose modificar cuando el cliente lo desee. Este diseño puede ser modificado si es de interés del cliente separar aun mas las aplicaciones en diferentes servidores debido a que todas las aplicaciones son independientes, también se pueden llegar a juntar si el cliente lo considera oportuno. A su vez, este será el despliegue presentado en la defensa ya que se realizará en la misma máquina de los desarrolladores

3.4 – Commerce

3.4.1 – Vista de componentes y conectores.

3.4.1.1 – Representación primaria



3.4.1.2 – Interfaces

Elemento	Responsabilidades
HttpRequest*	Expone una “interfaz” REST, con las operaciones de commerce hacia el mundo exterior.
PayYouNowRequest	Solicita la “interfaz” REST a payYouNow para poder comunicarse con el Gateway y demás a modo de efectuar las transacciones
AuthenticationRequest	Solicita la “interfaz” REST a Authentication para poder registrarse, autenticarse y validar los tokens.
ErrorRequest	Solicita la “interfaz” REST a ErrorHandler para poder llevar una trazabilidad de las operaciones realizadas y para tener un mecanismo de control de defectos y fallas

3.4.1.3 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Controllers	Contiene todos los controllers de la aplicación.
CommerceController	Contiene toda la lógica asociada al negocio, se encarga de ajustar los datos y de mandarlos de forma correcta hacia el HttpService de forma de realizar las comunicaciones
AuthenticationController	Encargado de todo lo referido a la autenticación y su comunicación con el servicio.
LoggerController	Encargado de todo lo referido al manejo de errores y su comunicación con el servicio.
ErrorService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo ErrorHandler.
HttpService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo PayYouNow, también puede encargarse de otro tipo de comunicación http.
AuthenticationService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo AuthenticationService.
Managers	Contiene otra capa que en este apartado vendría a representar la persistencia y el manejo de cache, contiene los “manejadores” que se encargan de estas tareas descriptas.
RedisManager	Encargado de manejar el cache por medio de una database de Redis.
DatabaseManager	Encargado de manejar toda la lógica y de realizar las operaciones referidas a la base de datos.

3.4.1.4 – Decisiones de diseño

Las decisiones tomadas en este paquete siempre fueron pensadas de forma de aumentar la modificabilidad, dando así un extra para poder mantener y extender en el futuro el modulo.
Se trato de mantener una buena separación de responsabilidades con quizás tres capas bien marcadas, y también se trato de tener en cuenta los principios de paquetes respetando en gran medida los principios de:

- Principio de dependencias a cíclicas
- Principio de dependencias estables

Manteniendo estos principios en mente se disminuye el acoplamiento entre paquetes y nos potencia aun mas la modificabilidad.

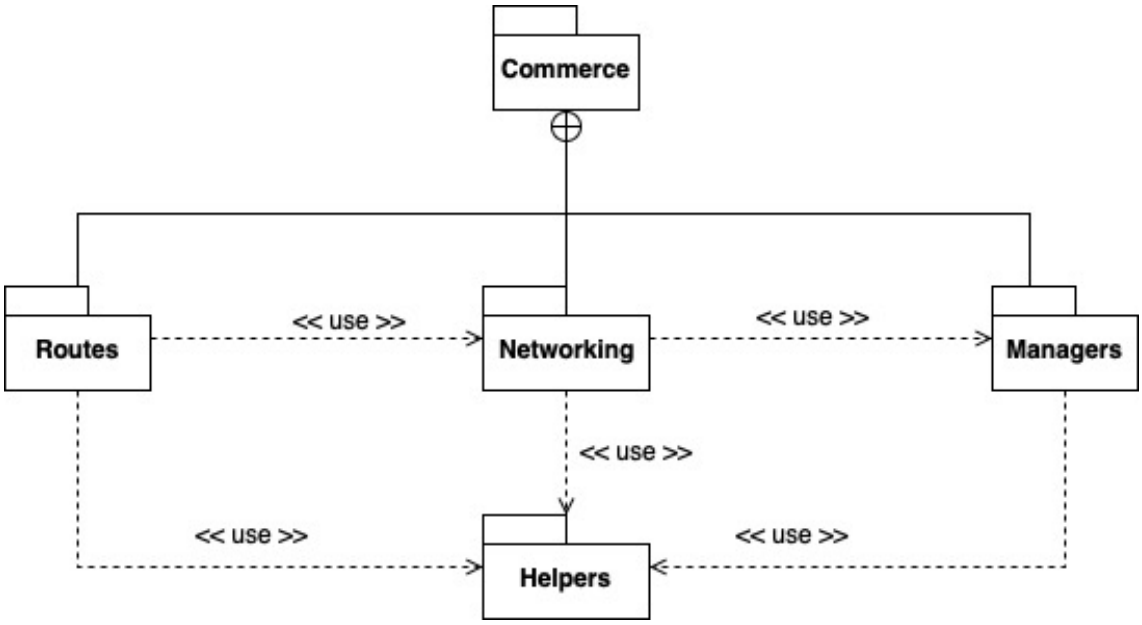
Cada una de estas capas como se puede ver cada capa superior, delega responsabilidades hacia las capas inferiores, no esta acoplada a la siguiente y se podría cambiar la implementación sin afectar demasiado al proyecto.

El temas de performance se trato de usar Redis para el manejo de la memoria cache ya que sus prestaciones en tema de velocidad son muy superiores a otras opciones.
Además esta memoria cache se uso para almacenar el token de autenticación del modulo dando un toque de seguridad al modulo.

3.4.2 – Vista de módulos.

3.4.2.1 – Vista de usos

3.4.2.1.1 – Representación primaria



3.4.2.1.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable

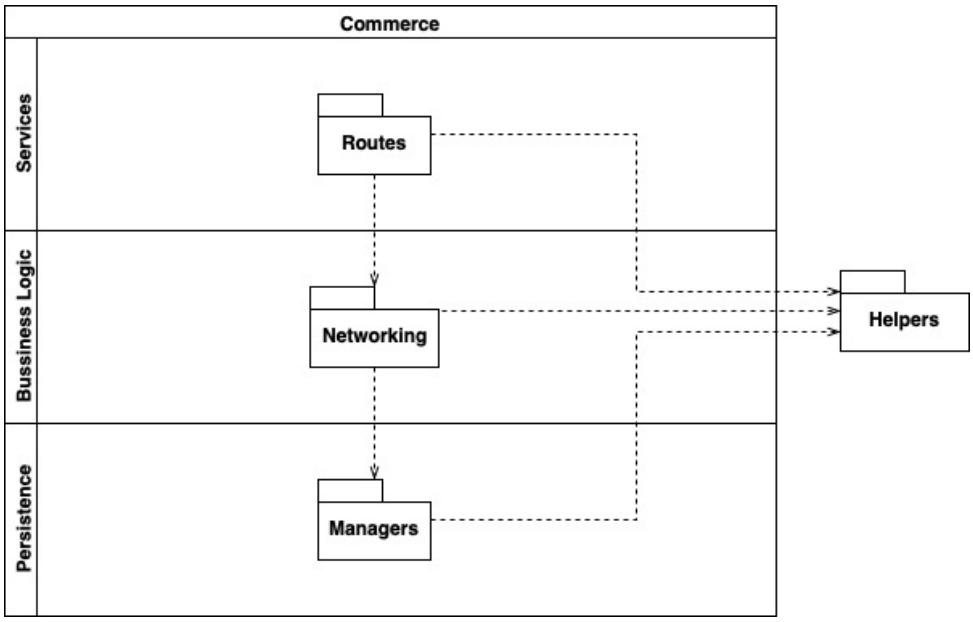
3.4.2.1.3 – Decisiones de diseño

Podemos observar que solo hay cuatro módulos. Esto es debido a que se tomó la decisión de tratar de mantener una buena separación de responsabilidades, con el fin

de reducir el acoplamiento en el paquete y lograr una buena mantenibilidad. También debido a esto se aumenta la extensibilidad, siendo esto algo muy positivo para el modulo Commerce pudiendo expandir y agregar nuevas funcionalidades fácilmente sin afectar a la vieja funcionalidad.

3.4.2.1 – Vista de Layers

3.4.2.1.1 – Representación primaria



3.4.2.1.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable

3.4.2.1.3 – Decisiones de diseño

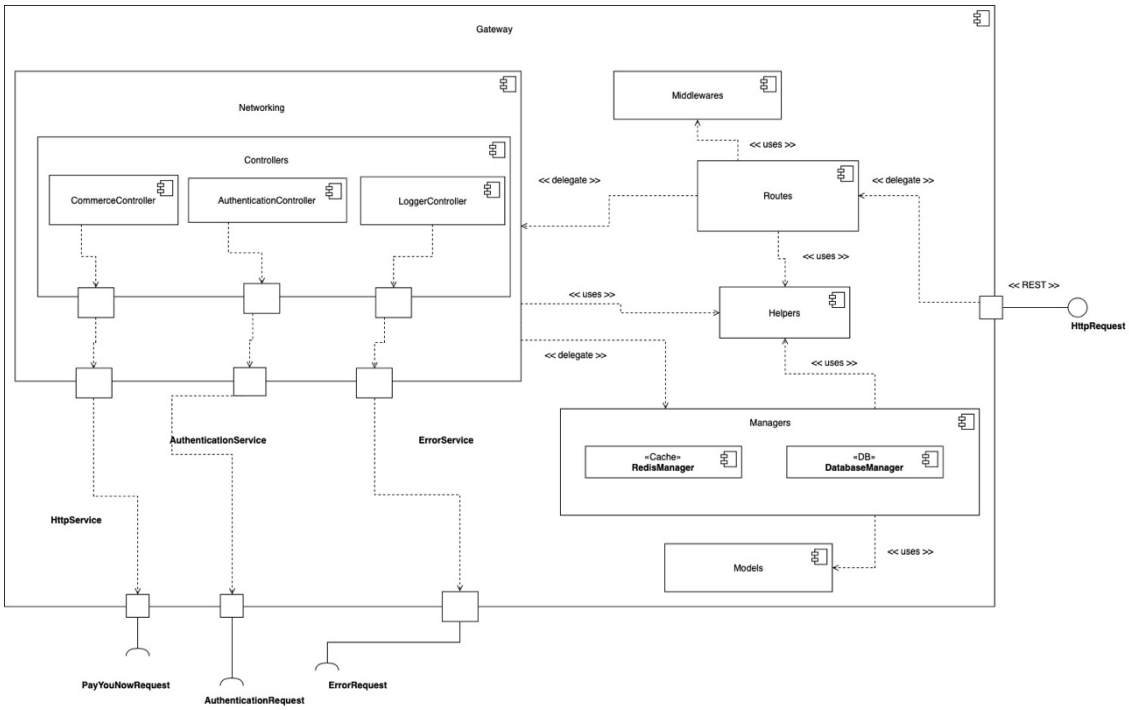
En esta vista podemos observar mas claramente como se manejan las responsabilidades internamente. Como se describió en el apartado anterior siempre se priorizo el tema de tener una buena separación de responsabilidades para darle un aumento a la mantenibilidad, mantener bajo el acoplamiento y lograr extensibilidad, siguiendo con los principios SOLID y respetando también principios de paquetes como son el Principio de dependencias estables.
Esta separación de capas ayuda en este propósito dándole una “arquitectura” robusta a los módulos, que además de ser robusta gana en mantenibilidad.

Se decidió descomponer la primera jerarquía de módulos por capas. Se distinguen capas muy definidas que son la de services y la implementación de la lógica de negocio. Se cree que estas dos capas son fundamentales de separar ya que tienen responsabilidades muy definidas y en el caso de los services, se cree que es probable que en algún momento se quiera agregar otro tipo de interfaz por lo que la división parece necesaria. La capa de web services utiliza a la capa de lógica a la de persistence y así sucesivamente.

3.5 – Gateway

3.5.1 – Vista de componentes y conectores.

3.5.1.1 – Representación primaria



3.5.1.2 – Interfaces

Elemento	Responsabilidades
HttpRequest*	Expone una "interfaz" REST, con las operaciones de commerce hacia el mundo exterior.
PayYouNowRequest	Solicita la "interfaz" REST a payYouNow para poder comunicarse con el Gateway y demás a modo de efectuar las transacciones
AuthenticationRequest	Solicita la "interfaz" REST a Authentication para poder registrarse, autenticarse y validar los tokens.
ErrorRequest	Solicita la "interfaz" REST a ErrorHandler para poder llevar una trazabilidad de las operaciones realizadas y para tener un mecanismo de control de defectos y fallas

3.5.1.3 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Controllers	Contiene todos los controllers de la aplicación.
GatewayController	Contiene toda la lógica asociada al negocio, se encarga de ajustar los datos y de mandarlos de forma correcta hacia el HttpService de forma de realizar las comunicaciones
AuthenticationController	Encargado de todo lo referido a la autenticación y su comunicación con el servicio.
LoggerController	Encargado de todo lo referido al manejo de errores y su comunicación con el servicio.
ErrorService	Service que funciona como "interfaz" y esta encargado de toda la comunicación que se realiza directamente con el modulo ErrorHandler.
HttpService	Service que funciona como "interfaz" y esta encargado de toda la comunicación que se realiza directamente con el modulo PayYouNow, también puede encargarse de otro tipo de comunicación http.

Elemento	Responsabilidades
AuthenticationService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo AuthenticationService.
Managers	Contiene otra capa que en este apartado vendría a representar la persistencia y el manejo de cache, contiene los “manejadores” que se encargan de estas tareas descriptas.
RedisManager	Encargado de manejar el cache por medio de una database de Redis.
DatabaseManager	Encargado de manejar toda la lógica y de realizar las operaciones referidas a la base de datos.
Models	Contiene las entidades del sistema y los modelos de la base de datos.
Middlewares	Funciones especiales que oportan una lógica especial al sistema
Helpers	Funciones auxiliares y utilidades, dan mas mantenibilidad y extensibilidad al código

3.5.1.4 – Decisiones de diseño

Las decisiones tomadas para este paquete también como ocurrió en el paquete de comercio siempre fueron pensadas de forma de aumentar la modificabilidad.
La diferencia con Commerce mas que nada radica en que se usa una capa mas en la cual se almacenan las entidades del sistema y los modelos de la base de datos.
Se decidió hacer esta separación para mantener el principio de responsabilidad única, basándonos en los SOLID.

Para temas de performance se implemento Redis, la cual es una base de datos para manejo de cache, este manejo de cache es extremadamente mas performante que MongoDB como ya se planteo en apartados anteriores.

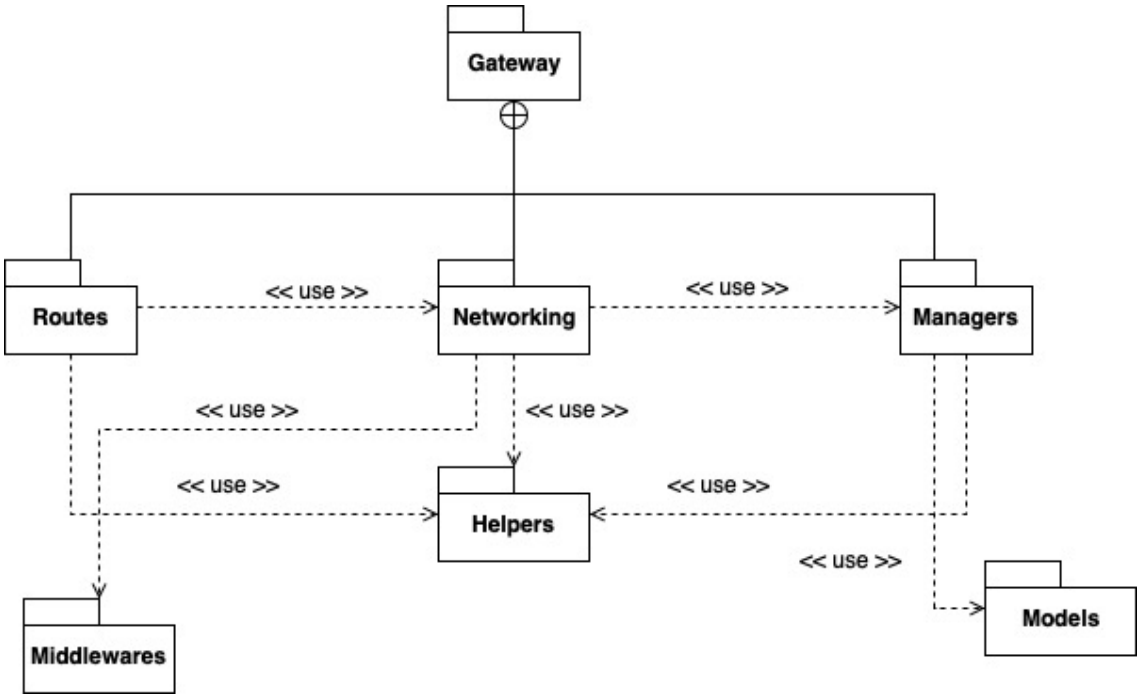
Además esta memoria cache se uso para almacenar el token de autenticación del modulo dando un toque de seguridad al modulo.

El manejo de logs y errores se hace a través del modulo ErrorHandler, esto se decidió hacer así también para reducir el acoplamiento y evitar sobrecargar al modulo con responsabilidades que no son propias de el. El funcionamiento en si es que el modulo dispara su mensaje de error, ya que esto si es responsabilidad del módulo y el encargado de dejarlo registrado y ver que hacer con el es el ErrorHandler el cual utiliza el servicio web de Rollbar para cumplir con el cometido.

3.5.2 – Vista de módulos

3.5.2.1 – Vista de usos

3.5.2.1.1 – Representación primaria



3.5.2.1.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable
Middlewares	Contienen lógica especial, funcionan como interceptores de las request.
Models	Contienen las entidades del sistema

3.5.2.1.3 – Decisiones de diseño

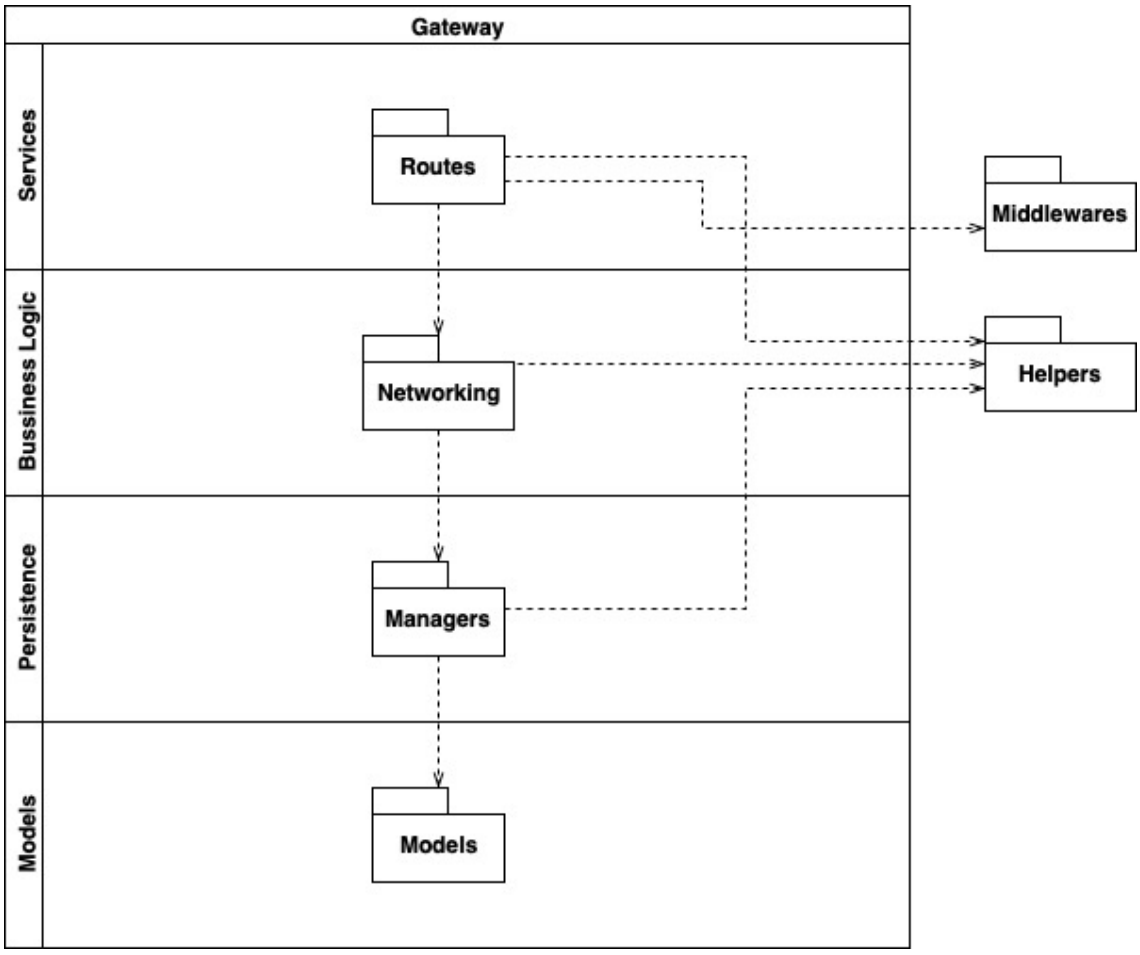
Se mantienen la arquitectura descripta en el apartado superior de comercio pero además, en este paquete podemos observar que se incluyen dos módulos mas a la arquitectura: Middlewares y Models.
La decisión de aportar estos dos nuevos módulos a la arquitectura fue debido a seguir manteniendo una buena separación de responsabilidades, mantenibilidad, etc.

Los Middlewares además aportan una lógica para la autenticación, que de no haberlo implementado de esta manera la misma estaría dispersa y repetida por todo el código siendo esto indeseable.

Models viene a contener las entidades del sistema, para futuras iteraciones se puede pensar en incluso agregar validaciones propias a las entidades mismas como pueden ser funciones especiales de validación, éstas pueden ser agregadas sin ningún impacto en el código.

3.5.2.2 – Vista de Layers

3.5.2.2.1 – Representación primaria



3.5.2.2.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable
Middlewares	Contienen lógica especial, funcionan como interceptores de las request.
Models	Contienen las entidades del sistema

3.5.2.2.3 – Decisiones de diseño

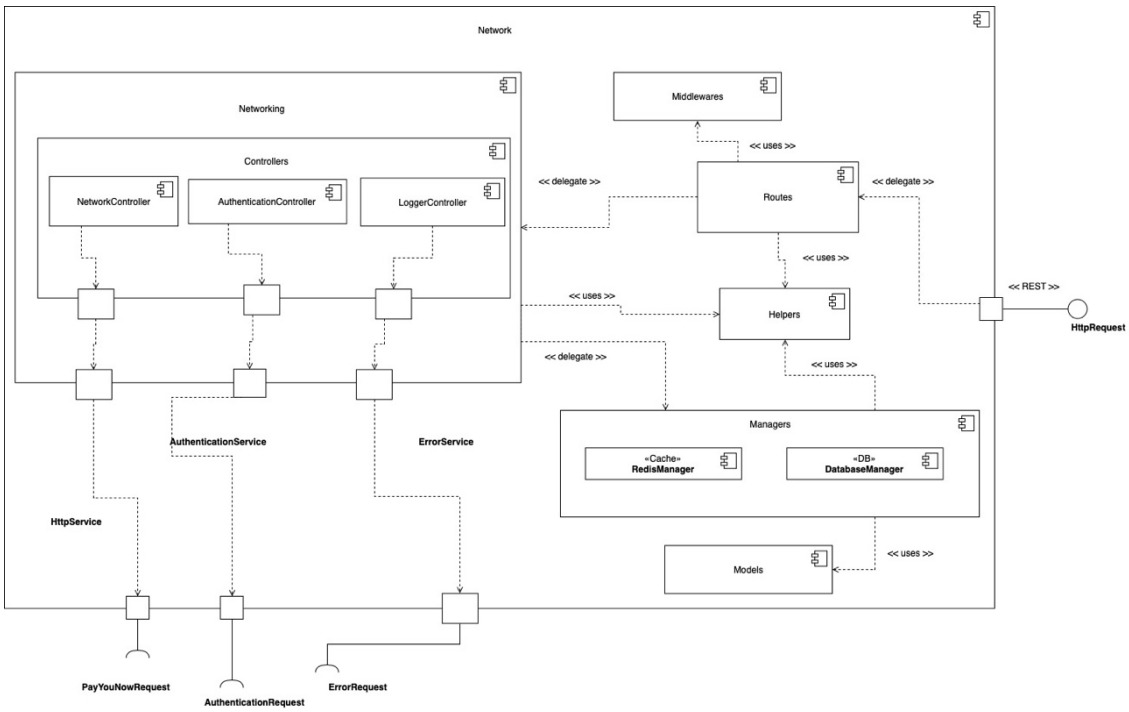
En esta vista de layers podemos observar mas claramente como se manejan las responsabilidades internamente en Gateway. Podemos observar como se agrego una nueva capa para las entidades y como también tenemos transversalmente a la solución el modulo de Middlewares. Como se describió en el apartado anterior siempre se priorizo el tema de tener una buena separación de responsabilidades para darle un aumento a la mantenibilidad, mantener bajo el acoplamiento y lograr extensibilidad, respetando los principios SOLID y patrones de paquetes como el de dependencias estables.
Esta separación de capas ayuda en este propósito, dando lo descripto arriba.

Se distinguen capas muy definidas que son la de services y la implementación de la lógica de negocio. Se cree que estas dos capas son fundamentales de separar ya que tienen responsabilidades muy definidas y en el caso de los services, se cree que es probable que en algún momento se quiera agregar otro tipo de interfaz por lo que la división parece necesaria. La capa de web services utiliza a la capa de lógica a la de persistence y esta con la capa que contiene las entidades.
También como se puede observar los módulos de middlewares y helpers se encuentran aislados de la separación por capas siendo transversales a las mismas

3.6 – Network

3.6.1 – Vista de componentes y conectores.

3.6.1.1 – Representación primaria



3.6.1.2 – Interfaces

Elemento	Responsabilidades
HttpRequest*	Expone una “interfaz” REST, con las operaciones de commerce hacia el mundo exterior.
PayYouNowRequest	Solicita la “interfaz” REST a payYouNow para poder comunicarse con el Gateway y demás a modo de efectuar las transacciones
AuthenticationRequest	Solicita la “interfaz” REST a Authentication para poder registrarse, autenticarse y validar los tokens.
ErrorRequest	Solicita la “interfaz” REST a ErrorHandler para poder llevar una trazabilidad de las operaciones realizadas y para tener un mecanismo de control de defectos y fallas

3.6.1.3 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Controllers	Contiene todos los controllers de la aplicación.
NetworkController	Contiene toda la lógica asociada al negocio, se encarga de ajustar los datos y de mandarlos de forma correcta hacia el HttpService de forma de realizar las comunicaciones
AuthenticationController	Encargado de todo lo referido a la autenticación y su comunicación con el servicio.
LoggerController	Encargado de todo lo referido al manejo de errores y su comunicación con el servicio.
ErrorService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo ErrorHandler.
HttpService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo PayYouNow, también puede encargarse de otro tipo de comunicación http.
AuthenticationService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo AuthenticationService.
Managers	Contiene otra capa que en este apartado vendría a representar la persistencia y el manejo de cache, contiene los “manejadores” que se encargan de estas tareas descriptas.
RedisManager	Encargado de manejar el cache por medio de una database de Redis.
DatabaseManager	Encargado de manejar toda la lógica y de realizar las operaciones referidas a la base de datos.
Models	Contiene las entidades del sistema y los modelos de la base de datos.
Middlewares	Funciones especiales que oportan una lógica especial al sistema
Helpers	Funciones auxiliares y utilidades, dan mas mantenibilidad y extensibilidad al código

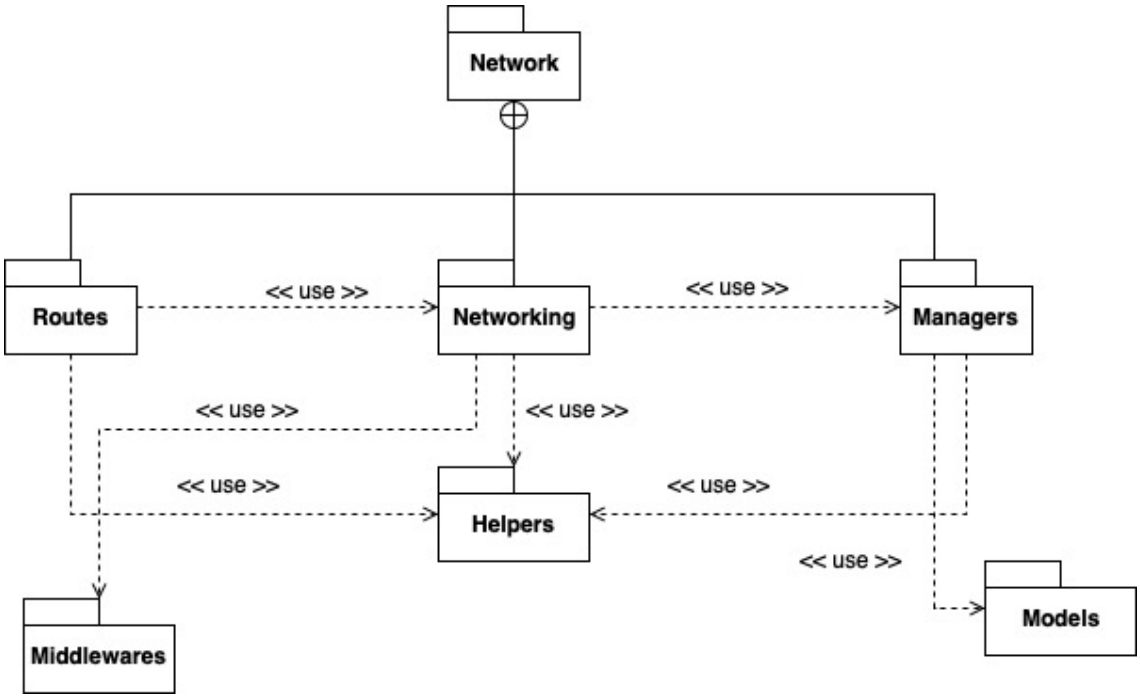
3.6.1.4 – Decisiones de diseño

Como la arquitectura de Network y Gateway es igual, las decisiones van a ser las mismas que ya se describieron en Gateway por lo tanto se decidió por omitir este apartado y no copiar lo mismo.
Se agrega una referencia al apartado de Gateway que describe el mismo.

3.6.2 – Vista de módulos

3.6.2.1 – Vista de usos

3.6.2.1.1 – Representación primaria



3.6.2.1.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable

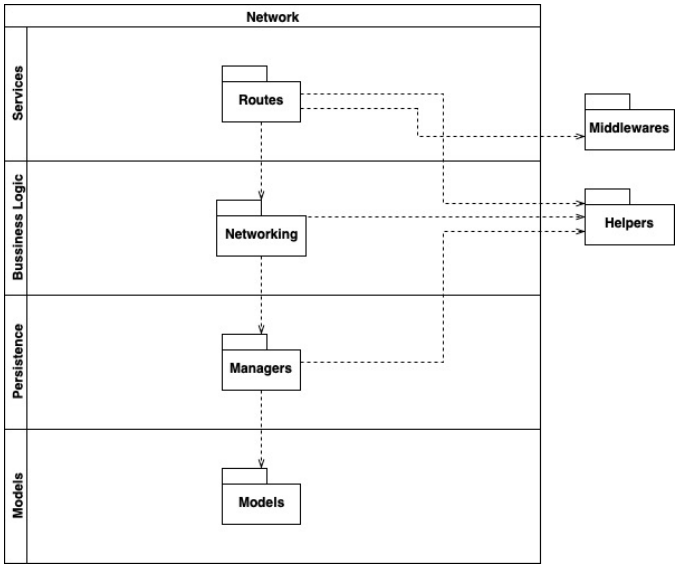
Elemento	Responsabilidades
Middlewares	Contienen lógica especial, funcionan como interceptores de las request.
Models	Contienen las entidades del sistema

3.6.2.1.3 – Decisiones de diseño

Como la arquitectura de Network y Gateway es igual, las decisiones van a ser las mismas que ya se describieron en Gateway por lo tanto se decidió por omitir este apartado y no copiar lo mismo.
Se agrega una referencia a Gateway que describe el mismo apartado.

3.6.2.2 – Vista de Layers

3.6.2.2.1 – Representación primaria



3.6.2.2.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.

Elemento	Responsabilidades
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable
Middlewares	Contienen lógica especial, funcionan como interceptores de las request.
Models	Contienen las entidades del sistema

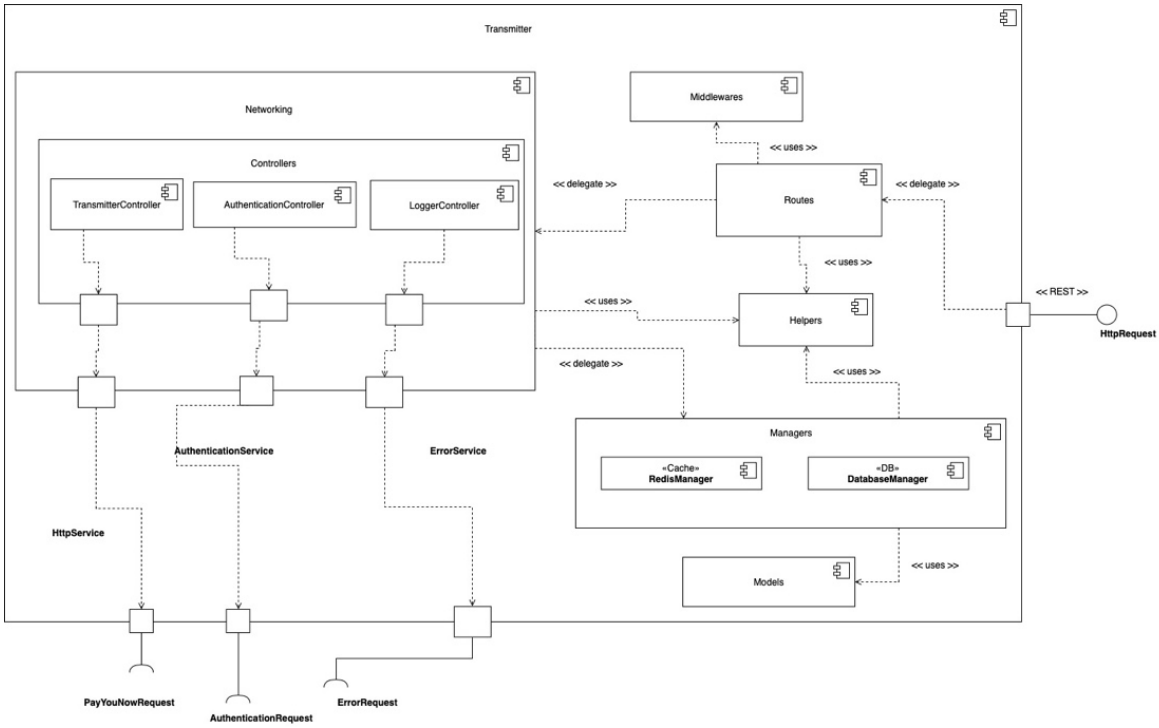
3.6.2.2.3 – Decisiones de diseño

Como la arquitectura de Network y Gateway es igual, las decisiones van a ser las mismas que ya se describieron en Gateway por lo tanto se decidió por omitir este apartado y no copiar lo mismo.
Se agrega una referencia a Gateway que describe el mismo apartado.

3.7 – Transmitter.

3.7.1 – Vista de componentes y conectores.

3.7.1.1 – Representación primaria



3.7.1.2 – Interfaces

Elemento	Responsabilidades
HttpRequest*	Expone una "interfaz" REST, con las operaciones de commerce hacia el mundo exterior.
PayYouNowRequest	Solicita la "interfaz" REST a payYouNow para poder comunicarse con el Gateway y demás a modo de efectuar las transacciones
AuthenticationRequest	Solicita la "interfaz" REST a Authentication para poder registrarse, autenticarse y validar los tokens.
ErrorRequest	Solicita la "interfaz" REST a ErrorHandler para poder llevar una trazabilidad de las operaciones realizadas y para tener un mecanismo de control de defectos y fallas

3.7.1.3 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Controllers	Contiene todos los controllers de la aplicación.
TransmitterController	Contiene toda la lógica asociada al negocio, se encarga de ajustar los datos y de mandarlos de forma correcta hacia el HttpService de forma de realizar las comunicaciones
AuthenticationController	Encargado de todo lo referido a la autenticación y su comunicación con el servicio.
LoggerController	Encargado de todo lo referido al manejo de errores y su comunicación con el servicio.
ErrorService	Service que funciona como "interfaz" y esta encargado de toda la comunicación que se realiza directamente con el modulo ErrorHandler.
HttpService	Service que funciona como "interfaz" y esta encargado de toda la comunicación que se realiza directamente con el modulo PayYouNow, también puede encargarse de otro tipo de comunicación http.
AuthenticationService	Service que funciona como "interfaz" y esta encargado de toda la comunicación que se realiza directamente con el modulo AuthenticationService.

Elemento	Responsabilidades
Managers	Contiene otra capa que en este apartado vendría a representar la persistencia y el manejo de cache, contiene los “manejadores” que se encargan de estas tareas descriptas.
RedisManager	Encargado de manejar el cache por medio de una database de Redis.
DatabaseManager	Encargado de manejar toda la lógica y de realizar las operaciones referidas a la base de datos.
Models	Contiene las entidades del sistema y los modelos de la base de datos.
Middlewares	Funciones especiales que oportan una lógica especial al sistema
Helpers	Funciones auxiliares y utilidades, dan mas mantenibilidad y extensibilidad al código

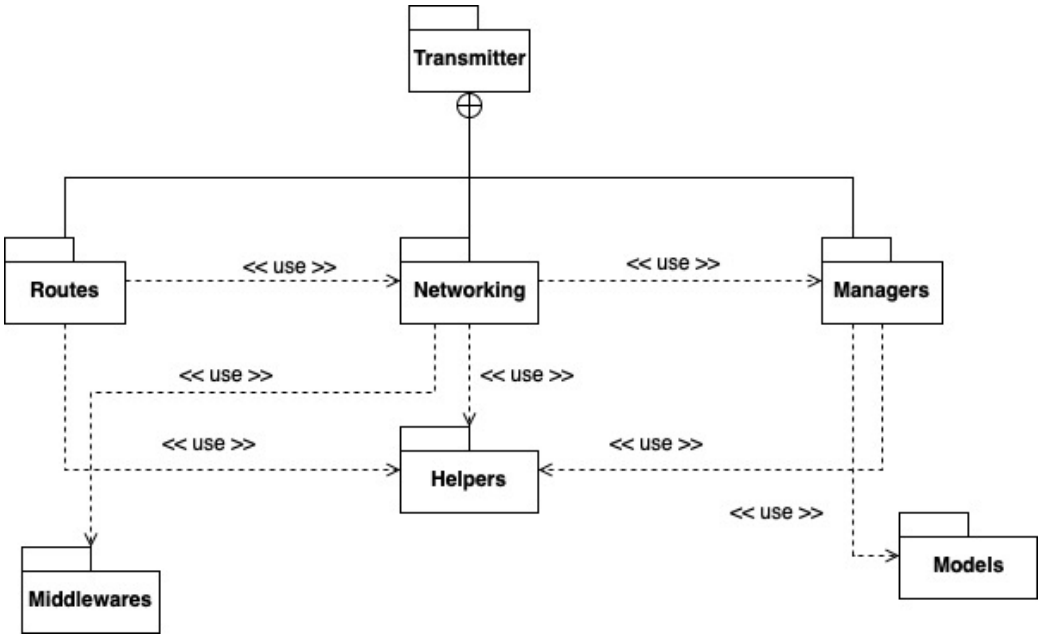
3.7.1.4 – Decisiones de diseño

Como la arquitectura de Network, Gateway y Transmitter es igual, las decisiones van a ser las mismas que ya se describieron en Gateway por lo tanto se decidió por omitir este apartado y no copiar lo mismo.
Se agrega una referencia a Gateway en el apartado que describe lo mismo.

3.7.2 – Vista de módulos

3.7.2.1 – Vista de usos

3.7.2.1.1 – Representación primaria



3.7.2.1.2 – Catalogo de elementos

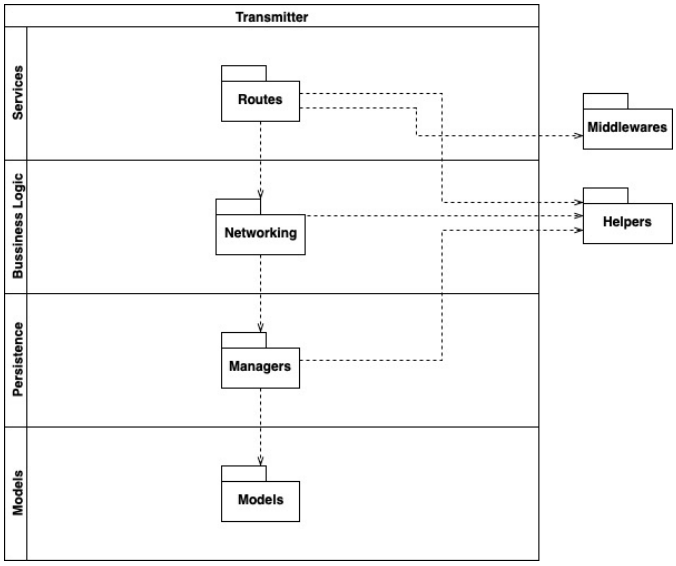
Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable
Middlewares	Contienen lógica especial, funcionan como interceptores de las request.
Models	Contienen las entidades del sistema

3.7.2.1.3 – Decisiones de diseño

Como la arquitectura de Network, Gateway y Transmitter es igual, las decisiones van a ser las mismas que ya se describieron en Gateway por lo tanto se decidió por omitir este apartado y no copiar lo mismo.
Se agrega una referencia a Gateway hacia el apartado que describe lo mismo.

3.7.2.2 – Vista de Layers

3.7.2.2.1 – Representación primaria



3.7.2.2.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable
Middlewares	Contienen lógica especial, funcionan como interceptores de las request.
Models	Contienen las entidades del sistema

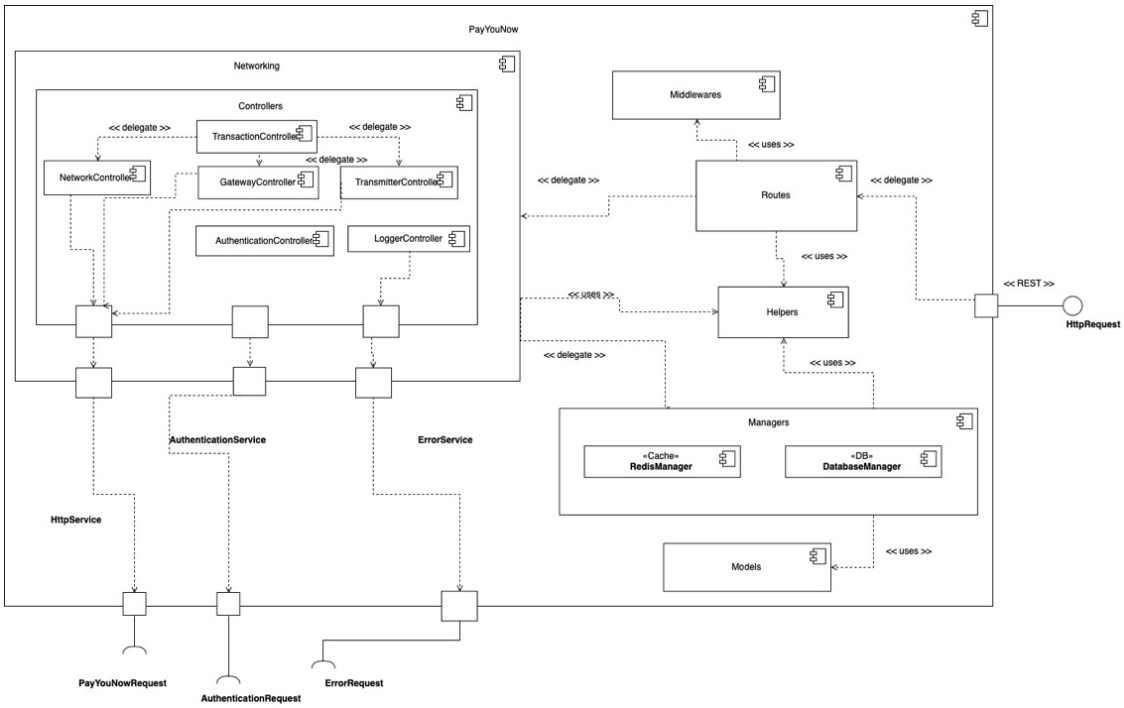
3.7.2.2.3 – Decisiones de diseño

Como la arquitectura de Network, Gateway y Transmitter es igual, las decisiones van a ser las mismas que ya se describieron en Gateway por lo tanto se decidió por omitir este apartado y no copiar lo mismo.
Se agrega una referencia a Gateway al apartado que describe lo mismo.

3.8 – PayYouNow

3.8.1 – Vista de componentes y conectores.

3.8.1.1 – Representación primaria



3.8.1.2 – Interfaces

Elemento	Responsabilidades
HttpRequest*	Expone una “interfaz” REST, con las operaciones de commerce hacia el mundo exterior.
PayYouNowRequest	Solicita la “interfaz” REST a payYouNow para poder comunicarse con el Gateway y demás a modo de efectuar las transacciones
AuthenticationRequest	Solicita la “interfaz” REST a Authentication para poder registrarse, autenticarse y validar los tokens.
ErrorRequest	Solicita la “interfaz” REST a ErrorHandler para poder llevar una trazabilidad de las operaciones realizadas y para tener un mecanismo de control de defectos y fallas

3.8.1.3 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Controllers	Contiene todos los controllers de la aplicación.
TransactionController	Contiene toda la lógica asociada al negocio, la transacción en si, se encarga de ajustar los datos y de delegar a los diferentes controladores las peticiones en caso de necesitarlo o las realiza el mismo en caso de no ser necesario.
NetworkController	Contiene toda la lógica asociada al negocio de la parte de la Red y la comunicación con la misma, se encarga de ajustar los datos y de mandarlos de forma correcta hacia el HttpService de forma de realizar las comunicaciones
GatewayController	Contiene toda la lógica asociada al negocio de la parte del Gateway y la comunicación con el mismo, se encarga de ajustar los datos y de mandarlos de forma correcta hacia el HttpService de forma de realizar las comunicaciones
TransmitterController	Contiene toda la lógica asociada al negocio de la parte del Emisor y la comunicación con el mismo, se encarga de ajustar los datos y de mandarlos de forma correcta hacia el HttpService de forma de realizar las comunicaciones
AuthenticationController	Encargado de todo lo referido a la autenticación y su comunicación con el servicio.
LoggerController	Encargado de todo lo referido al manejo de errores y su comunicación con el servicio.
ErrorService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo ErrorHandler.
HttpService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo PayYouNow, también puede encargarse de otro tipo de comunicación http.

Elemento	Responsabilidades
AuthenticationService	Service que funciona como “interfaz” y esta encargado de toda la comunicación que se realiza directamente con el modulo AuthenticationService.
Managers	Contiene otra capa que en este apartado vendría a representar la persistencia y el manejo de cache, contiene los “manejadores” que se encargan de estas tareas descriptas.
RedisManager	Encargado de manejar el cache por medio de una database de Redis.
DatabaseManager	Encargado de manejar toda la lógica y de realizar las operaciones referidas a la base de datos.
Models	Contiene las entidades del sistema y los modelos de la base de datos.
Middlewares	Funciones especiales que oportan una lógica especial al sistema
Helpers	Funciones auxiliares y utilidades, dan mas mantenibilidad y extensibilidad al código

3.8.1.3 – Decisiones de diseño

PayYouNow es quizás el modulo mas importante en toda la solución, debido a esto además de tener la arquitectura que ya se ha descripto en los módulos anteriores, con las capaz ya mencionadas y la forma de comunicación ya descripta este modulo además cuenta con otras consideraciones necesarias para lograr manejar el papel de ser intermediario de una manera eficaz y performante.

Para lograr esto se agregaron controladores específicos para soportar la lógica asociada a cada parte de la comunicación de la solución, un controlador para Gateway, para Network, para Transmitter y un controlador superior a estos en jerarquía que se encarga de orquestar la transacción, decide a que controlador le delega la tarea en caso de ser necesario, agregando una capa de abstracción y separación.

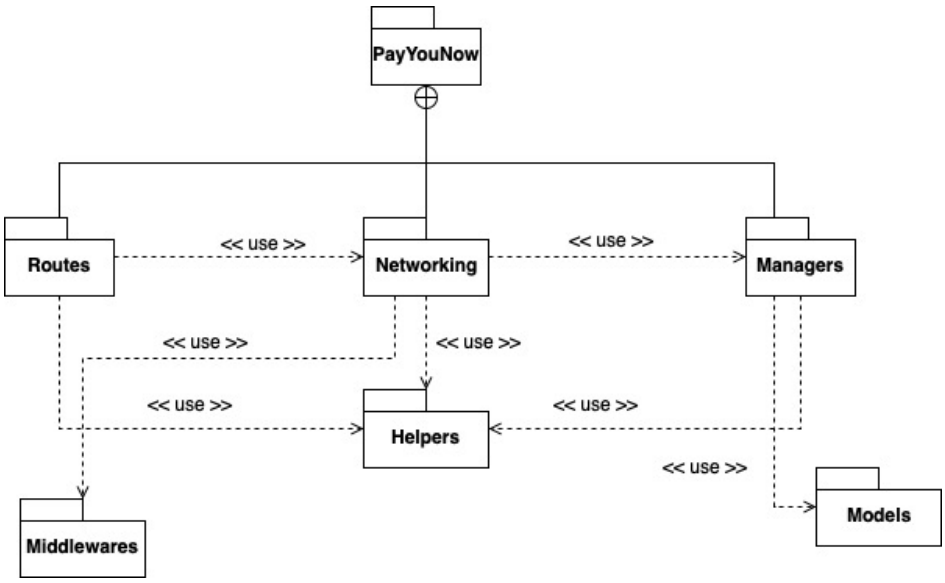
Este modulo también es el que provee lógica para el registro de nuevos servicios, los servicios para registrarse solo necesitan especificar su interfaz en un formato acordado en el cual le indican, url, nombre, métodos y parámetros.
A partir de esta interfaz después el modulo provee una interfaz REST para consumir estos métodos registrados adaptando los datos necesarios y los formatos que son especificados en la documentación.

Al ser un modulo tan importante y ya con mucha responsabilidad se opto por separar la autenticación y los errores en dos módulos diferentes, por mas que de podrían estar en este modulo ya que el centraliza todo. Así logramos aislar a este modulo que solo sea encargado de orquestar cosa de no restarle performance en operaciones innecesarias.

3.8.2 – Vista de módulos

3.8.2.1 – Vista de usos

3.8.2.1.1 – Representación primaria



3.8.2.1.2 – Catalogo de elementos

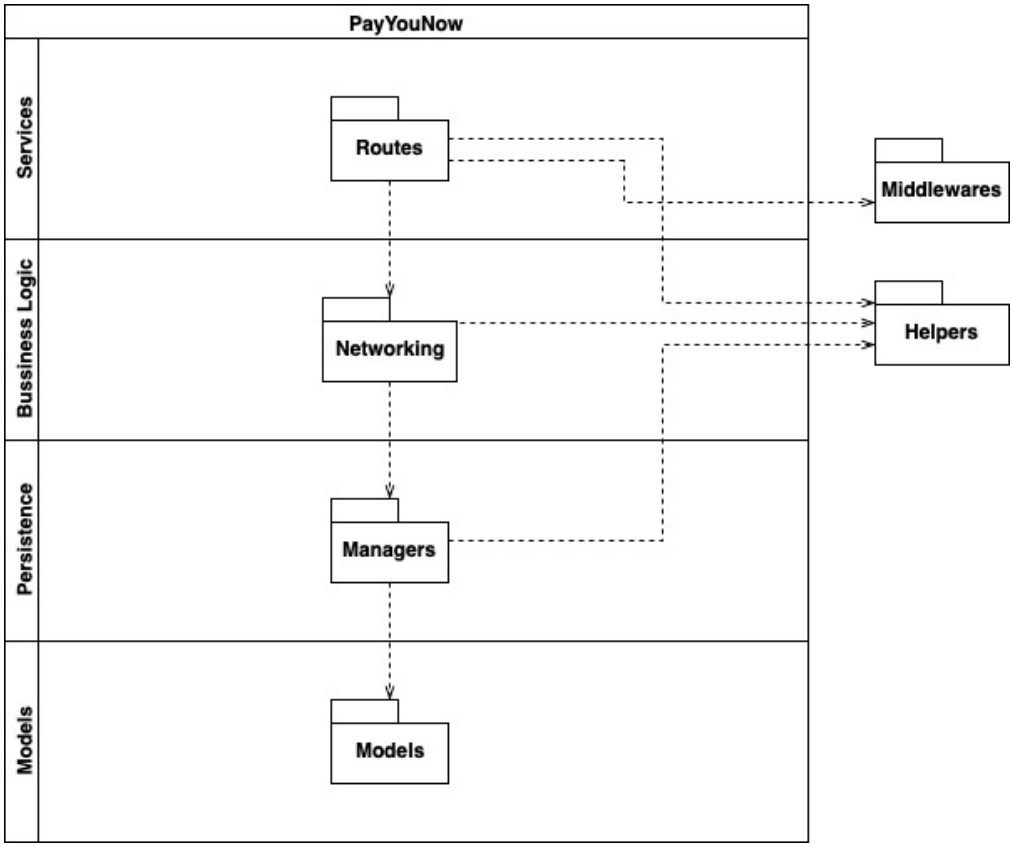
Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable
Middlewares	Contienen lógica especial, funcionan como interceptores de las request.
Models	Contienen las entidades del sistema

3.8.2.1.3 – Decisiones de diseño

Ya que PayYouNow es el encargado de dirigir peticiones y funcionar como intermediario se trato de mantenerlo lo mas estable posible, respetando que no hubiera acoplamiento con el fin de que el modulo que fuera mantenible y extensible. A su vez se trato de usar técnicas de guardado y manejo de cache para optimizar los pedidos y lograr una buena performance siendo quizás este el atributo de calidad mas requerido debido a el rol que cumple.

3.8.2.2 – Vista de Layers

3.8.2.2.1 – Representación primaria



3.8.2.2.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.

Elemento	Responsabilidades
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable
Middlewares	Contienen lógica especial, funcionan como interceptores de las request.
Models	Contienen las entidades del sistema

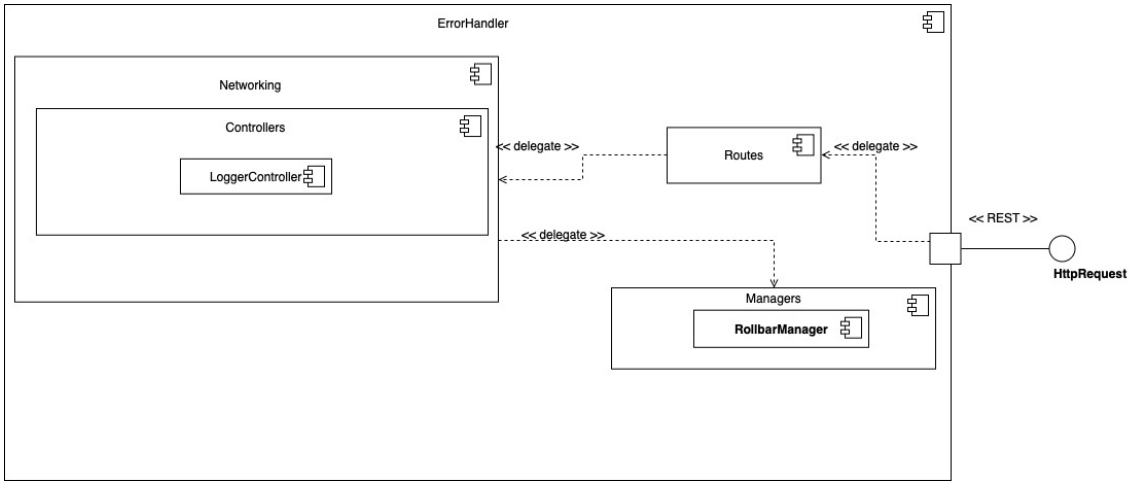
3.8.2.2.3 – Decisiones de diseño

Como la separación de capas de PayYouNow es igual a las de Gateway, Network y Transmitter, las decisiones de diseño que fueron tomadas son las mismas que ya se describieron en Gateway por lo tanto se decidió por omitir este apartado y no copiar lo mismo.

3.9 – ErrorHandler

3.9.1 – Vista de componentes y conectores.

3.9.1.1 – Representación primaria



3.9.1.2 – Interfaces

Elemento	Responsabilidades
HttpRequest*	Expone una “interfaz” REST, con las operaciones de commerce hacia el mundo exterior.

3.9.1.3 – Catalogo de elementos

Elemento	Tipo	Responsabilidades
Routes		Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking		Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Controllers		Contiene todos los controllers de la aplicación.
AuthenticationController		Encargado de todo lo referido a la autenticación y su comunicación con el servicio.
Managers		Contiene otra capa que en este apartado vendría a representar la persistencia y el manejo de cache, contiene los “manejadores” que se encargan de estas tareas descriptas.
RollbarManager		Encargado de manejar toda la lógica y de realizar las operaciones referidas a conectarse con el servicio de Rollbar.

3.9.1.3 – Decisiones de diseño

Para la realización de este modulo se opto por contar con el servicio de Rollbar el cual es un servicio externo que propone una solución para el manejo de Logs, y aporta trazabilidad de defectos y fallas.

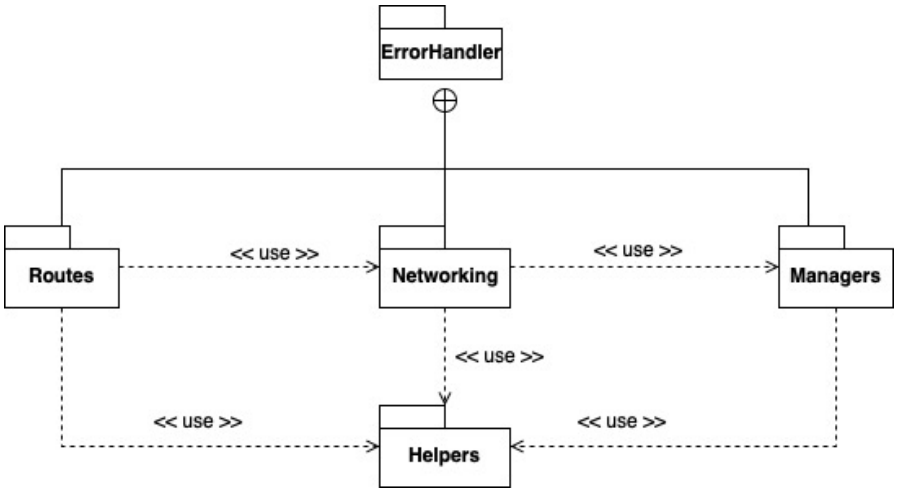
El único pero es que se queda acoplado a Rollbar en este sentido, pero también como es un proyecto aparte, esta implementación puede ser cambiada sin muchos problemas sin afectar a los módulos que usan este servicio por lo que consideramos que los beneficios son suficientes y opacan a los “trade-off”.

En caso de ocurrir defectos o fallas o querer ver los logs basta con entrar en la web de Rollbar y buscar el proyecto asociado para ver que fue lo ocurrido.

3.9.2 – Vista de módulos

3.9.2.1 – Vista de usos

3.9.2.1.1 – Representación primaria



3.9.2.1.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable

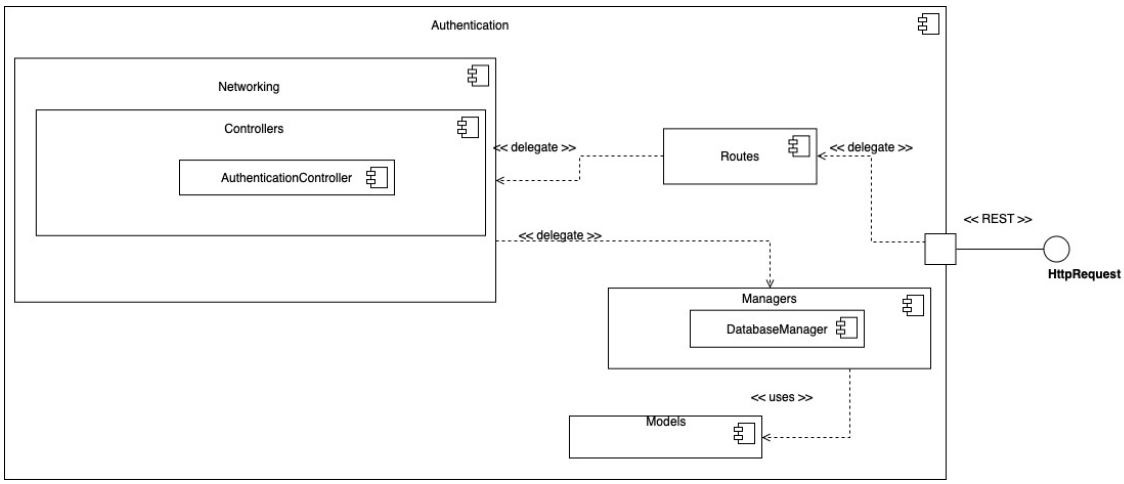
3.9.2.1.3 – Decisiones de diseño

Podemos observar que solo hay cuatro módulos en esta solución. Esto es debido a que se tomó la decisión de tratar de mantener una buena separación de responsabilidades, con el fin de reducir el acoplamiento en el paquete y lograr una buena mantenibilidad. También debido a esto se aumenta la extensibilidad.

3.10 – Authentication.

3.10.1 – Vista de componentes y conectores

3.10.1.1 – Representación primaria



3.5.1.2 – Interfaces

Elemento	Responsabilidades
HttpRequest*	Expone una “interfaz” REST, con las operaciones de commerce hacia el mundo exterior.

3.10.1.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers y además los servicios para comunicarse con las diferentes funcionalidades como Autenticación y ErrorHandler.
Controllers	Contiene todos los controllers de la aplicación.
AuthenticationController	Encargado de todo lo referido a la autenticación y su comunicación con el servicio.
Managers	Contiene otra capa que en este apartado vendría a representar la persistencia y el manejo de cache, contiene los “manejadores” que se encargan de estas tareas descriptas.
DatabaseManager	Encargado de manejar toda la lógica y de realizar las operaciones referidas a la base de datos.

Elemento	Responsabilidades
Models	Contiene las entidades del sistema y los modelos de la base de datos.

3.10.1.3 – Decisiones de diseño

La realización de este modulo se llevo a cabo para poder tener una entidad superior que se encargue de dar credenciales y validarlas, de manera de obtener una capa de seguridad en los diferentes módulos de la solución.

Para manejar la autenticación se opto por usar el método de autenticación por Token siendo JsonWebToken el método elegido debido a los innumerables números de beneficios con lo que estos cuentan, siendo algunos los siguientes:

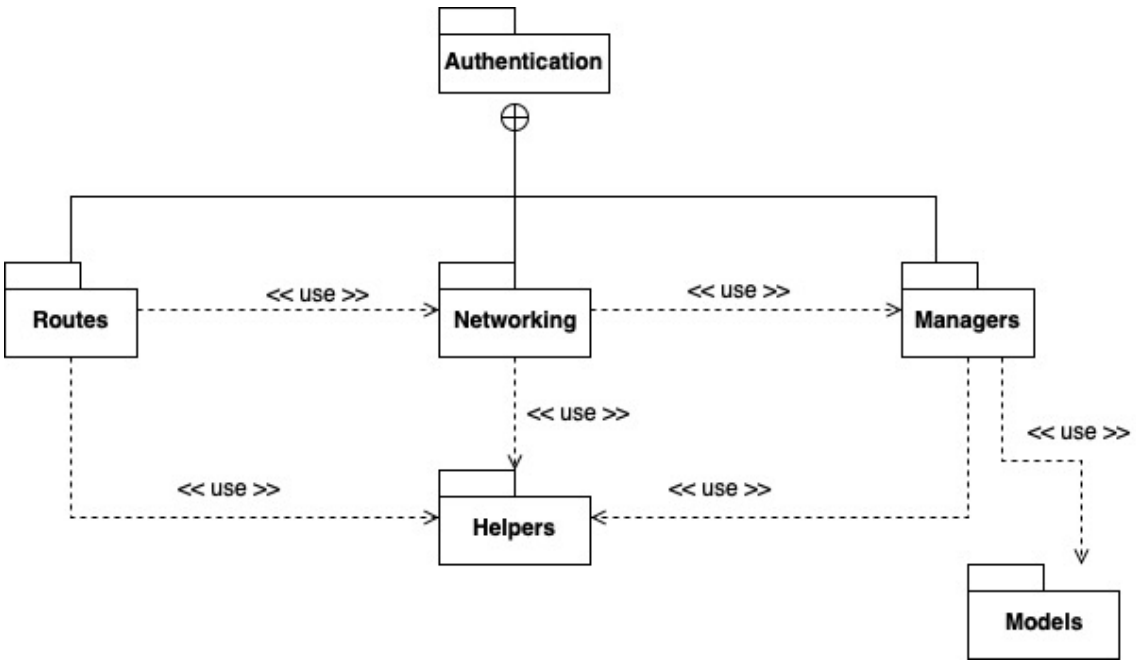
- Funcionan en diferentes lenguajes de programación por lo que permite su uso en diferentes escenarios.
- Son autónomos, cargan con toda la información necesaria.
- Tienen un tamaño muy pequeño, lo que favorece la performance.

Otra cosa que se pensó implementar pero no se llego debido al tiempo del equipo de desarrollo fue una política de refresco de los tokens, JsonWebToken ya provee esta funcionalidad siendo muy fácil de añadirla en un futuro.

3.10.2 – Vista de módulos

3.10.2.1 – Vista de usos

3.10.2.1.1 – Representación primaria



3.10.2.1.2 – Catalogo de elementos

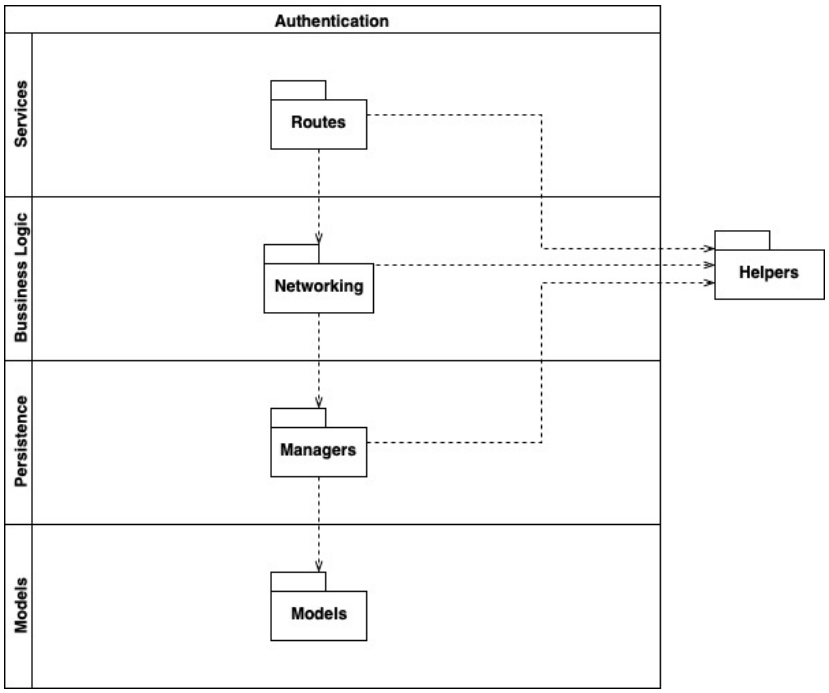
Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable
Models	Contienen las entidades del sistema.

3.10.2.1.3 – Decisiones de diseño

Podemos observar que solo hay cinco módulos en esta solución. Esto es debido a que se tomó la decisión de tratar de mantener una buena separación de responsabilidades, con el fin de reducir el acoplamiento en el paquete y lograr una buena mantenibilidad. Con esta separación logramos extensibilidad también siendo muy fácil agregar nueva funcionalidad en el modulo de autenticación.

3.10.3 – Vista de Layers

3.10.3.1 – Representación primaria



3.10.3.2 – Catalogo de elementos

Elemento	Responsabilidades
Routes	Encargado de manejar las peticiones/request que llegan al modulo, delega sus tareas a la capa lógica de networking
Networking	Contenedor donde se encuentran los controllers.
Managers	Contiene manejadores necesarios para el manejo de la base de datos y el cache
Helpers	Contiene funciones auxiliares y utilidades que ayudan al código a ser mas mantenible y modificable
Models	Contienen las entidades del sistema.

3.10.3.3 – Decisiones de diseño

Podemos observar que solo hay cinco módulos en esta solución. Esto es debido a que se tomó la decisión de tratar de mantener una buena separación de responsabilidades, con el fin de reducir el acoplamiento en el paquete y lograr una buena mantenibilidad. También debido a esto se aumenta la extensibilidad.

4 - Anexo

4.1 - Diagramas

Se añade una carpeta diagramas con todos los diagramas realizados en la documentación.

4.2 - Pruebas de carga

Se agrega una carpeta pruebas de carga donde se añaden los reportes de las pruebas de carga de:

- Compra en Gateway
- Compra en Network
- Cierre de lotes

Como podemos observar en los diferentes reportes, por mas que la performance quizás no es la deseada tenemos unos muy buenos resultados siendo lo bastante cercanos a lo esperado, esperando en futuros reléase lograr las mejoras deseadas.

Podemos ver también que se maneja una gran cantidad de usuarios concurrentes y en general la aplicación lo tolera bien.

También a partir de estar pruebas se puede concluir que el mayor impacto en la performance es realizado por las DB, en este caso MongoDB y algunos subSchemas hacen que la performance sufra caídas pudiendo incluso a contestar con 500.
Esto podría ser subsanado con maquinas especiales para servidores donde tuviera muchos mas recursos y no estuviera corriendo todo en una sola maquina que fue donde se realizaron los Test.