**DOCUMENTO DE PROCESO DE DISEÑO**

VERSIÓN 1.0



SANTIAGO ANDRÉS CAROPRESE HIDALGO

DANIEL HERNÁNDEZ GARCÍA

JUAN CARLOS SUÁREZ JAIMES

16 DE MARZO DE 2021

**Requisitos Arquitectónicamente Significativos**

|  |  |  |  |
| --- | --- | --- | --- |
| Prioridad | ID | Impacto arquitectónico | Requisito |
| ALTA | RF001 | Medio | El cliente deberá poder realizar pedidos de productos a domicilio a su franquicia disponible más cercana. |
| ALTA | RF002 | Medio | El cliente deberá poder realizar pedidos de productos para recoger en la franquicia de su elección. |
| ALTA | RF003 | Medio | Al realizar un pedido, el cliente deberá poder seleccionar uno o más productos disponibles en la franquicia seleccionada. |
| ALTA | RF005 | Medio | El cliente deberá poder confirmar su pedido para que la franquicia empiece a prepararlo. |
| ALTA | RF007 | Alto | Si el cliente selecciona pagar en línea, este deberá ingresar sus datos de pago y el sistema deberá realizar la transacción al confirmar el pedido. |
| ALTA | RF008 | Alto | La franquicia deberá ser informada de los pedidos que se le han encargado o asignado. |
| ALTA | RF012 | Medio | Cuando un pedido se entregará a domicilio, se le asignará un repartidor en la franquicia. |
| ALTA | RF026 | Medio | El empleado de una franquicia deberá poder agregar repartidores a esta. |
| ALTA | RF027 | Alto | Los usuarios deberán poder autenticarse. |
| ALTA | RF032 | Medio | El empleado de la franquicia deberá poder actualizar los inventarios de sus productos. |
| ALTA | RF033 | Alto | El cliente deberá poder visualizar el catálogo de productos |
| ALTA | RF034 | Medio | Al realizar un pedido, el cliente deberá poder seleccionar los ingredientes de su sándwich. |
| ALTA | RF047 | Alto | Si se cancela un pedido que ya se pagó, se deberá efectuar un reembolso. |
| ALTA | RF051 | Medio | El sistema deberá permitir que el empleado registre un pedido realizado por teléfono. |
| ALTA | RNF001 | Alto | El sistema debe estar disponible el 99.9% del horario en el que las franquicias están abiertas. |
| ALTA | RNF002 | Alto | El sistema deberá poder manejar 3 millones de usuarios simultáneamente sin una degradación grave del rendimiento. |
| ALTA | RNF004 | Medio | El sistema deberá ser accesible desde dispositivos móviles |
| ALTA | RNF009 | Medio | El sistema deberá ser capaz de interactuar con sistemas externos de pago en línea. |
| MEDIA | RF009 | Alto | El empleado de la franquicia deberá poder actualizar el estado del pedido. |
| MEDIA | RF010 | Alto | Al confirmar el pedido, el cliente deberá recibir una confirmación a través de mensajes asincrónicos. |
| MEDIA | RF011 | Alto | Desde el momento en el que el cliente realiza el pedido, el sistema deberá permitirle al cliente revisar el estado en el que se encuentra su pedido. |
| MEDIA | RF016 | Alto | Al realizar un pedido, si este se recogerá en la tienda, el sistema deberá proporcionarle al cliente indicaciones para llegar a la tienda. |
| MEDIA | RF018 | Alto | El sistema deberá registrar información de cada pedido para poder generar estadísticas de calidad de servicio. |
| MEDIA | RF025 | Alto | El administrador del sistema deberá poder agregar nuevas franquicias. |
| MEDIA | RF028 | Alto | El super administrador deberá poder crear administradores. |
| MEDIA | RF030 | Alto | Cuando una franquicia se desconecta y se reconecta, el sistema deberá recuperar la información de los pedidos en curso. |
| MEDIA | RF031 | Alto | Si una franquicia se desconecta por más 5 minutos, se intentará buscar otra franquicia cercana a la que se le pueda reasignar. |
| MEDIA | RF045 | Alto | Si una franquicia rechaza un pedido, se intentará buscar otra franquicia cercana a la que se le pueda reasignar. |
| MEDIA | RF046 | Alto | Si no se encuentra ninguna franquicia a la que se le pueda reasignar el pedido, este se cancelará. |
| MEDIA | RF049 | Alto | Si el usuario decide reasignar el pedido dentro de los primeros 30 minutos desde que recibió el mensaje de reasignación, el pedido se reasignará a la franquicia especificada. |
| MEDIA | RNF003 | Alto | El sistema deberá poder agregar y remover servidores dependiendo de la carga. |
| MEDIA | RNF012 | Alto | El tiempo de respuesta del sistema deberá tener un límite superior de 4 segundos. |
| MEDIA | RNF013 | Alto | La complejidad de modificación del sistema deberá corresponder a un mínimo de 2 modificaciones por hora. |

**Iteraciones del Proceso ADD**

**Iteración 1**

1. **Elemento escogido**

Todo el sistema

1. **Requisitos arquitectónicamente significativos del elemento escogido**

Dado que se escogió todo el sistema, se tendrán en cuenta, de manera general, todos los requisitos arquitectónicamente significativos porque estos se traducen a responsabilidades que serán asignadas a los componentes gruesos del sistema que se definirán. En cuanto a los atributos de calidad, esta iteración se enfocará principalmente en la disponibilidad del sistema, que incluye, a su vez, escalabilidad y elasticidad, abarcando los requisitos arquitectónicamente significativos RNF001, RNF002 y RNF003. Asimismo, se tendrá en cuenta el desempeño del sistema propuesto, por lo que también se tendrá presente el requisito RNF012. De igual manera, la división de los componentes también considerará la modificabilidad, relacionada con el requisito RNF013.

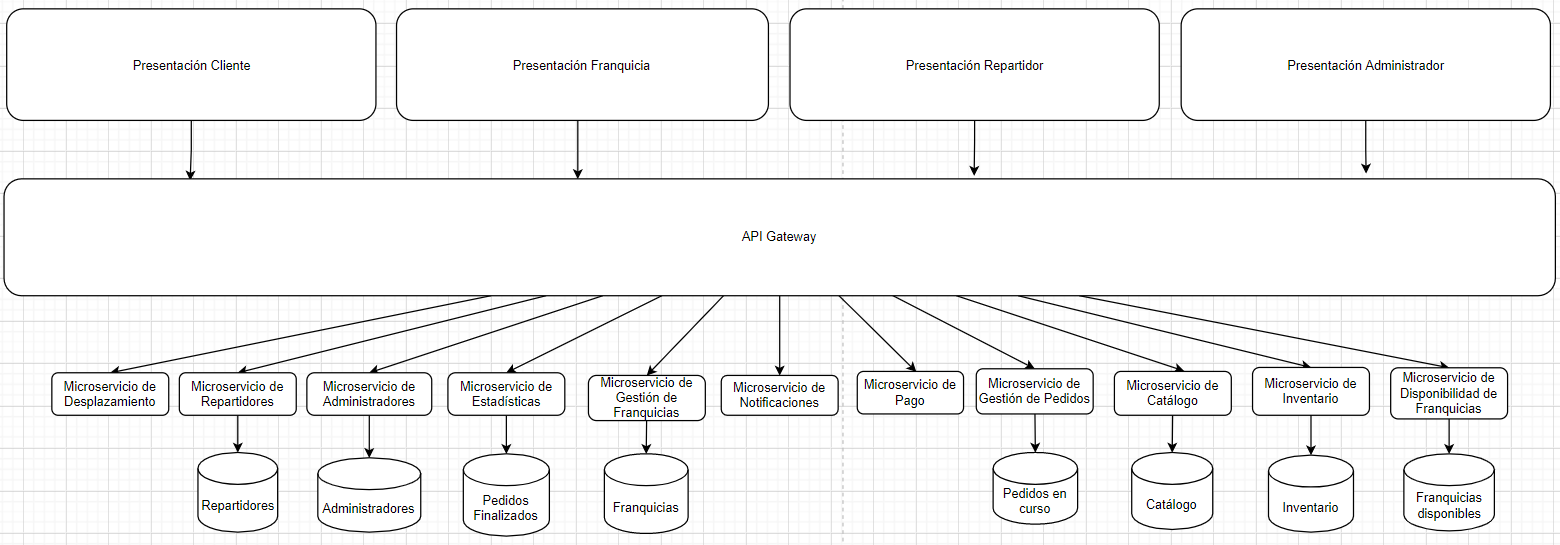
1. **Solución de diseño**

**Vista Lógica**

Se utilizó el patrón arquitectónico de Microservicios. Este patrón se escogió porque se identificó que las diferentes funciones del sistema tendrán cargas diferentes, por lo que dividir el sistema en microservicios permitiría escalar cada uno de manera independiente.

En el siguiente diagrama, se ilustra la división del sistema en los microservicios, cuyas responsabilidades son, a grandes rasgos, las siguientes:

* Servicio de Desplazamiento: Este microservicio se integrará con sistemas externos para proporcionar rutas para desplazarse de un lugar a otro y las distancias entre dos lugares. Se encuentra relacionado con los requisitos arquitectónicamente significativos RF001, RF016, RF031, RF045.
* Servicio de Repartidores: Este microservicio accederá a la base de datos que almacena a los datos de los repartidores, permitiendo modificar y acceder a la información de los repartidores inscritos en cada franquicia. Se encuentra relacionado con los requisitos arquitectónicamente significativos RF012, RF026.
* Servicio de Administradores: Este microservicio accederá a la base de datos que almacena la información de los administradores del sistema, permitiendo modificar las cuentas de los administradores autorizados para gestionar el sistema. Se encuentra relacionado con el requisito arquitectónicamente significativo RF028.
* Servicio de Estadísticas: Este microservicio accederá a la base de datos de pedidos finalizados, permitiendo registrar pedidos completados o cancelados y generar estadísticas de calidad del servicio que serán accedidas por los administradores. Se encuentra relacionado con el requisito arquitectónicamente significativo RF018.
* Servicio de Gestión de Franquicias: Este microservicio accederá a la base de datos de franquicias, permitiendo agregar y eliminar franquicias. Se encuentra relacionado con el requisito arquitectónicamente significativo RF025.
* Servicio de Notificaciones: Este microservicio se integrará con sistemas externos de mensajería para enviarle mensajes asincrónicos al usuario, cuando sea necesario. Se encuentra relacionado con el requisito arquitectónicamente significativo RF010.
* Servicio de Pago: Este microservicio se integrará con sistemas externos de pagos en línea para realizar los pagos y reembolsos de pedidos. Se encuentra relacionado con los requisitos arquitectónicamente significativos RF007, RF047, RNF009.
* Servicio de Gestión de Pedidos: Este microservicio accederá a la base de datos de pedidos en curso y se encargará de manejar el seguimiento y gestión de los pedidos en curso. Se encuentra relacionado con los requisitos arquitectónicamente significativos RF001, RF002, RF003, RF005, RF007, RF008, RF009, RF011, RF012, RF030, RF031, RF034, RF045, RF046, RF049, RF051.
* Servicio de Catálogo: Este microservicio accederá a la base de datos del catálogo de la tienda, permitiendo gestionar y acceder a los productos del catálogo de la tienda. Se encuentra relacionado con los requisitos arquitectónicamente significativos RF003, RF033, RF034.
* Servicio de Inventario: Este microservicio accederá a la base de datos de inventario, permitiendo gestionar el inventario de cada tienda y revisar si se tienen suficientes existencias de cada producto. Se encuentra relacionado con los requisitos arquitectónicamente significativos RF003, RF032, RF034.
* Servicio de Disponibilidad de Franquicias: Este microservicio accederá a la base de datos de franquicias disponibles, permitiendo buscar la franquicia más adecuada para asignar o reasignar un pedido. Se encuentra relacionado con los requisitos arquitectónicamente significativos RF001, RF002, RF031, RF045.



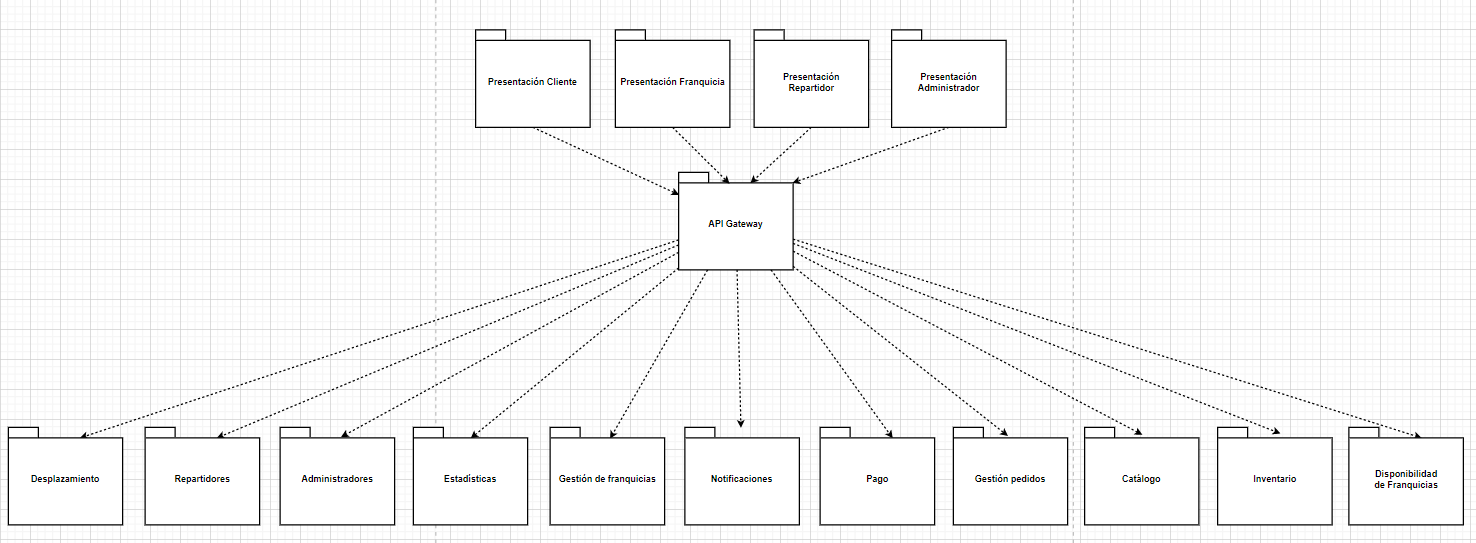
Para facilitar el acceso a los diversos microservicios, se decidió utilizar el patrón API Gateway. Este patrón implica agregar un nuevo componente, el API Gateway, que sirve como único punto de entrada para las solicitudes de la capa de presentación del sistema, encargándose de delegar las solicitudes a cada uno de los microservicios. En algunos casos, una única solicitud recibida implicará realizar solicitudes a más de un microservicio. El uso de este patrón permite separar de la presentación la responsabilidad de enviar solicitudes a los microservicios específicos, beneficiando la modificabilidad (RNF013).

Se decidió que al API Gateway se le asignarán las responsabilidades de autenticación y autorización, dado que es un elemento transversal que deberá tenerse en cuenta para la gran mayoría de los microservicios (RF027).

Cabe mencionar que se identificaron cuatro tipos de presentación independientes que accederán al API Gateway, una por cada tipo de usuario que accede al sistema: Cliente, Franquicia, Repartidor y Administrador (incluye administrador y súper administrador). Es importante aclarar que la presentación de Cliente también se utiliza para el empleado de la franquicia que atiende las llamadas telefónicas. Esta división de la presentación también beneficia la modificabilidad (RNF013).

**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.

****

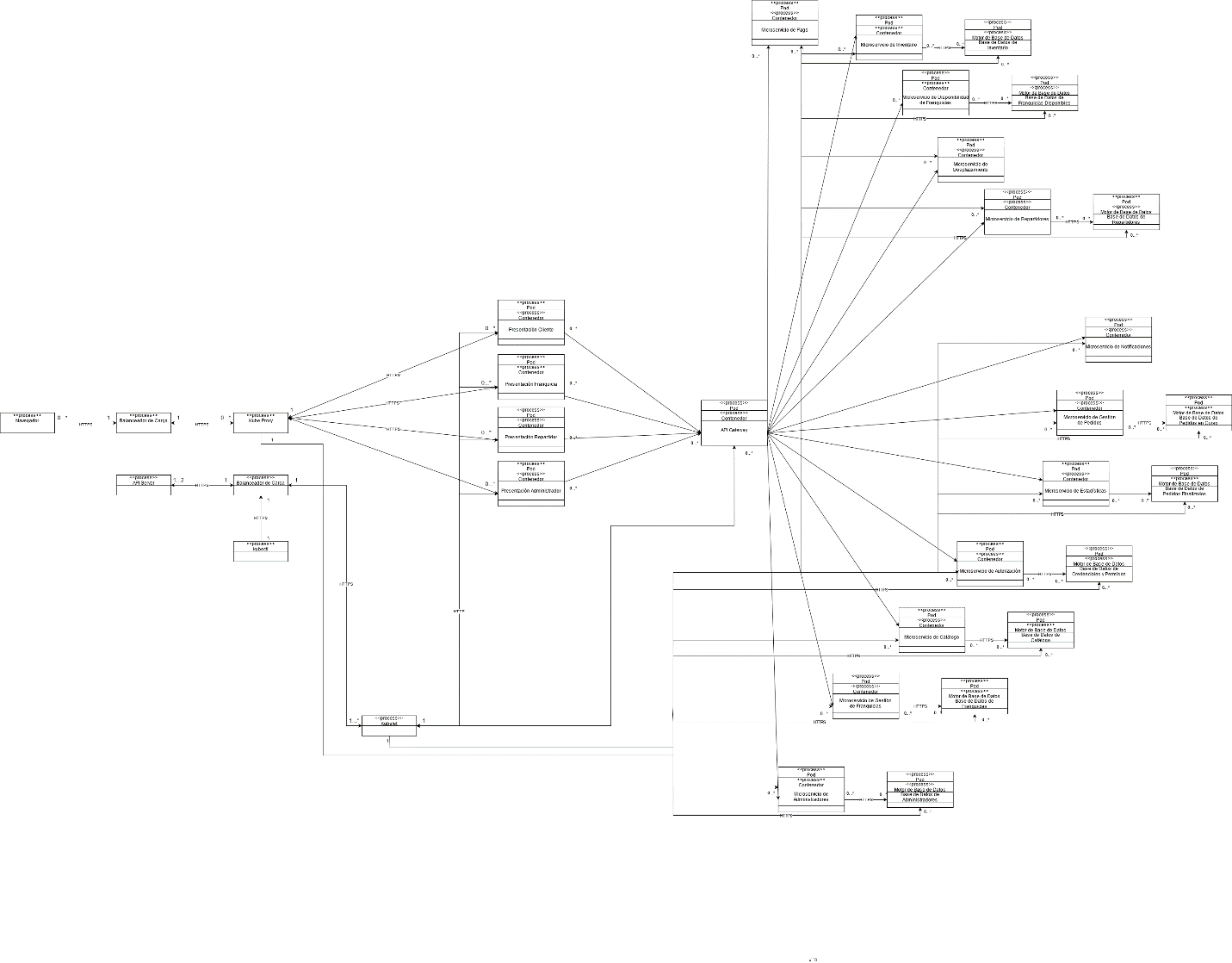
**Vista de Proceso**

Se decidió utilizar Kubernetes para el despliegue del sistema, debido a que permite orquestar diversos contenedores desplegados en diferentes nodos de manera dinámica. Esto tiene algunas implicaciones importantes para la vista de proceso del sistema. En primer lugar, los componentes de la vista lógica se ejecutarán dentro de contenedores que estarán, a su vez, dentro de un “pod”. Cada uno de estos pods podrá ser escalado de manera independiente según sea necesario, por lo que se decidió que cada pod va a contener solamente un componente de la vista lógica. Las bases de datos también se ejecutarán dentro de pods.

También es importante tener en cuenta que el uso de Kubernetes implica la inclusión de procesos involucrados en su funcionamiento. En primer lugar, se incluyó el proceso API Server, que es el encargado de la configuración, monitoreo y coordinación de los demás nodos del cluster de Kubernetes. Se determinó que deberá haber más de uno de estos procesos para evitar que este se convierta en un punto único de falla. El proceso del API Server puede recibir solicitudes del proceso kubectl, el cual puede modificar la configuración del cluster en tiempo de ejecución.

Cada uno de los demás nodos del cluster deberá tener un proceso kubelet, un proceso kube proxy y cero o más pods ejecutándose. El proceso kubelet se encarga de comunicarse con el API Server para actualizar la configuración del nodo, mientras que el proceso kube proxy se encarga del enrutamiento de paquetes del nodo. Por lo tanto, estos dos procesos se comunicarán con todos los pods del nodo.

A partir de esto, para que los pods reciban las solicitudes de los clientes, se estableció que el proceso del navegador se comunicará con un balanceador de carga que se encargará de enrutar las solicitudes al proceso kube proxy de los diferentes nodos.

****

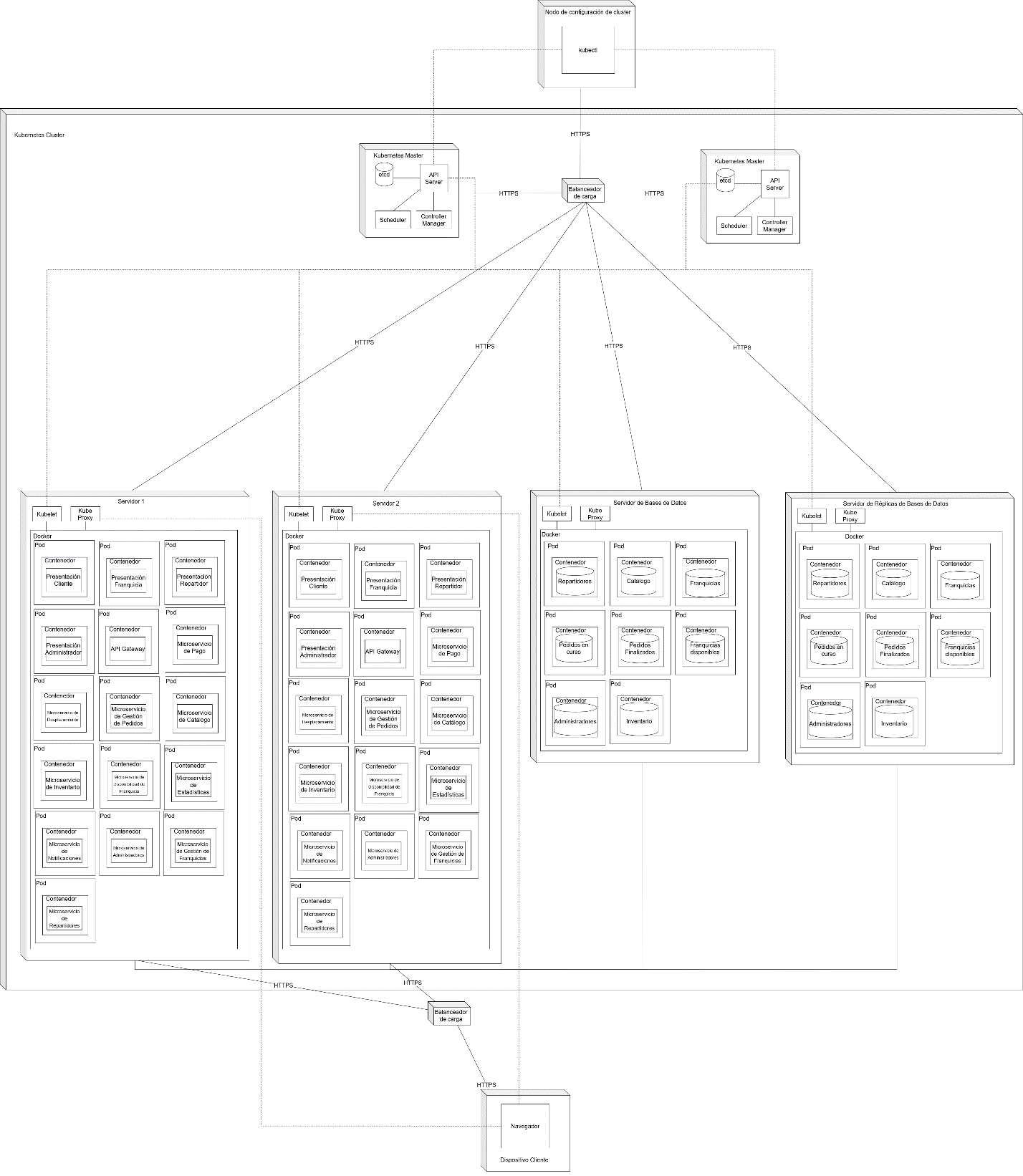
**Vista Física**

El uso de Kubernetes para el despliegue del sistema tiene un gran impacto sobre la vista física del mismo, dado que deberán agruparse los procesos descritos en la vista de proceso para ejecutarse en diferentes nodos. En primer lugar, se decidió que se utilizarán dos nodos Master para ejecutar el API Server, los cuales recibirán solicitudes del kubectl desde un nodo que se encuentra fuera del cluster, el cual permitirá modificar la configuración del cluster en tiempo de ejecución.

En cuanto a los nodos en los que se ejecutarán los pods, se decidió utilizar, como mínimo cuatro nodos diferentes: dos nodos en los que se ejecutarán los pods de los componentes de presentación, API Gateway y microservicios, y dos nodos en los que se mantendrán las bases de datos del sistema. Se decidió que uno de estos nodos de bases de datos permitirá operaciones de lectura y escritura mientras que el otro tendrá réplicas de bases de datos que solo aceptará operaciones de lectura. Este enfoque busca permitir que las bases de datos sean escalables sin comprometer su consistencia.

La idea de agrupar los componentes de presentación, el API Gateway y los microservicios en el mismo nodo busca reducir la cantidad de solicitudes entre nodos. Dado que el API Gateway accede a todos los microservicios, ubicar estos componentes en nodos diferentes podría aumentar bastante la latencia. De esta manera, se busca que solo se tenga que acceder a nodos diferentes para el acceso a las bases de datos.

En el diagrama de despliegue se ilustra únicamente la agrupación de los pods en los nodos, pero se busca que, a través de los servicios de Kubernetes, el número de instancias de cada uno varíe dependiendo de la carga de trabajo registrada en cada momento. Asimismo, sería posible, e incluso deseable, agregar más nodos al cluster, bien sean de componentes o de réplicas de base de datos.

****

1. **Verificación de Requisitos**

Se considera que, en esta iteración, la arquitectura de microservicios permitió satisfacer completamente los requisitos de disponibilidad (RNF001, RNF002 y RNF003). Dado que en esta iteración se realizó principalmente la delegación de requisitos arquitectónicamente funcionales restantes a los subcomponentes identificados en la vista lógica, estos deberán ser abordados con mayor detalle en las siguientes iteraciones.

**Iteración 2**

1. **Elemento escogido**

API Gateway

1. **Requisitos arquitectónicamente significativos del elemento escogido**

Esta iteración se enfocará en los requisitos arquitectónicamente significativos asignados previamente al API Gateway, que incluyen la autenticación y autorización asociadas al requisito RF027, las cuales guardan también relación con el atributo de calidad de seguridad.

1. **Solución de diseño**

**Vista Lógica**

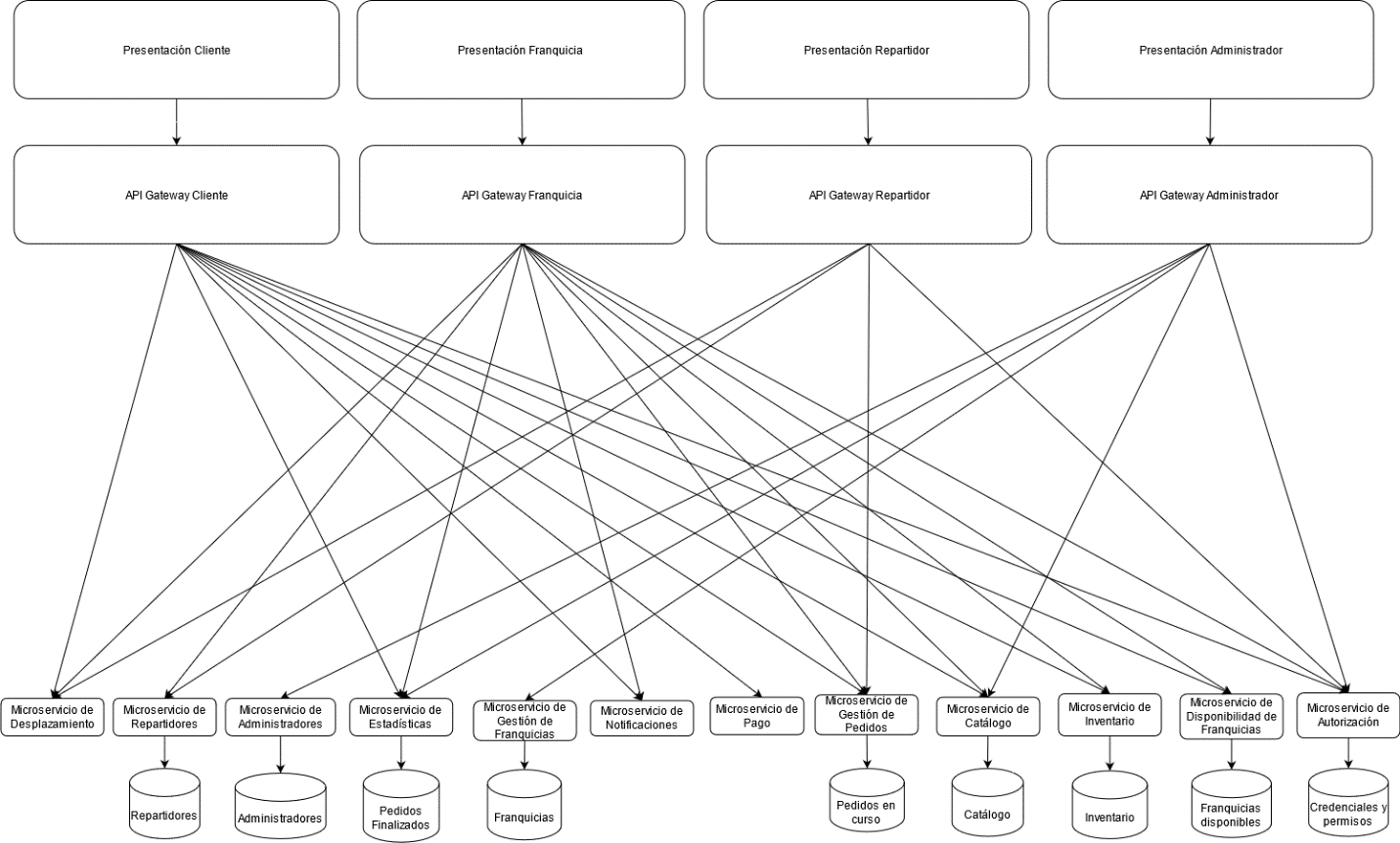
En primer lugar, como sugiere el requisito funcional RF027, se buscará proporcionarle seguridad al sistema a través de las tácticas arquitectónicas de autenticar actores y autorizar actores. Si bien en la iteración anterior se determinó que esta responsabilidad recaería sobre el API Gateway, en esta iteración se decidió asignarle esta responsabilidad a un nuevo microservicio de autorización, que cumplirá la función de servidor de autorización.

Este nuevo microservicio será el encargado de autenticar usuarios y generar los tokens correspondientes, por lo que accederá a la base de datos de credenciales y permisos. Cabe mencionar que, para una protección mayor contra el acceso no autorizado, esta información se guardará encriptada, aplicando la táctica de encriptación de datos.

Cuando un usuario intente autenticarse, el API Gateway le enviará una solicitud al servidor de autorización, que revisará si los datos de autenticación son correctos. En caso de ser así, el servidor de autorización devolverá el token, en donde se indican los permisos del usuario. De ahí en adelante, todas las solicitudes del usuario deberán incluir este token.

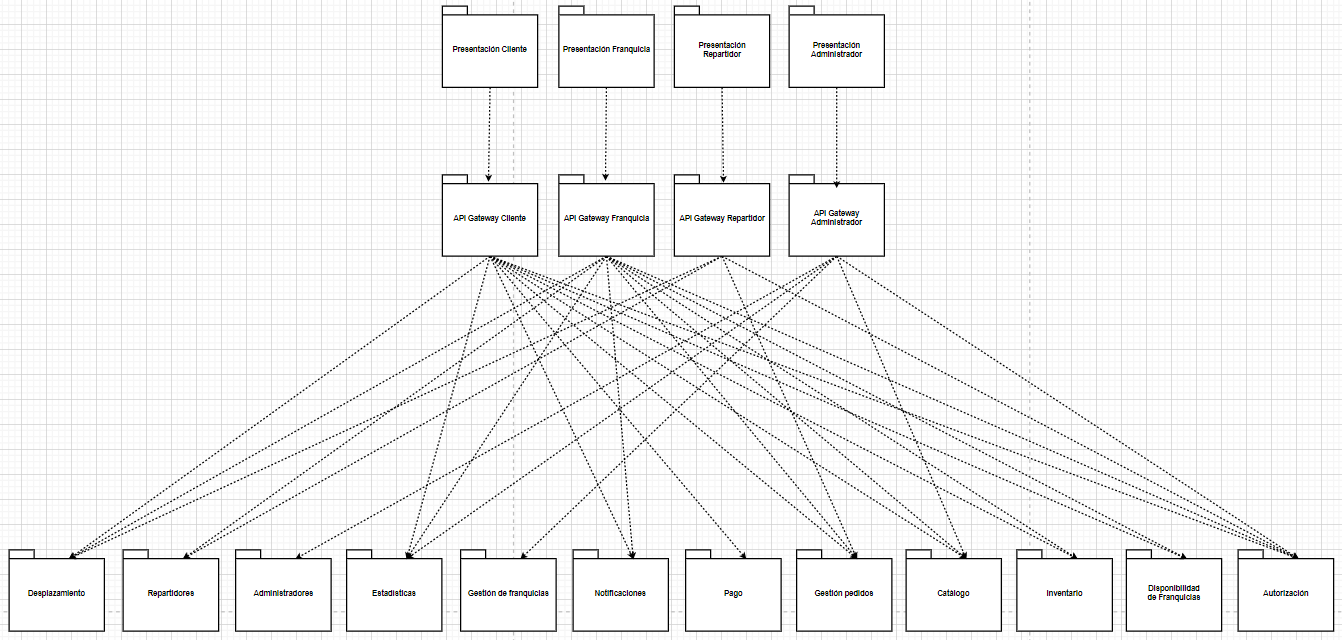
Algunos microservicios requerirán autorización por lo que antes de llamar estos servicios, el API Gateway validará el token y verificará que el usuario tenga permisos para acceder a ese servicio y, en caso de que esto se cumpla, le enviará la solicitud con ese mismo token al microservicio, el cual también deberá encargase de validar el token.

Como se puede observar en el siguiente diagrama, en esta iteración también se tomó la decisión de dividir el API Gateway en componentes más pequeños, dado que se identificó que su tamaño era demasiado grande y poseía muchas responsabilidades que no se encontraban lo suficientemente relacionadas. A partir de esto, se aplicaron las tácticas de dividir módulo y aumentar coherencia semántica, dividiendo el API Gateway en cuatro, uno por cada tipo de presentación. Cabe mencionar que algunos servicios siguen siendo compartidos por los diferentes API Gateways.



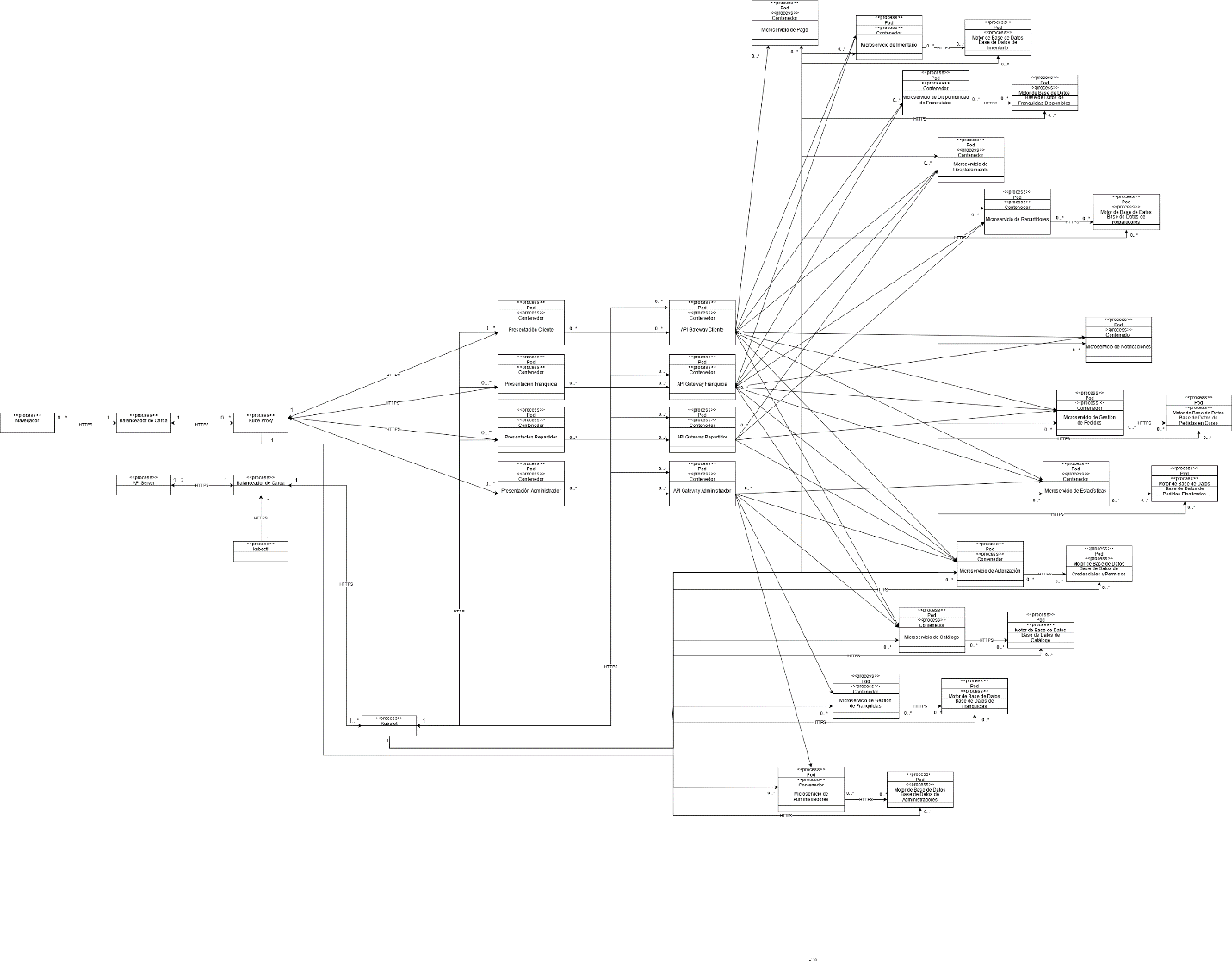
**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.

****

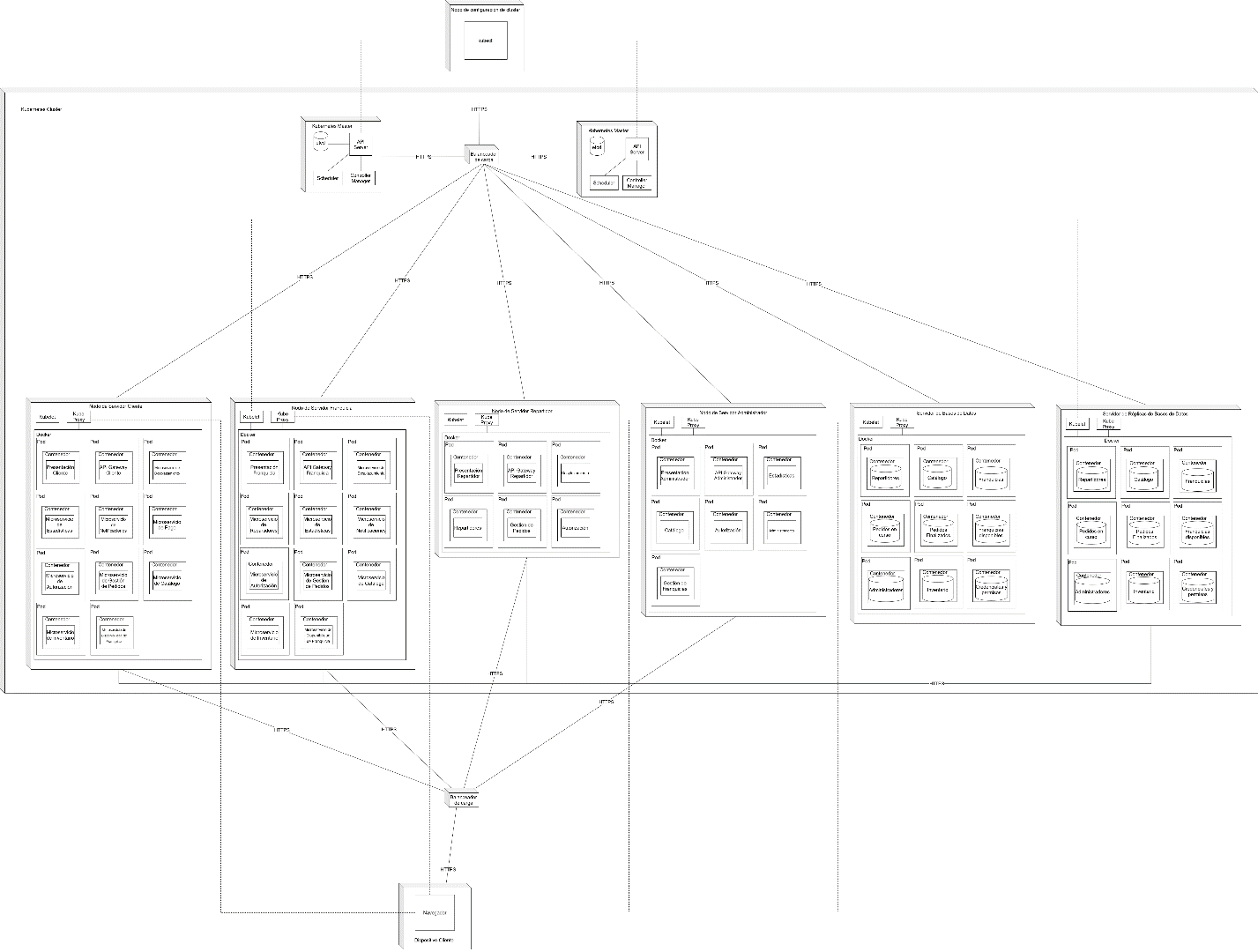
**Vista de Proceso**

La división de los API Gateways implicó la inclusión de los procesos correspondientes a los cuatro tipos de API Gateways en la vista de procesos del sistema, como se puede observar en el diagrama de proceso.

****

**Vista Física**

El dividir el API Gateway en cuatro componentes con responsabilidades diferentes permitió identificar la oportunidad de agrupar los pods en nodos de acuerdo al tipo de cliente. De esta manera, se obtuvieron cuatro nodos en los que se ejecutarán los pods de los componentes del sistema: una para cliente, uno para franquicia, uno para repartidor y uno para administrador. Cada uno de estos nodos tendrá el componente de presentación y API Gateway correspondientes, e incluirá también todos los microservicios utilizados por el API Gateway, de tal manera que se evite tener que hacer solicitudes a otros nodos. Es decir, se espera que en cada nodo estén todos los componentes necesarios para responder a una solicitud de ese tipo de cliente, excepto las bases de datos, que se siguen manteniendo en nodos aparte. Al igual que antes, se espera que la cantidad de instancias de cada uno de estos componentes varíe de acuerdo a la carga de trabajo.

****

1. **Verificación de Requisitos**

Se considera que, en esta iteración, los requisitos escogidos fueron delegados a los componentes en los que se dividió el API Gateway, por lo que será necesario abordarlos en iteraciones posteriores.

**Iteración 3**

1. **Elemento escogido**

Presentación. Dado que la estructura general de las cuatro presentaciones establecidas es a grandes rasgos la misma, se refinarán todas en esta iteración.

1. **Requisitos arquitectónicamente significativos del elemento escogido**

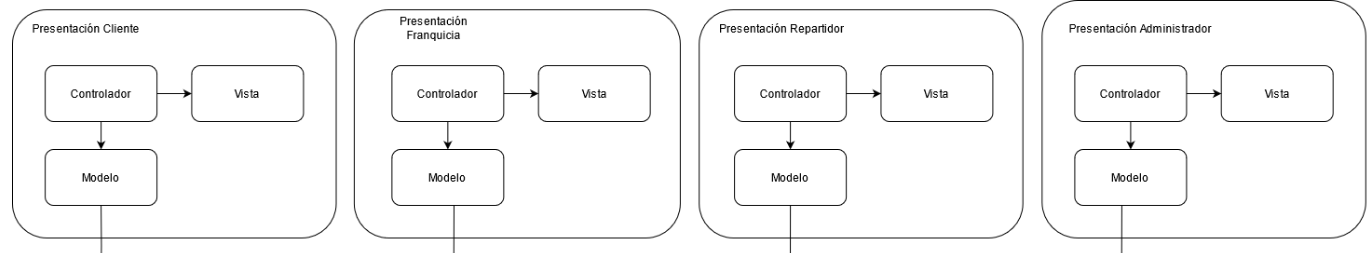
El requisito arquitectónicamente significativo más relevante para la presentación es el RNF004, el cual hace referencia a la adaptabilidad del software en dispositivos móviles, por lo que esta iteración se enfocará en cumplirlo.

1. **Solución de diseño**

**Vista Lógica**

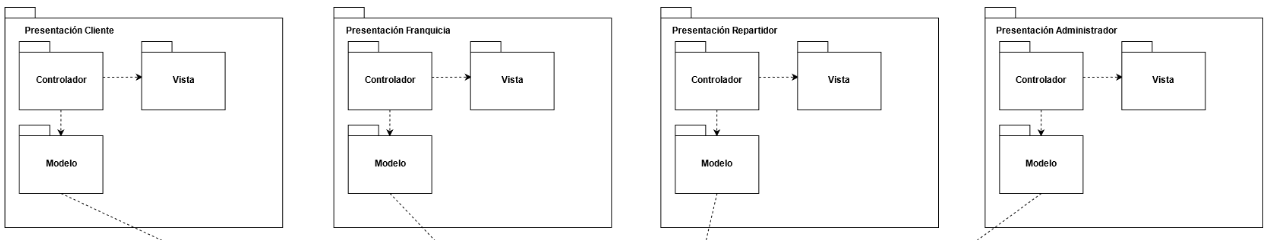
Se decidió que se la presentación se implementará en todos los casos como una aplicación web con cliente ligero. Para esto, se decidió utilizar el patrón MVC, el cual permite dividir las responsabilidades de los componentes. A partir de esto, como se puede observar en el siguiente diagrama, cada presentación se dividió en Controlador, Vista y Modelo. Cabe mencionar que el modelo no está todo contenido en la presentación, sino que llega a abarcar también el API Gateway y los microservicios utilizados.

Cuando el Controlador reciba las solicitudes HTTPS del navegador, este realizará las operaciones necesarias sobre el Modelo y retornará la Vista correspondiente, que será desplegada en el navegador. Se buscará que las interfaces de usuario del cliente y del repartidor sean responsive, de tal manera que puedan ser visualizadas correctamente en dispositivos móviles.



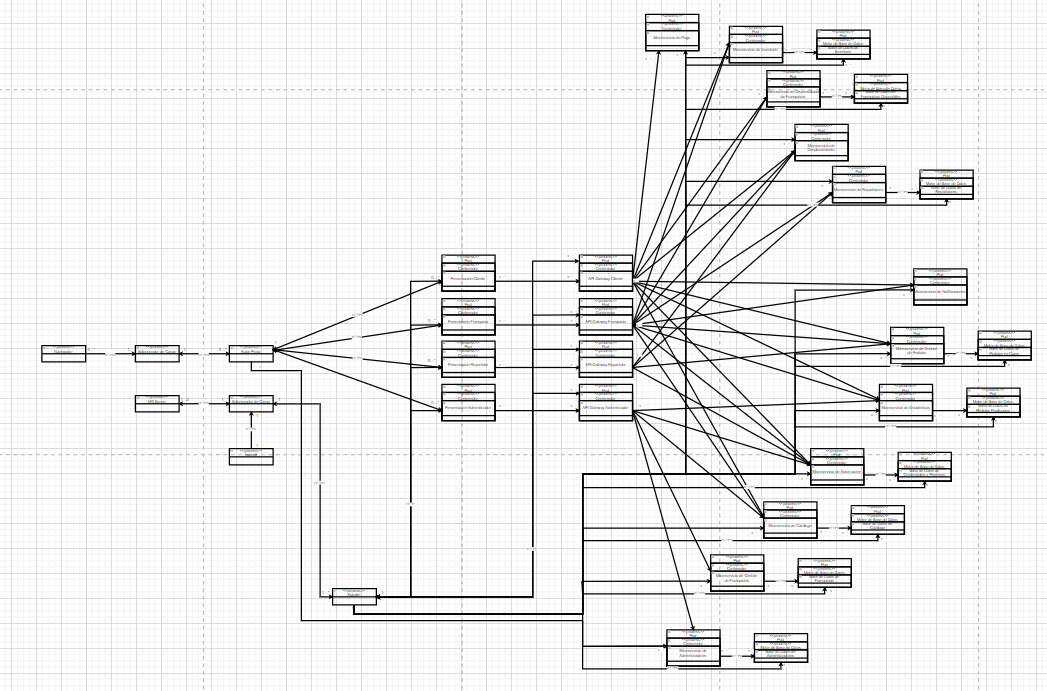
**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.



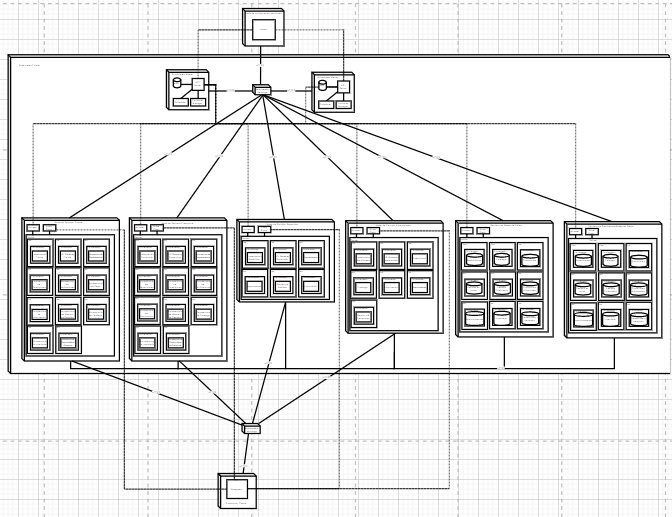
**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

****

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

****

1. **Verificación de Requisitos**

Se considera que, en esta iteración, a través de la especificación de la presentación como aplicaciones web responsive, se logró cumplir el requisito arquitecturalmente significativo RNF004, que hace referencia a la adaptabilidad del software en dispositivos móviles.

**Iteración 4**

1. **Elemento escogido**

API Gateways. Se establecerá una estructura uniforme para todos los API Gateways, por lo que se refinarán todos en esta iteración.

1. **Requisitos arquitectónicamente significativos del elemento escogido**

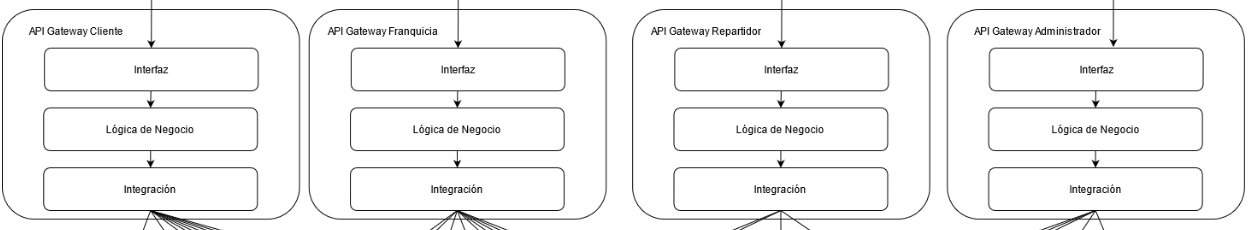
El API Gateway del cliente se encargará de recibir las solicitudes de la presentación del cliente y enviar solicitudes a los microservicios correspondientes. Por lo tanto, este elemento abarca RF001, RF002, RF003, RF005, RF007, RF011, RF016, RF018, RF027, RF033, RF034, RF047, RF049, RF051 y RNF009.

1. **Solución de diseño**

**Vista Lógica**

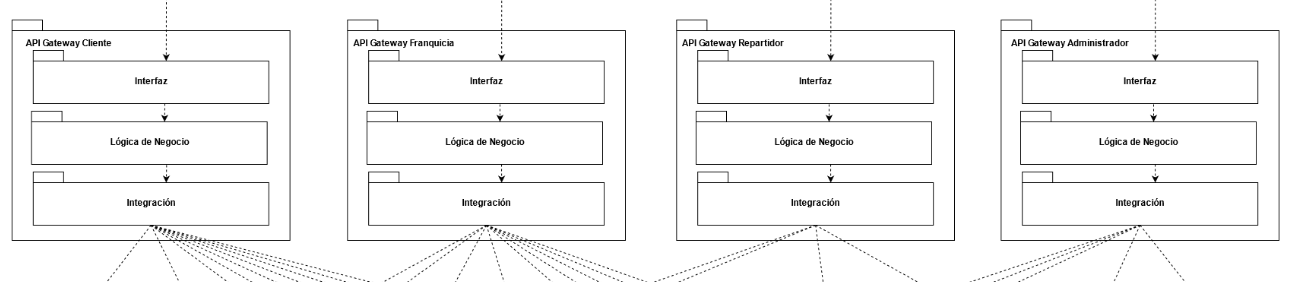
Se decidió estructurar todos los API Gateways a partir del patrón de capas, C

En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de la presentación; Lógica de Negocio, que determina qué métodos de los microservicios deben ser invocados y en qué orden, e Integración, que incluye la lógica para comunicarse con los microservicios.



**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.



**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Durante la iteración se logró continuar con el desarrollo del sistema, teniendo y manteniendo y elaborando los requisitos arquitectónicamente significativos, se considera que ninguno esta completado del todo, por lo que serán reevaluados en iteraciones posteriores.

**Iteración 5**

1. **Elemento escogido**

Microservicio de Desplazamiento.

1. **Requisitos arquitectónicamente significativos del elemento escogido**

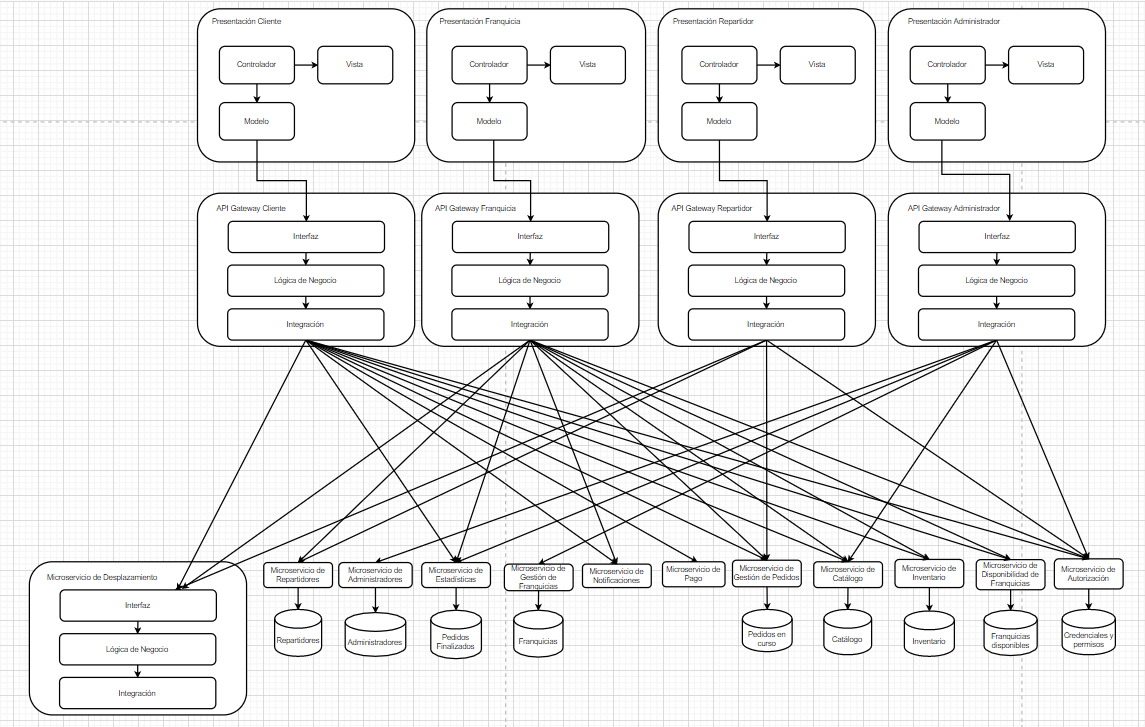
Este elemento se encuentra relacionado con RF001, RF016, RF031, RF045 RNF012 y RNF013.

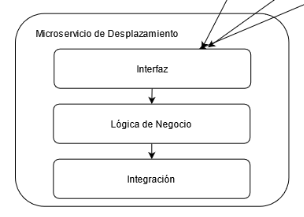
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar este microservicio a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

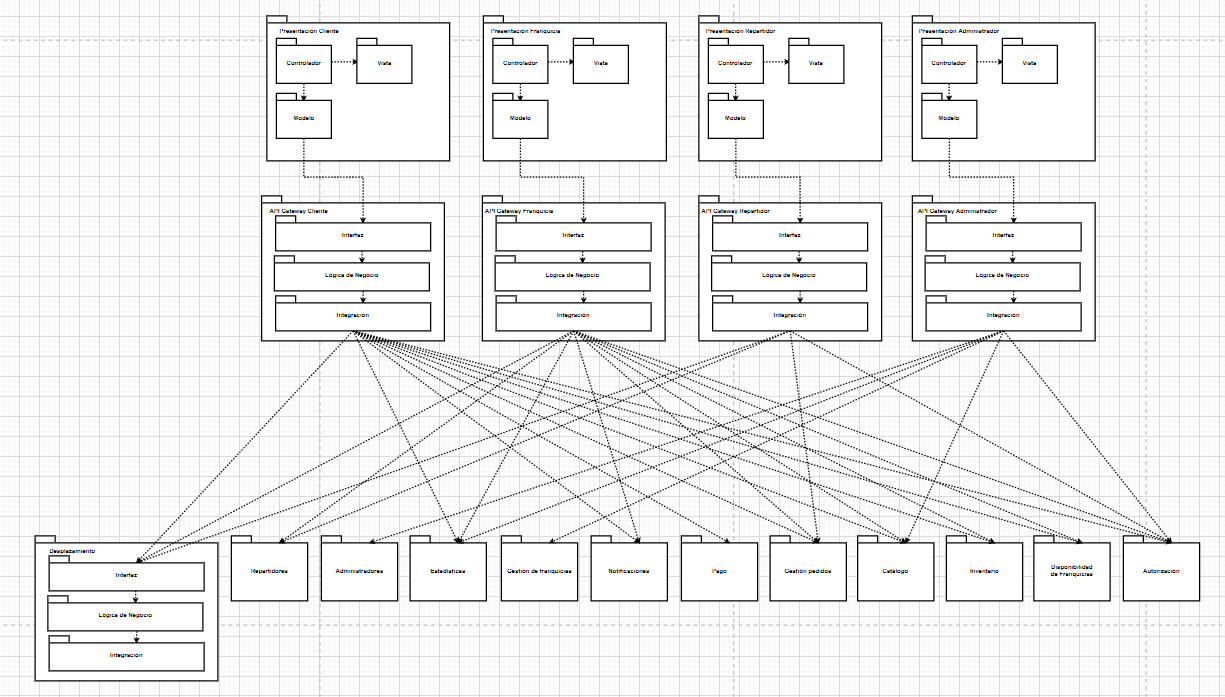
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, e Integración, que se encargará de la comunicación con los servicios externos de mapas y tráfico.

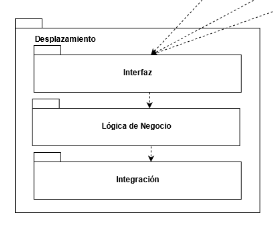




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.





**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Con esta iteración se considera el requisito RF016 completado, mientras que los demás requisitos deberán ser revisados en otras iteraciones debido a su gran abarque.

**Iteración 6**

1. **Elemento escogido**

Microservicio de Repartidores

1. **Requisitos arquitectónicamente significativos del elemento escogido**

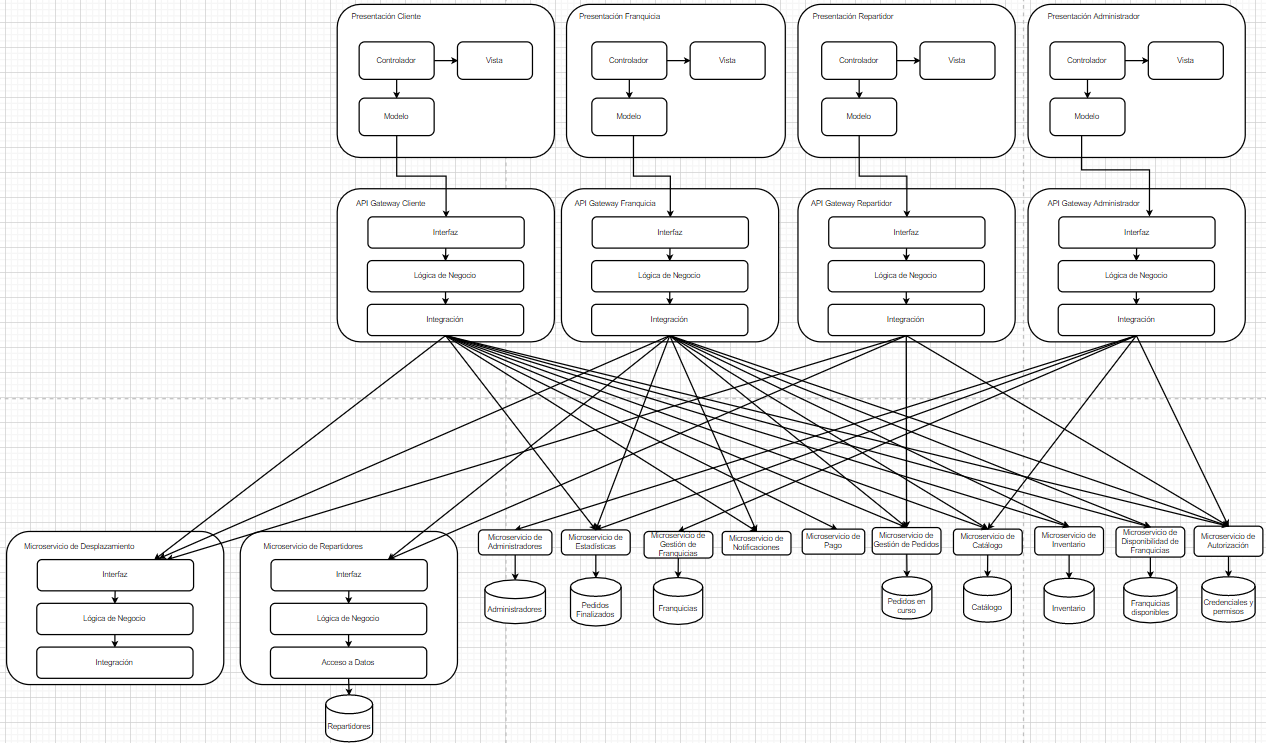
Este elemento se encuentra relacionado con RF012, RF026 RNF012 y RNF013.

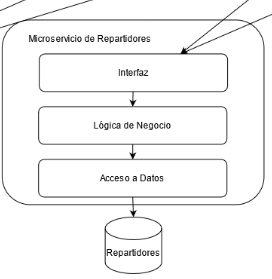
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar este microservicio a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

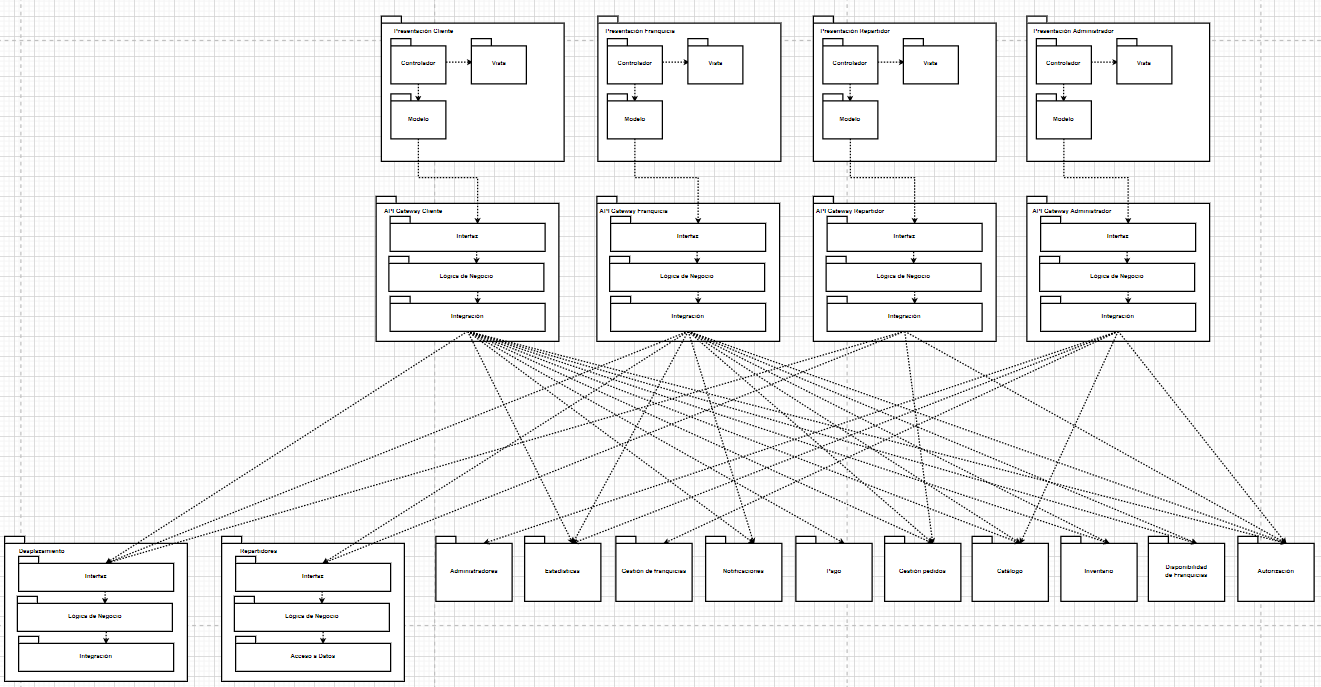
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, y Acceso a Datos, que se encargará del acceso a la base de datos de repartidores.

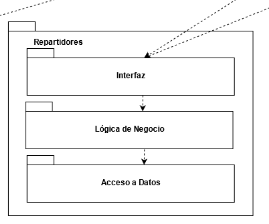




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.

****



**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Tras esta iteración se considera RF012 completado, mientras que el RF026 deberá ser revisado en iteraciones posteriores

**Iteración 7**

1. **Elemento escogido**

Microservicio de Administradores

1. **Requisitos arquitectónicamente significativos del elemento escogido**

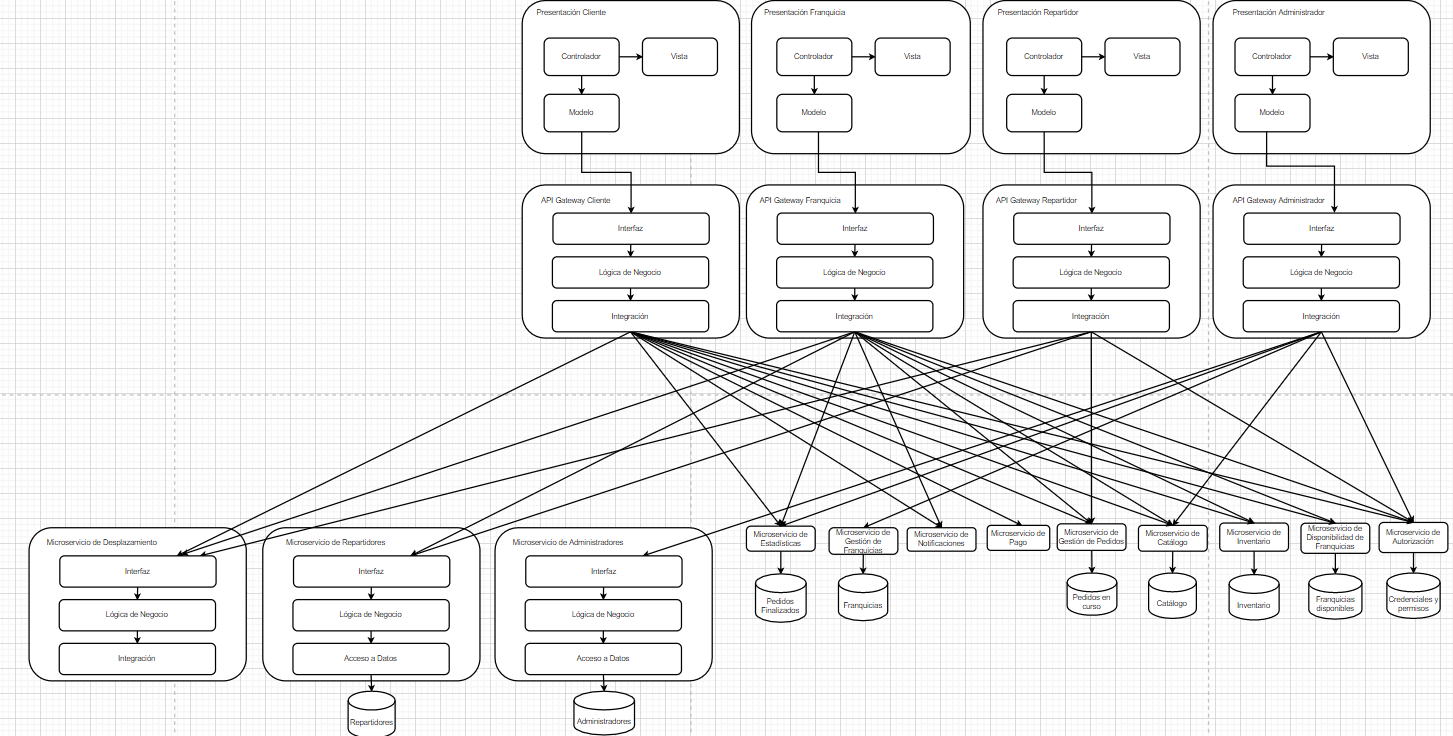
Este elemento se encuentra relacionado con RF028, RNF012 y RNF013.

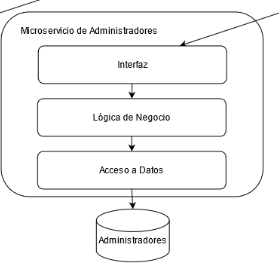
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar este microservicio a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

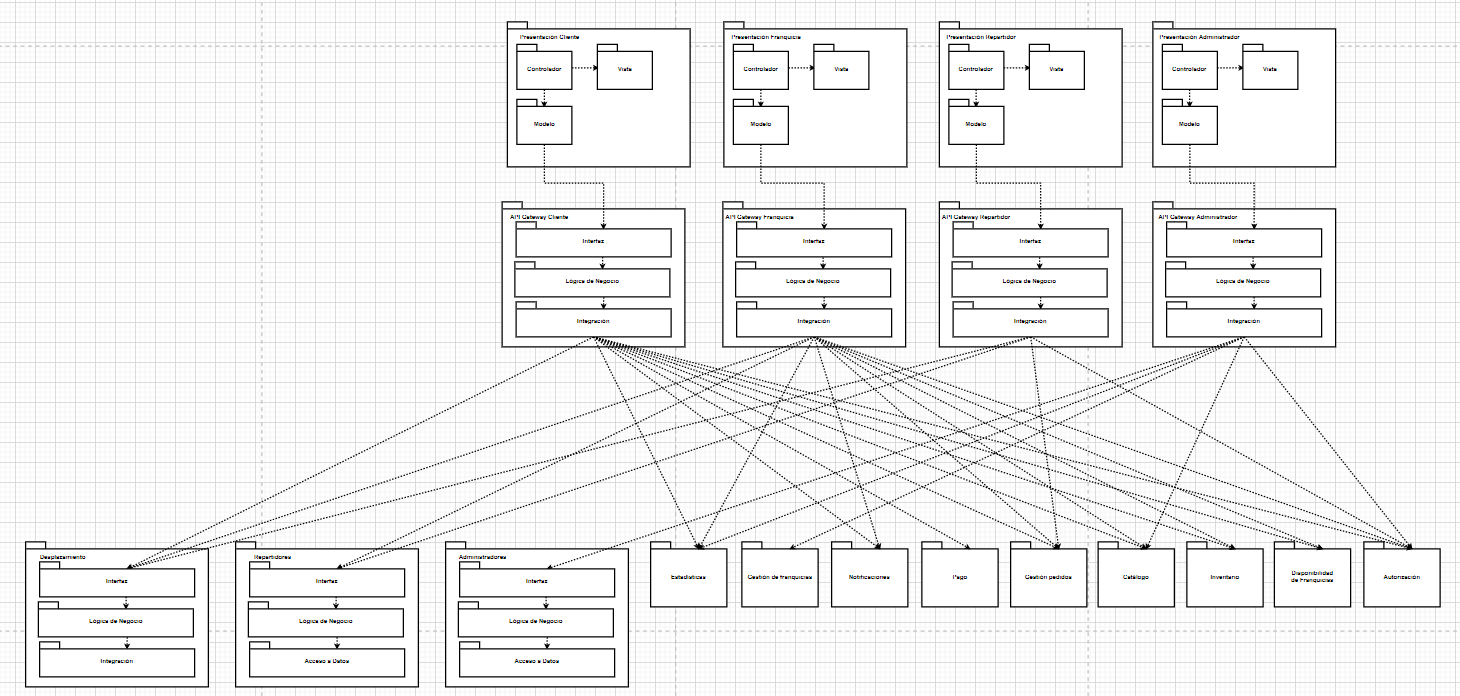
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, y Acceso a Datos, que se encargará del acceso a la base de datos de administradores.

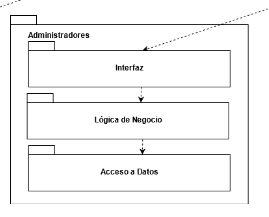




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.





**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Con esta iteración se considera el RF028 completado.

**Iteración 8**

1. **Elemento escogido**

Microservicio de Estadísticas

1. **Requisitos arquitectónicamente significativos del elemento escogido**

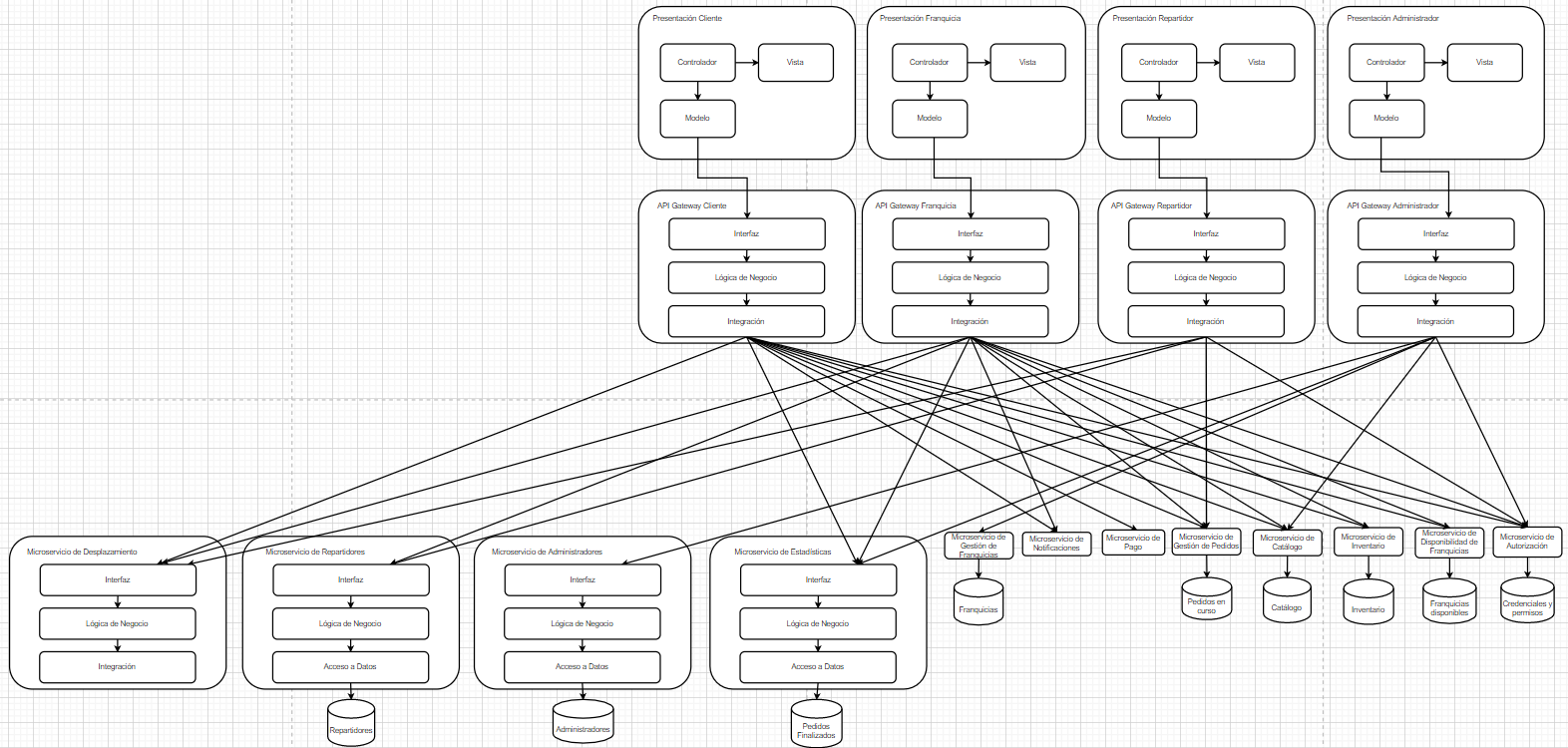
Este elemento se encuentra relacionado con RF018, RNF012 y RNF013.

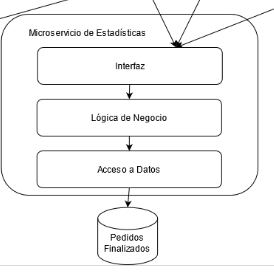
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar este microservicio a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

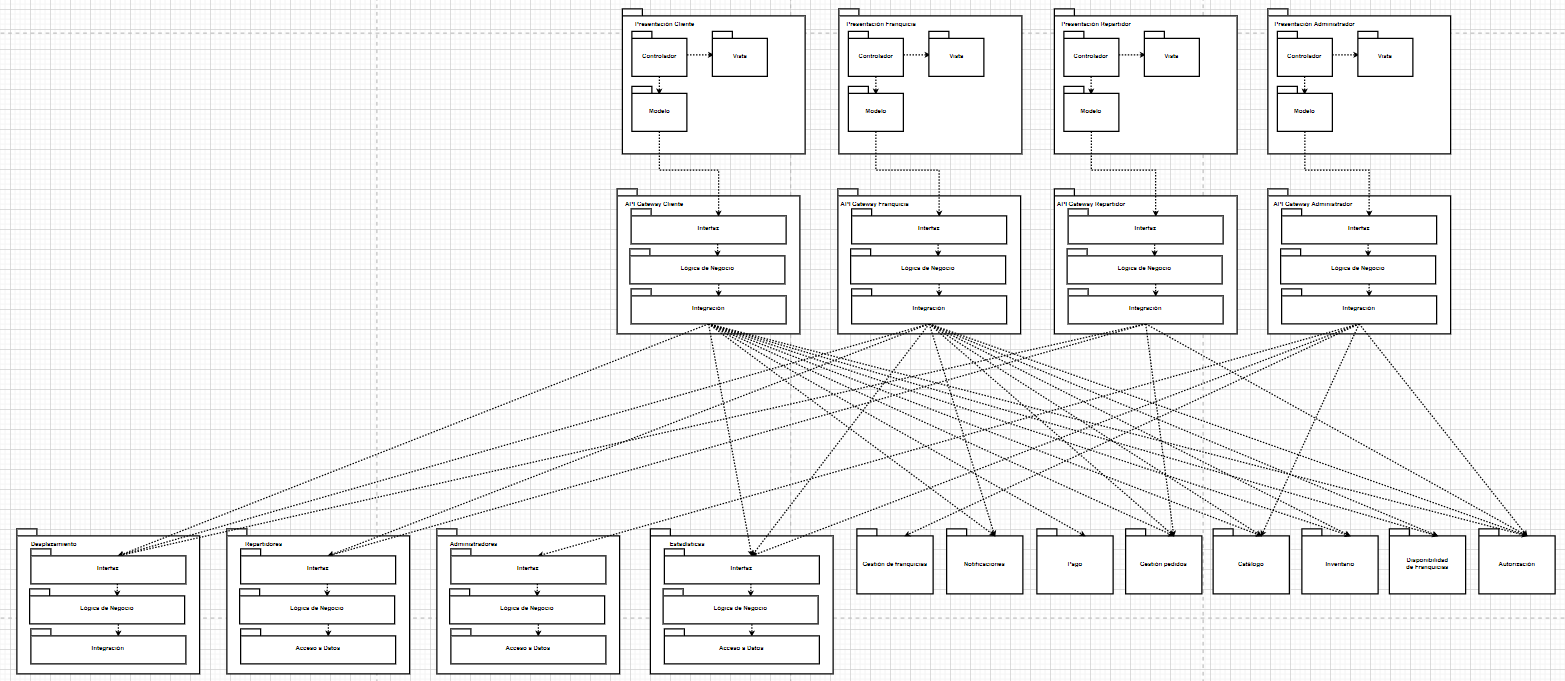
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, y Acceso a Datos, que se encargará del acceso a la base de datos de pedidos finalizados.

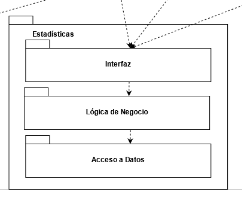




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.





**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Con esta iteración se considera el RF018 completado.

**Iteración 9**

1. **Elemento escogido**

Microservicio de Gestión de Franquicias

1. **Requisitos arquitectónicamente significativos del elemento escogido**

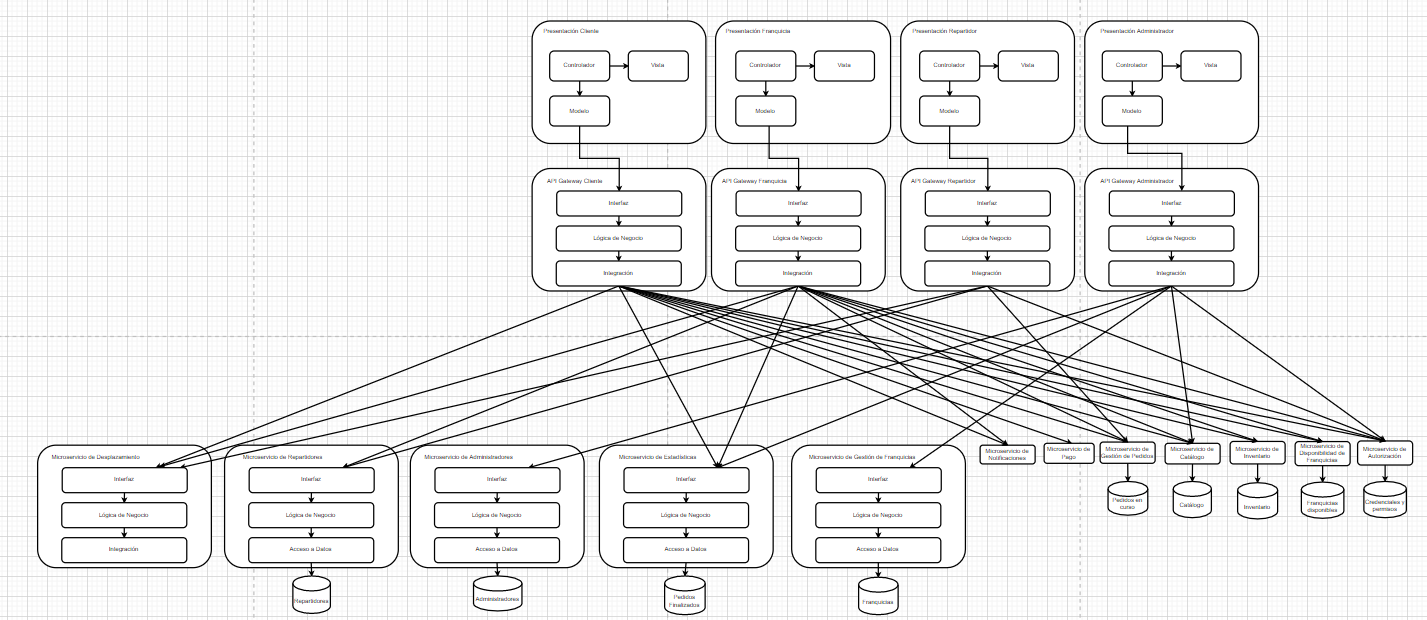
Este elemento se encuentra relacionado con el RF025, RF026, RNF012 y RNF013.

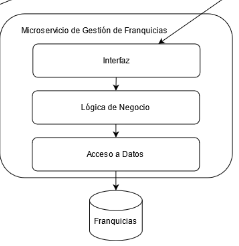
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar este microservicio a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

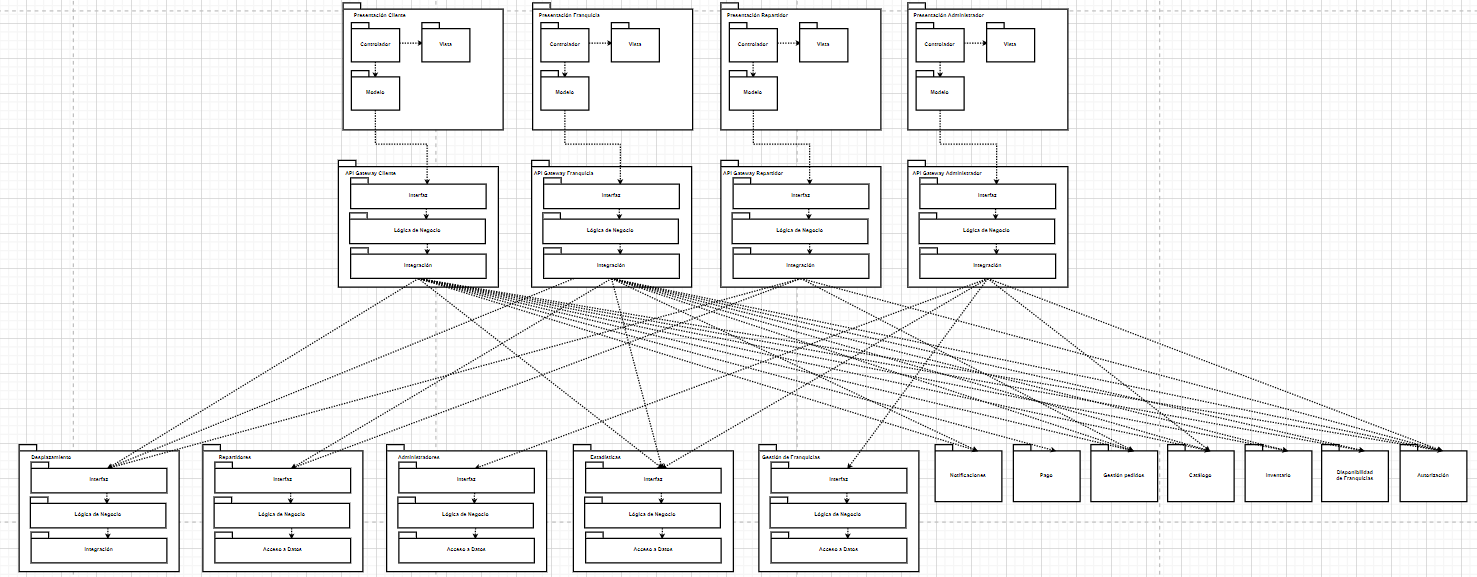
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, y Acceso a Datos, que se encargará del acceso a la base de datos de franquicias.

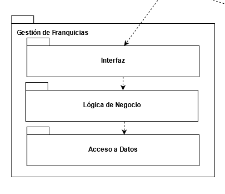




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.





**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Con esta iteración se considera el RF025 y RF026 completado.

**Iteración 10**

1. **Elemento escogido**

Microservicio de Notificaciones

1. **Requisitos arquitectónicamente significativos del elemento escogido**

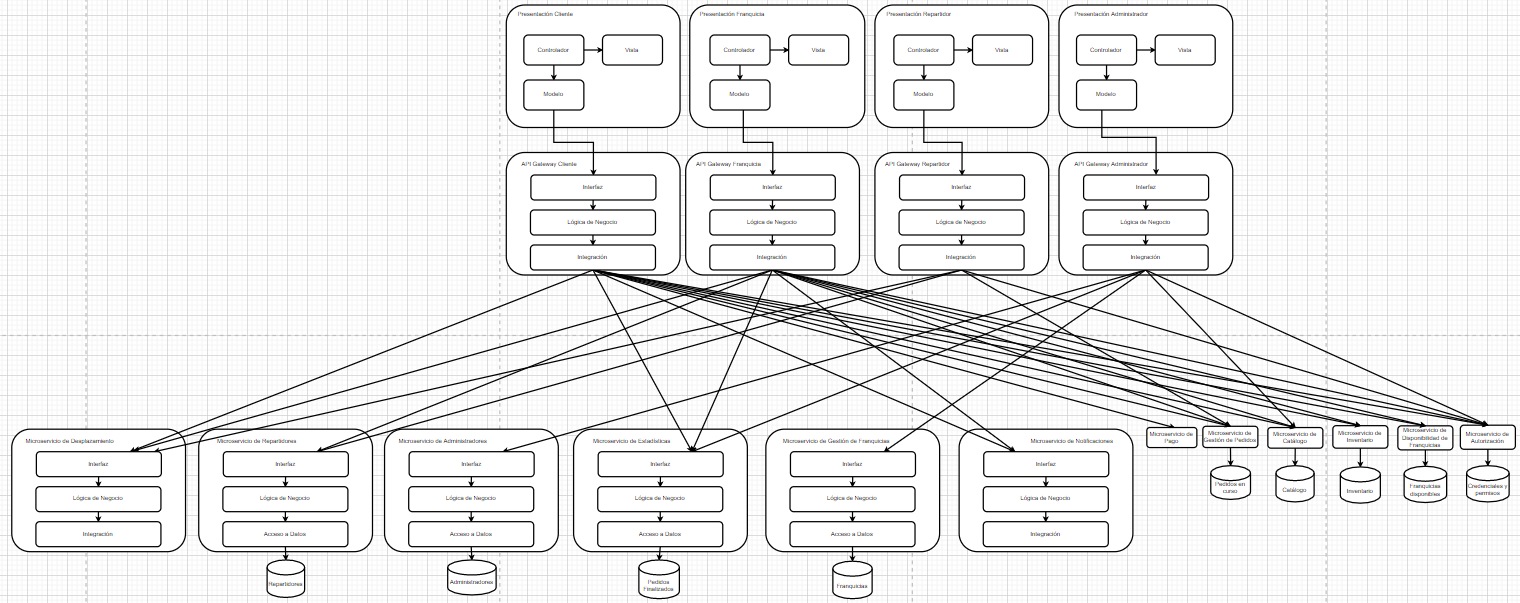
Este elemento se encuentra relacionado con el RF010, RF049, RNF012 y RNF013.

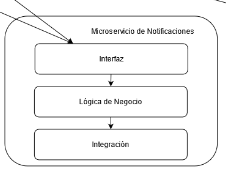
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar este microservicio a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

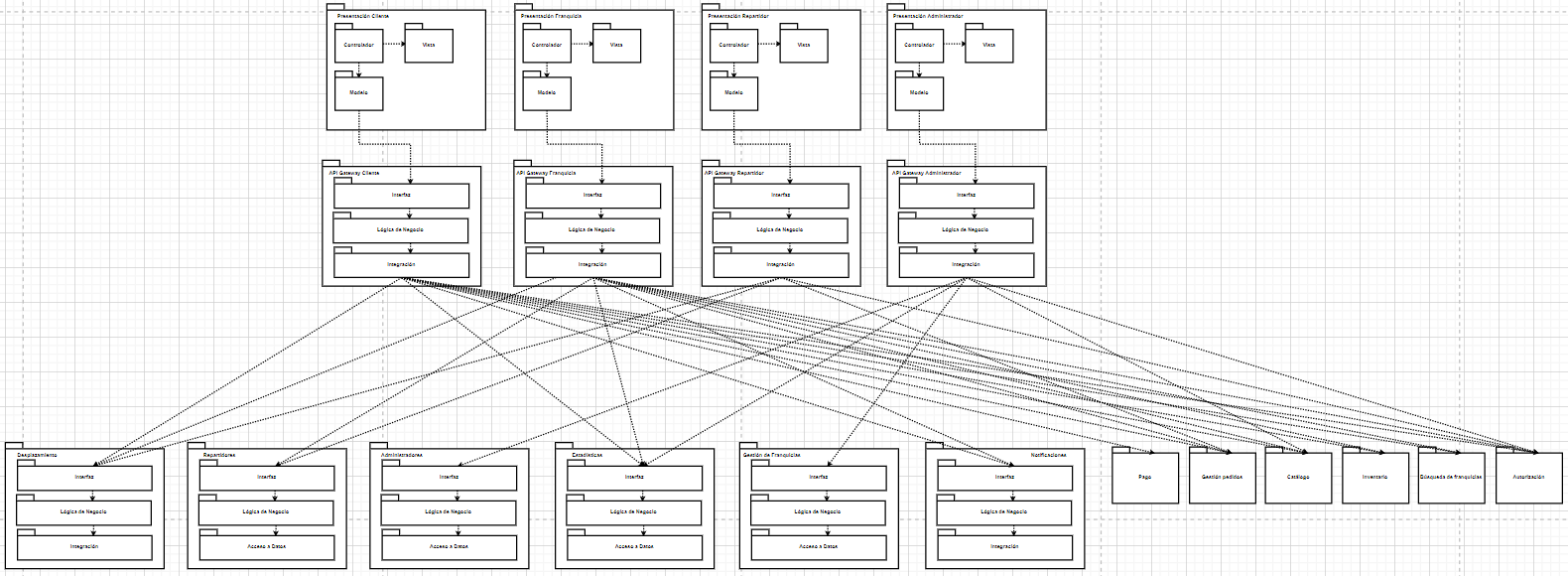
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, e Integración, que se encargará de la comunicación con los servicios externos de mensajería, como, por ejemplo, de correo electrónico.

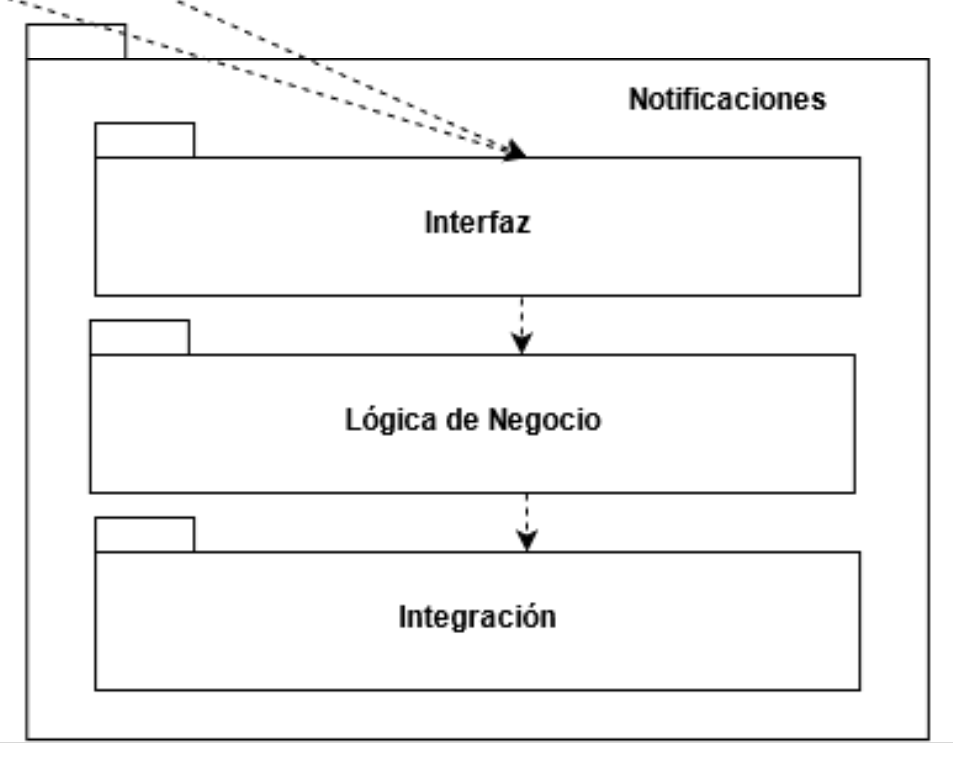




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.





**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Con esta iteración se considera el RF010 y RF049 completado.

**Iteración 11**

1. **Elemento escogido**

Microservicio de Pago

1. **Requisitos arquitectónicamente significativos del elemento escogido**

Este elemento se encuentra relacionado con RF007, RF047, RNF009, RNF012 y RNF013.

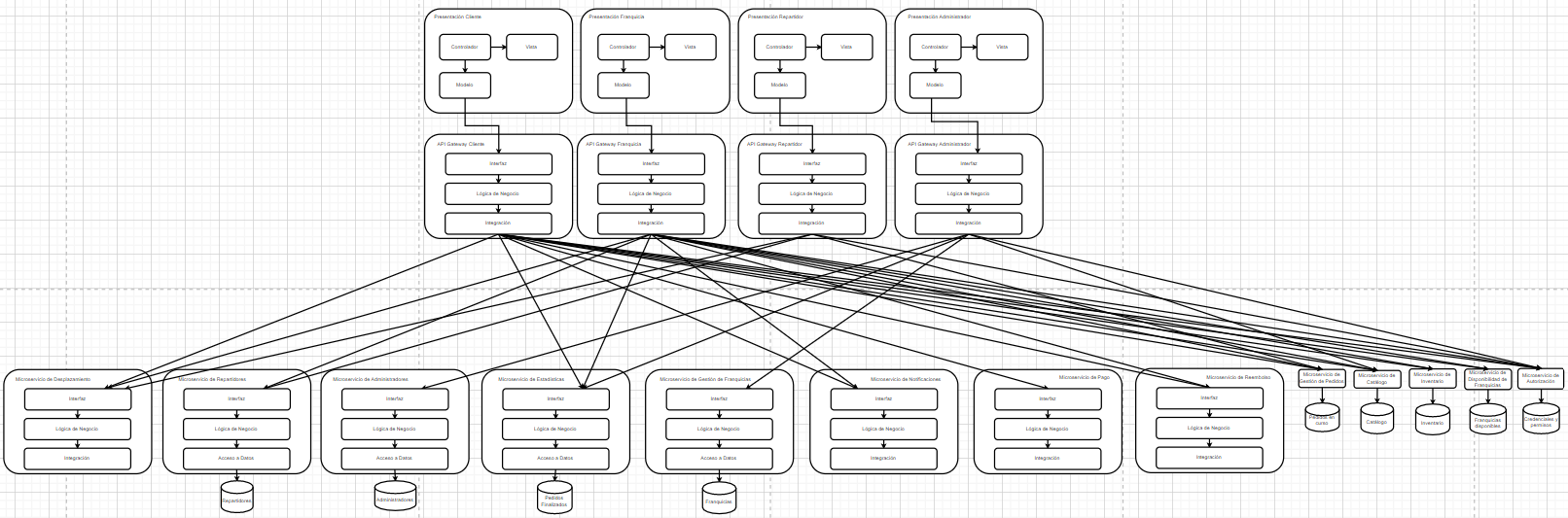
1. **Solución de diseño**

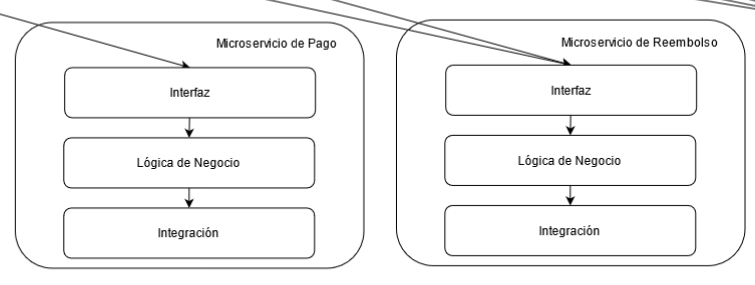
**Vista Lógica**

En primer lugar, se decidió dividir este microservicio en dos microservicios diferentes: Microservicio de Pagos y Microservicio de Reembolso. Esto se hizo con el objetivo de separar responsabilidades, dado que es probable que los mecanismos de pago y de reembolso sean similares. Sin embargo, ambos microservicios tendrán la misma estructura general.

Se decidió estructurar estos microservicios a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

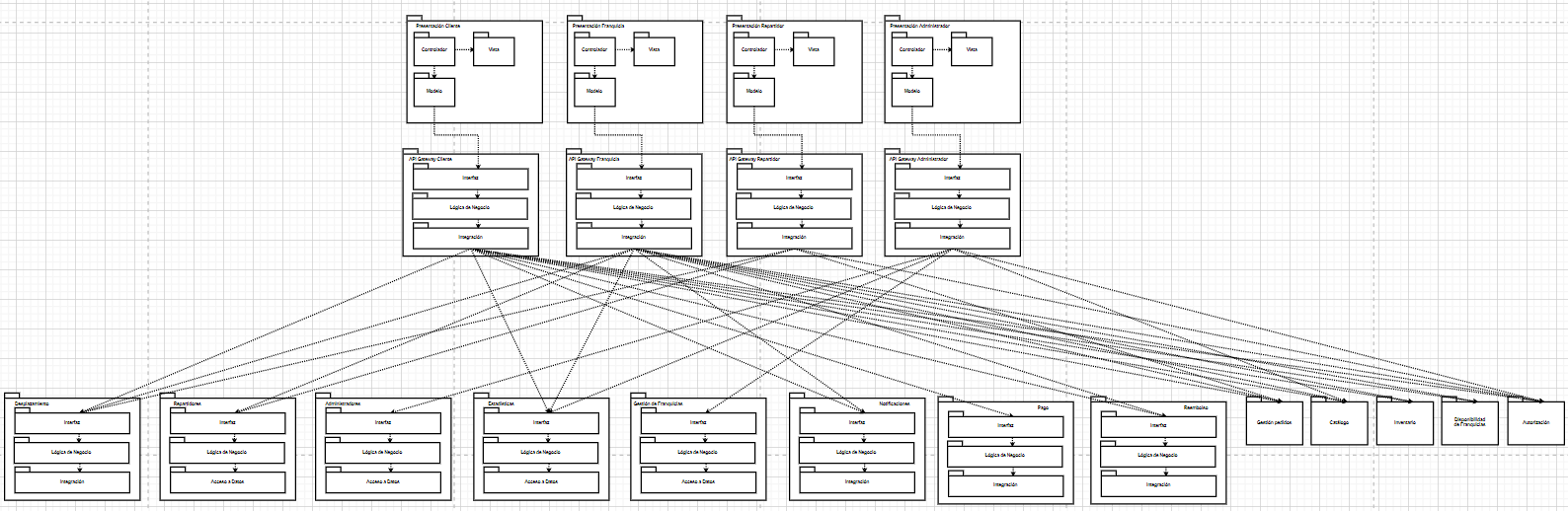
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, e Integración, que se encargará de la comunicación con los servicios externos de pagos.





**Vista de Desarrollo**

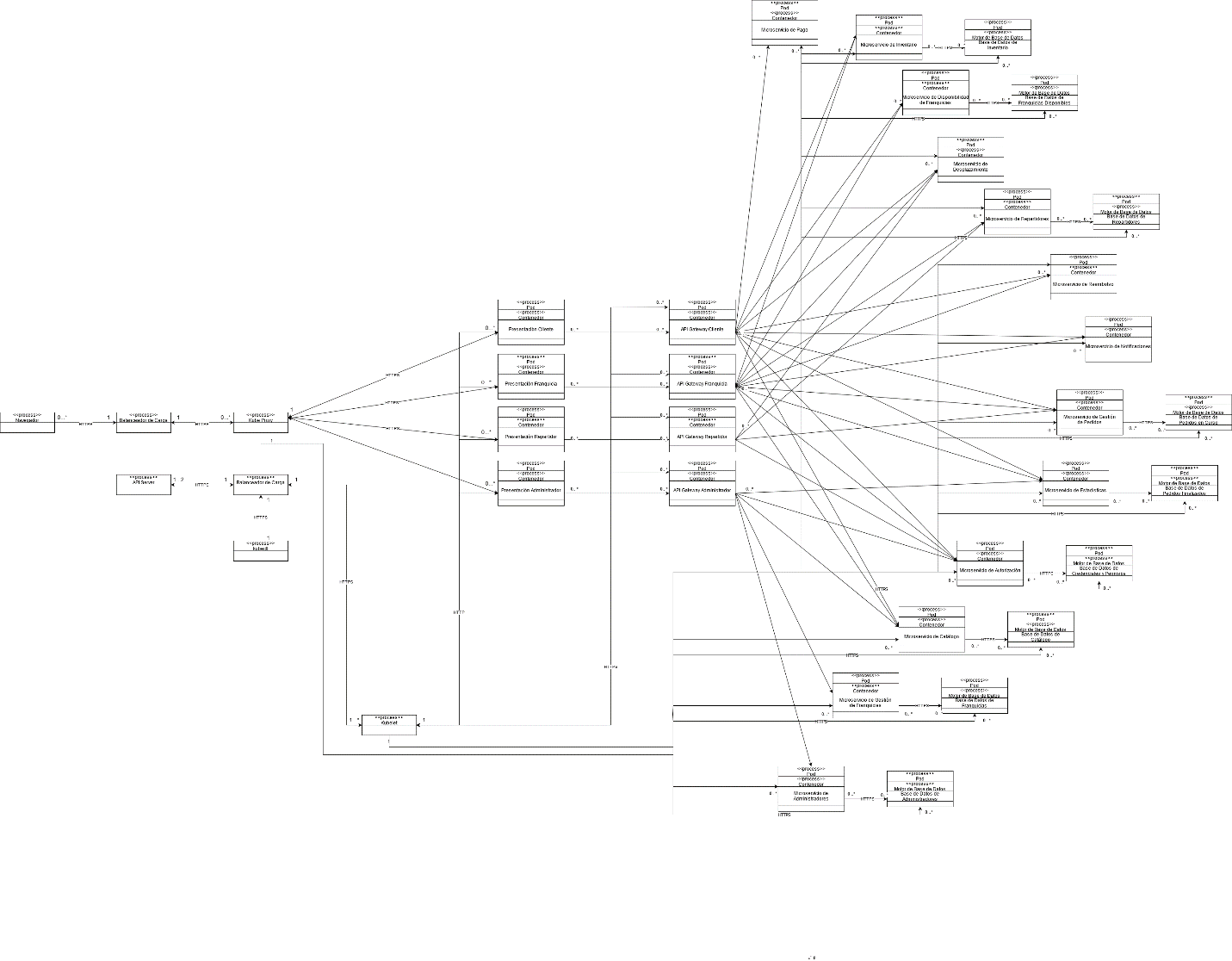
En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.

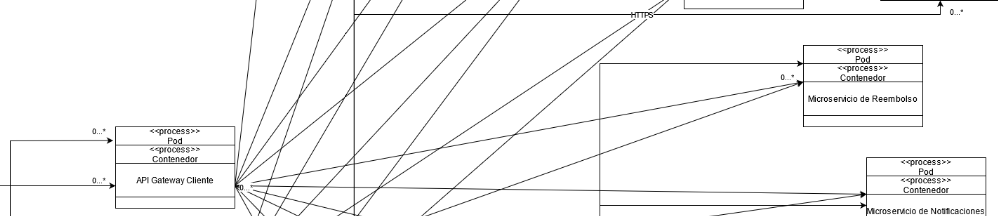




**Vista de Proceso**

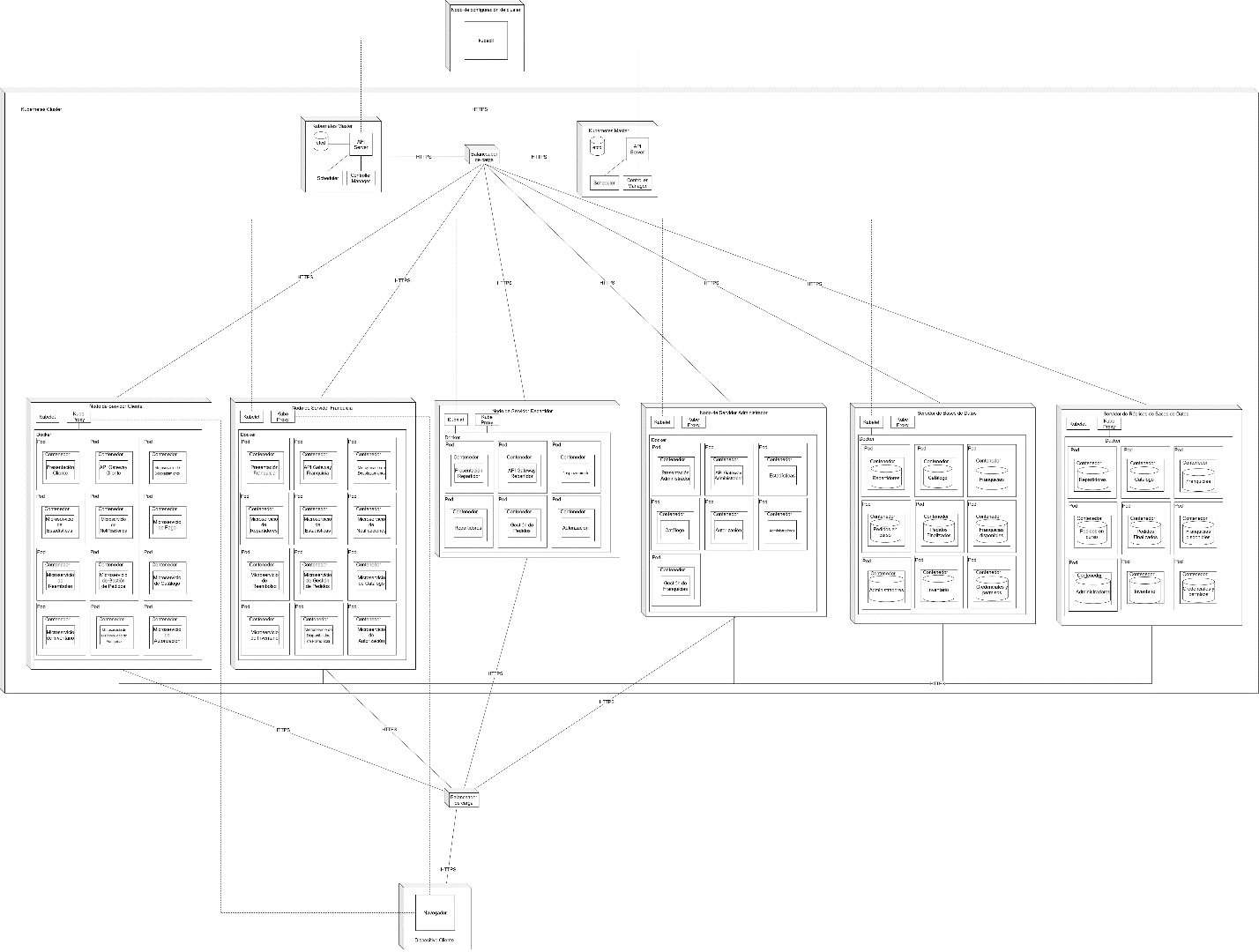
La división del microservicio de pago y de reembolso implicó agregar el proceso correspondiente al microservicio de reembolso en la vista de proceso.

****



**Vista Física**

La división del microservicio de pago y de reembolso implicó agregar los pods del microservicio de reembolso en el nodo de servidor cliente y en el nodo de servidor franquicia.

****

1. **Verificación de Requisitos**

Se consideran a los requisitos RF007, RF047 y RNF009 completados tras esta iteración.

**Iteración 12**

1. **Elemento escogido**

Microservicio de Gestión de Pedidos

1. **Requisitos arquitectónicamente significativos del elemento escogido**

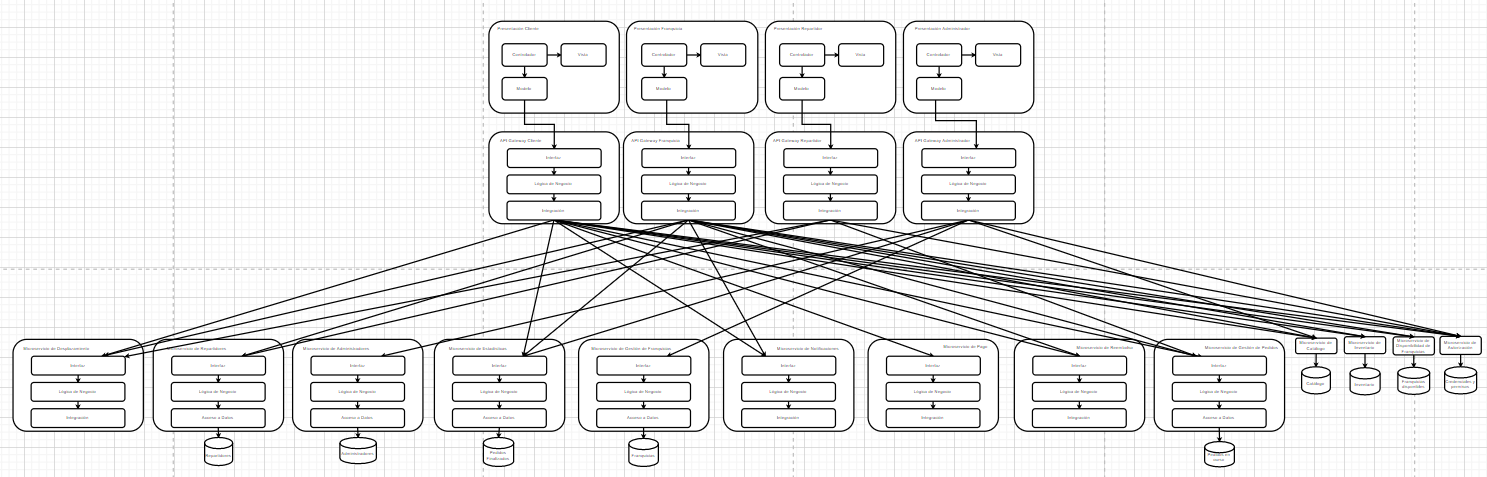
Este elemento se encuentra relacionado con RF001, RF002, RF003, RF005, RF008, RF009, RF011, RF030, RF031, RF034, RF045, RF046 y RF051.

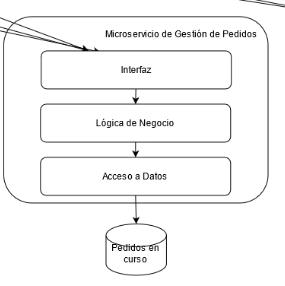
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar estos microservicios a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

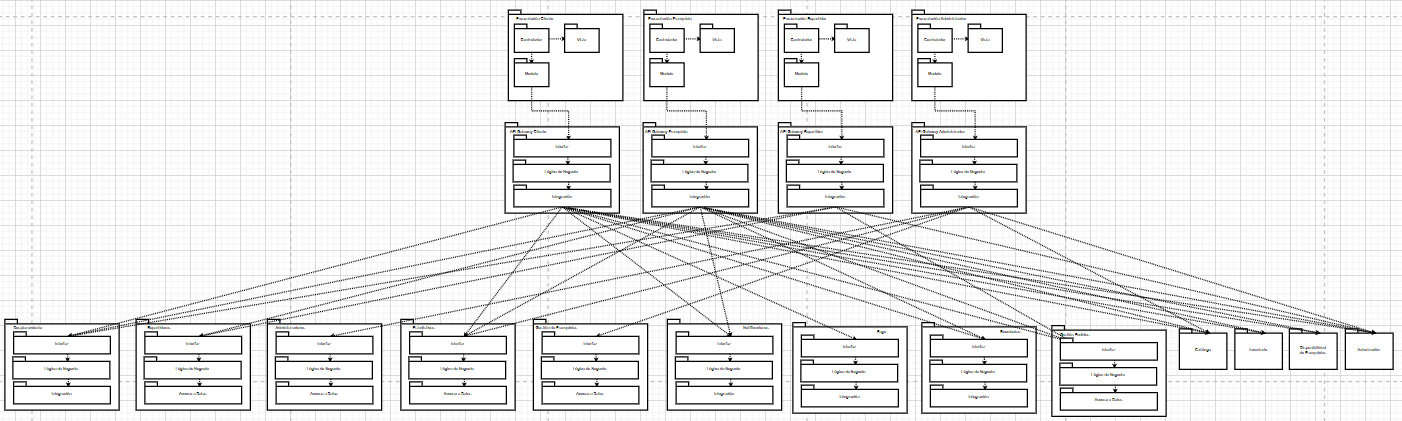
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, y Acceso a Datos, que se encargará del acceso a la base de datos de pedidos en curso.

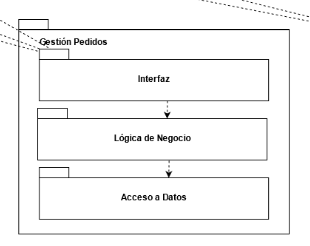




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.





**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Después de esta iteración, se consideran los requisitos RF005, RF008, RF009, RF011, RF030, RF031, RF046 y RF051 completados. Los demás requisitos deberán ser revisados en iteraciones posteriores.

**Iteración 13**

1. **Elemento escogido**

Microservicio de Catálogo

1. **Requisitos arquitectónicamente significativos del elemento escogido**

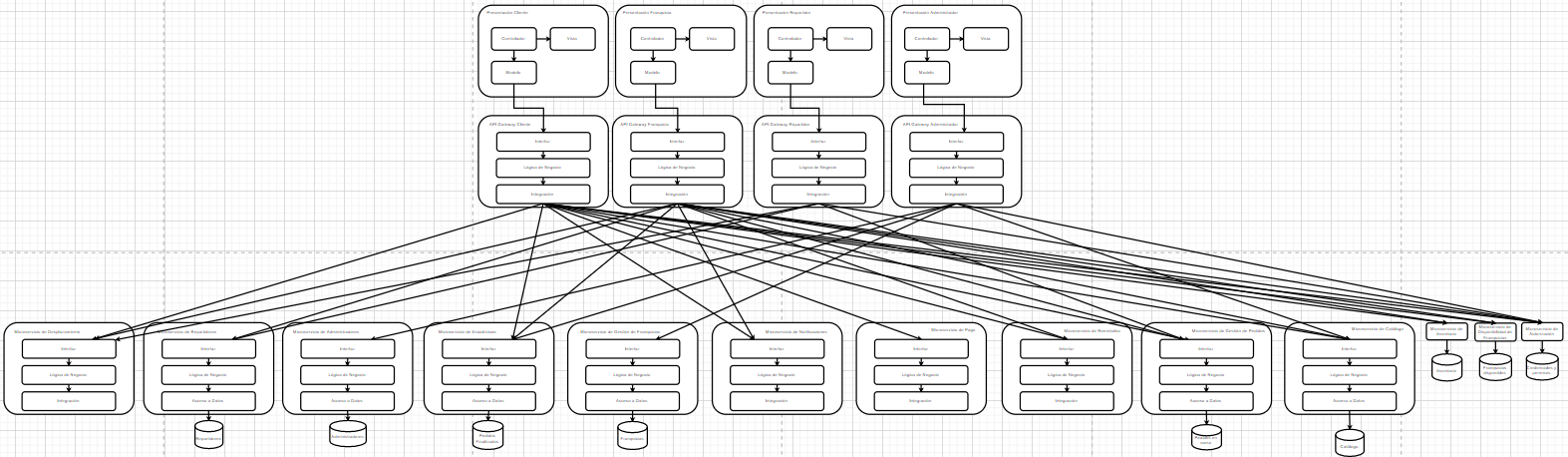
Este elemento se encuentra relacionado con el RF001, RF002, RF003, RF033, RF034, RNF012 y RNF013.

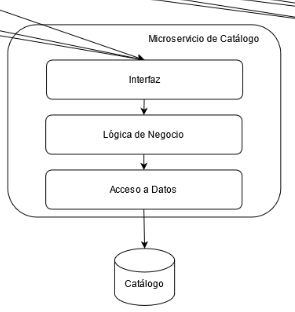
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar estos microservicios a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

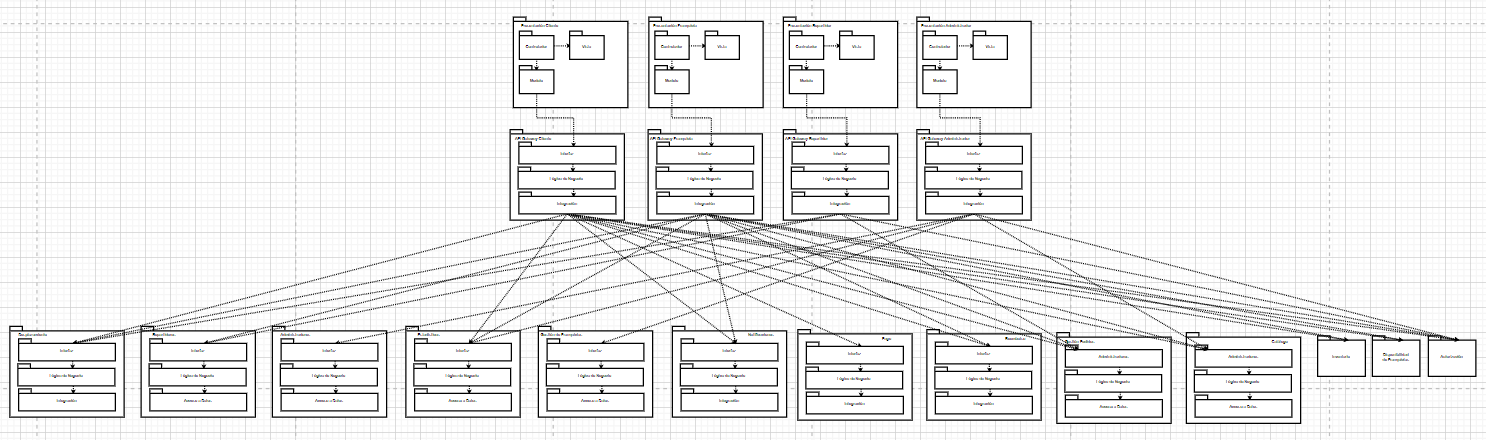
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, y Acceso a Datos, que se encargará del acceso a la base de datos de catálogo.

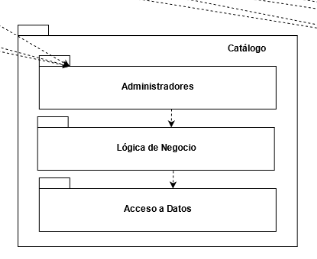




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.





**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Con esta iteración se considera el RF033 completo. Los demás requisitos deberán ser revisados en iteraciones posteriores.

**Iteración 14**

1. **Elemento escogido**

Microservicio de Inventario

1. **Requisitos arquitectónicamente significativos del elemento escogido**

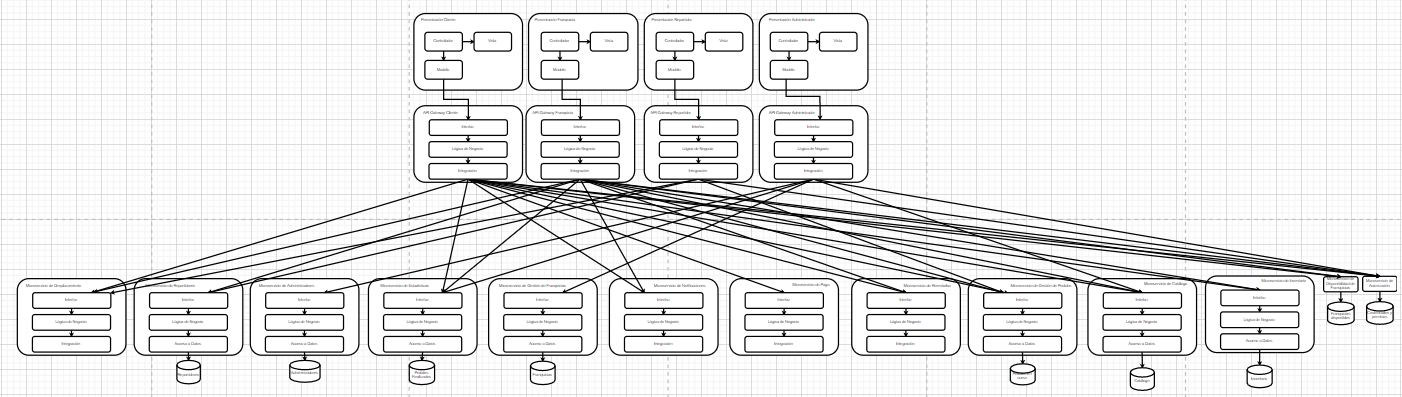
Este elemento se encuentra relacionado con el RF003, RF032, RF034, RNF012 y RNF013.

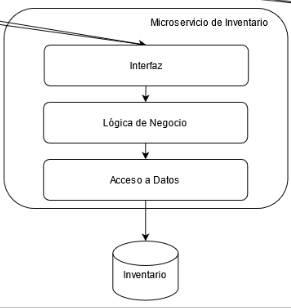
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar estos microservicios a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

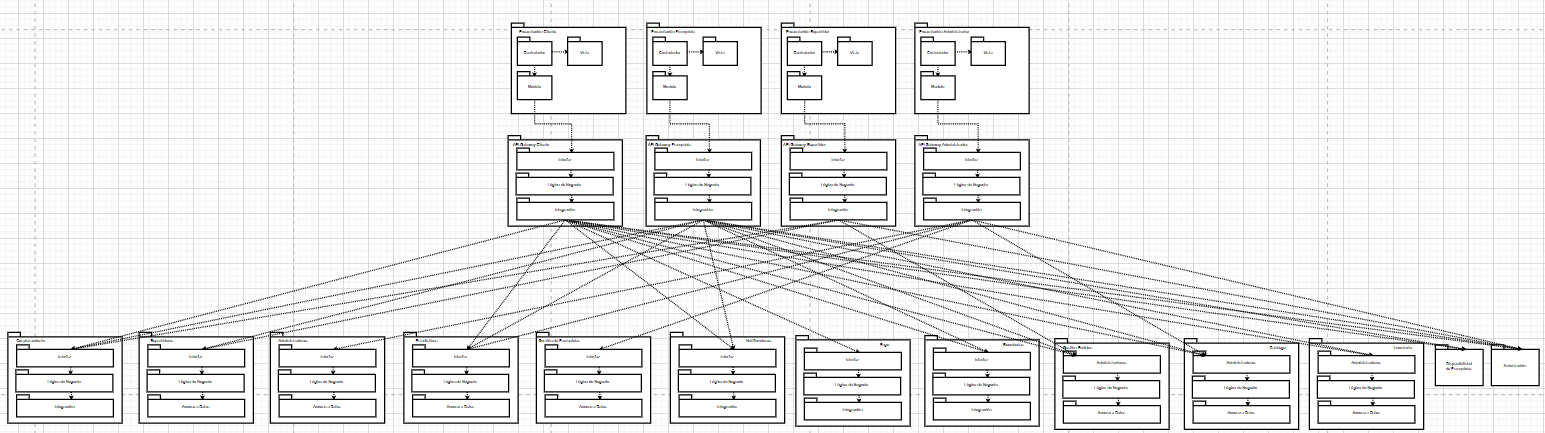
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, y Acceso a Datos, que se encargará del acceso a la base de datos de inventario.

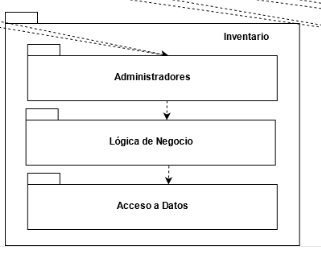




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.





**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Con esta iteración se considera el RF003, RF032 y RF034 completado.

**Iteración 15**

1. **Elemento escogido**

Microservicio de Autorización

1. **Requisitos arquitectónicamente significativos del elemento escogido**

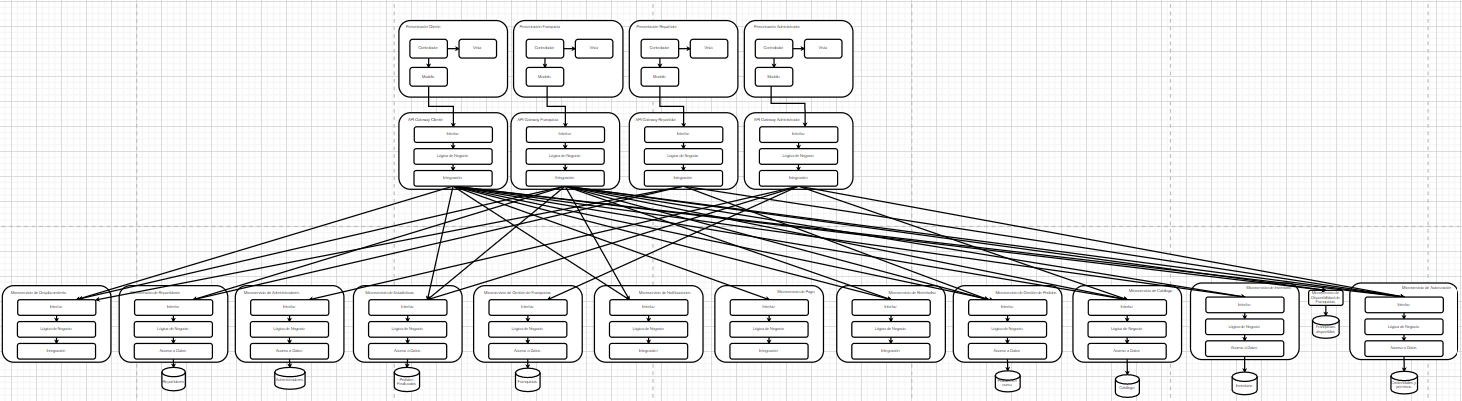
Este elemento se encuentra relacionado con el RF027, RNF012 y RNF013.

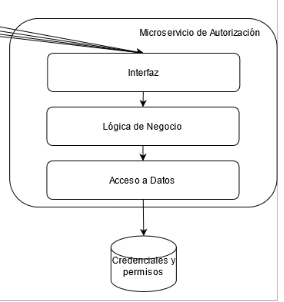
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar estos microservicios a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

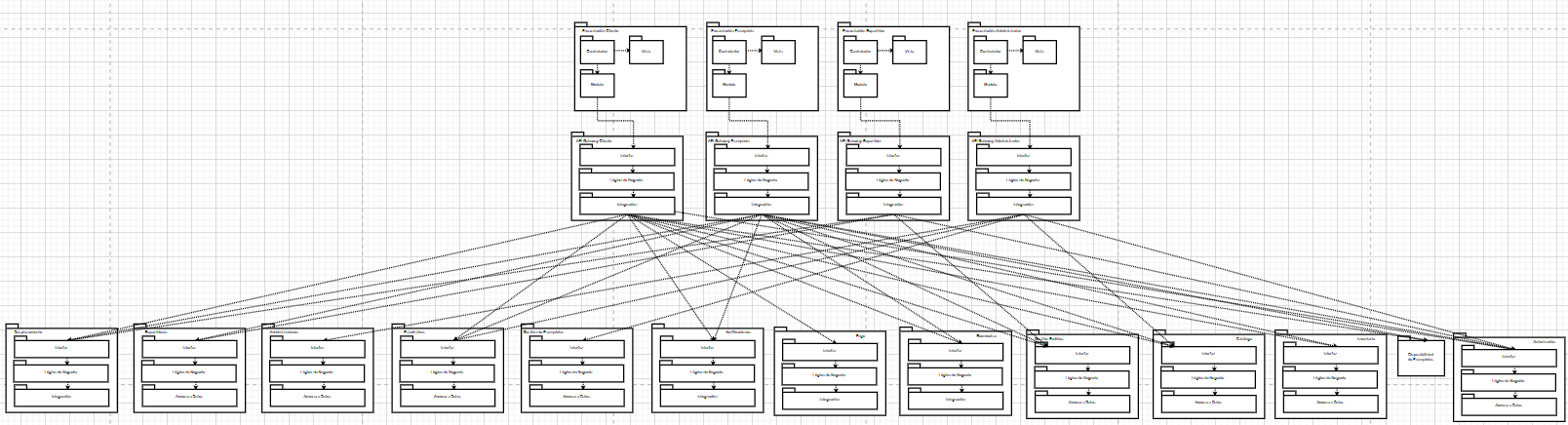
En este caso, las capas son: Interfaz, que recibe las solicitudes provenientes de los API Gateways; Lógica de Negocio, que incluye la lógica para realizar sus operaciones, y Acceso a Datos, que se encargará del acceso a la base de datos de credenciales y permisos.

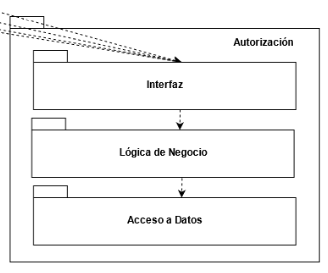




**Vista de Desarrollo**

En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.





**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.

1. **Verificación de Requisitos**

Con esta iteración se considera el RF027 completado.

**Iteración 16**

1. **Elemento escogido**

Microservicio de Disponibilidad de Franquicias

1. **Requisitos arquitectónicamente significativos del elemento escogido**

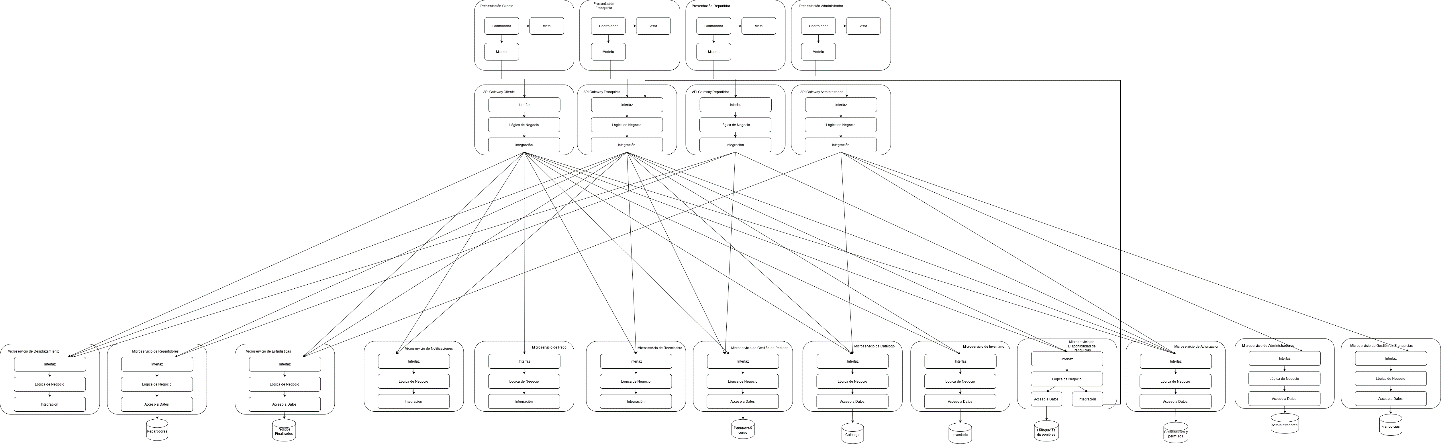
Este elemento se encuentra relacionado con RF001, RF002, RF030, RF031, RF045, RNF012 y RNF013.

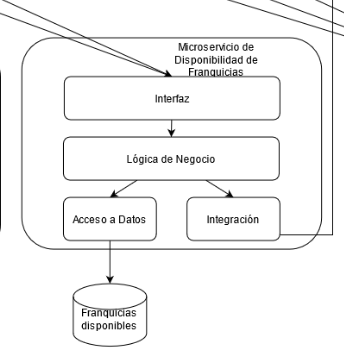
1. **Solución de diseño**

**Vista Lógica**

Se decidió estructurar estos microservicios a partir del patrón de capas, el cual permite dividir las responsabilidades de los componentes y restringir su comunicación a partir de interfaces bien definidas, de tal manera que cada capa solo consuma las interfaces de la capa inmediatamente inferior. Esto beneficia en gran medida la modificabilidad de estos componentes.

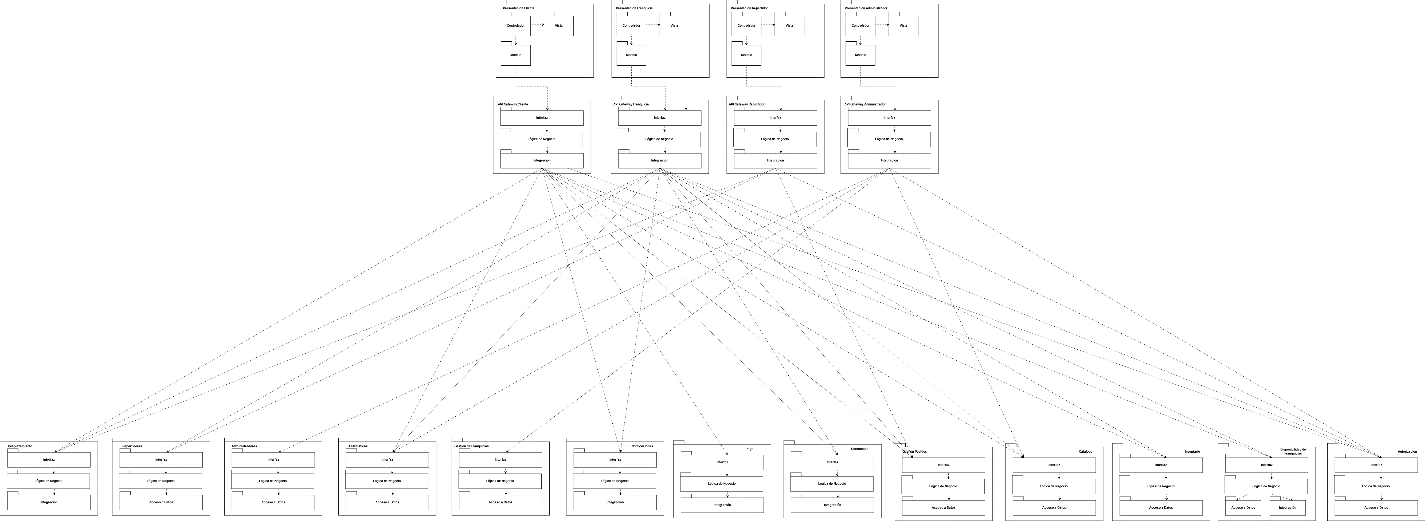
La capa de Interfaz recibe las solicitudes provenientes de los API Gateways mientras que la capa de Lógica de Negocio incluye la lógica para realizar sus operaciones. En este caso, la tercera capa se divide en dos componentes: Integración, que incluye la lógica para enviarle solicitudes al API Gateway de Franquicia, y Acceso a Datos, que se encargará del acceso a la base de datos de disponibilidad de franquicias.

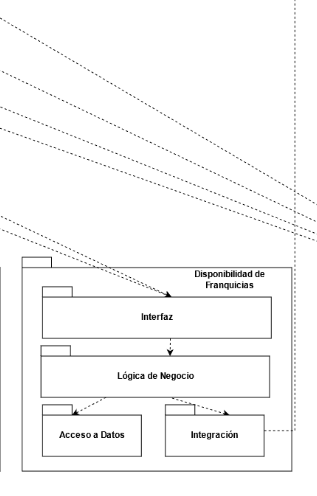
****



**Vista de Desarrollo**

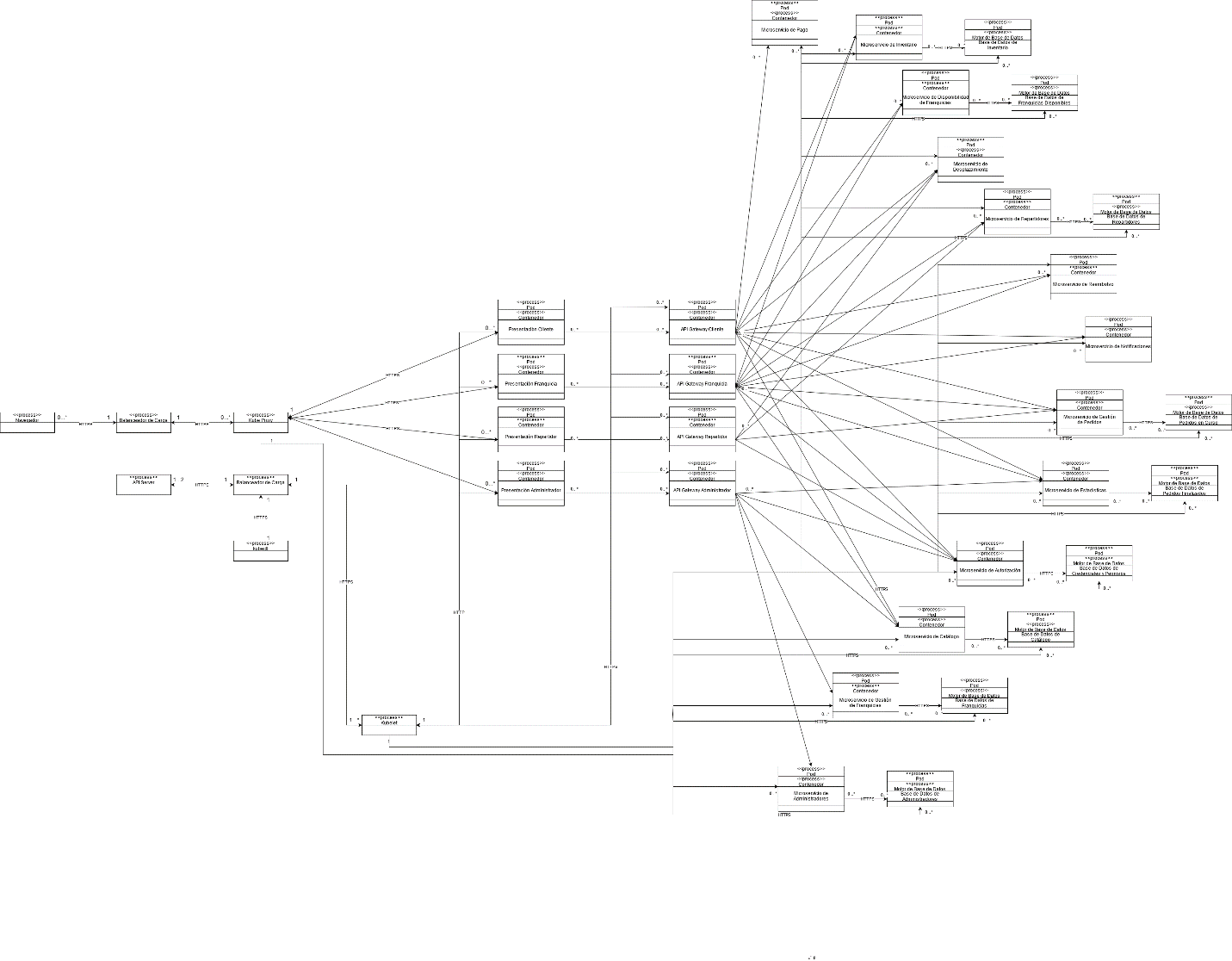
En la siguiente imagen, se puede observar el diagrama de paquetes del sistema, que es coherente con la división ilustrada en la vista lógica.

****



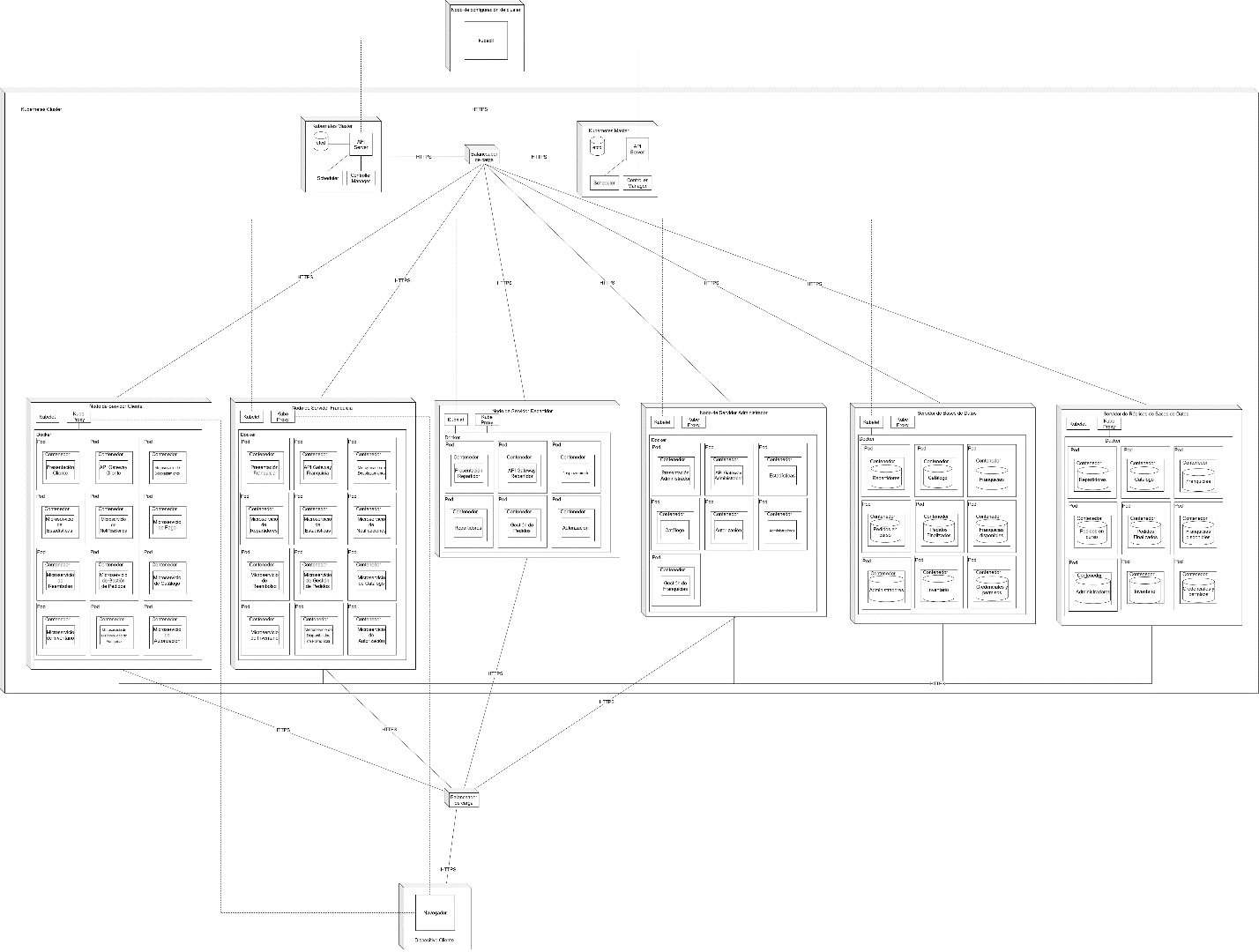
**Vista de Proceso**

Los refinamientos de esta iteración no implicaron cambios en la vista de proceso.

****

**Vista Física**

Los refinamientos de esta iteración no implicaron cambios en la vista física.



1. **Verificación de Requisitos**

Con esta iteración se consideran los RF001, RF002, RF030, RF031, RF045, RNF012 y RNF013 completados, permitiéndole al diseño satisfacer todos los requisitos arquitectónicos significativos.