Universidad ORT

Facultad de Ingeniería

Arquitectura de software

Obligatorio TePagoYa

Rodrigo Martínez - 154505

Ángel Díaz – 161761

Grupo: N7A

Docentes: Mathias Fonseca, Andres Calviño

Noviembre 2018

**Índice**

[1. Introducción 3](#_Toc391389969)

[1.1 Propósito 3](#_Toc391389970)

[2. Antecedentes 3](#_Toc391389971)

[2.1 Propósito del sistema 3](#_Toc391389972)

[2.2 Requerimientos significativos de Arquitectura 3](#_Toc391389973)

[2.2.1 Resumen de Requerimientos Funcionales 3](#_Toc391389974)

[2.2.2 Resumen de Requerimientos NoFuncionales 3](#_Toc391389975)

[3. Documentación de la arquitectura 4](#_Toc391389976)

[3.1 Vistas de Módulos 4](#_Toc391389977)

[3.1.1 Vista de Descomposición 4](#_Toc391389978)

[3.1.1.1 Representación primaria 4](#_Toc391389979)

[3.1.1.2 Catálogo de elementos (opcional) 5](#_Toc391389980)

[3.1.1.3 Decisiones de diseño (opcional) 5](#_Toc391389981)

[3.1.2 Vista de Uso 5](#_Toc391389982)

[3.1.2.1 Representación primaria 5](#_Toc391389983)

[] 5](#_Toc391389984)

[3.1.2.2 Catálogo de elementos (opcional) 5](#_Toc391389985)

[3.1.2.3 Decisiones de diseño 5](#_Toc391389986)

[3.1.3 Vista de Layers 6](#_Toc391389987)

[3.1.3.1 Representación primaria 6](#_Toc391389988)

[] 6](#_Toc391389989)

[3.1.3.2 Catálogo de elementos (opcional) 6](#_Toc391389990)

[3.1.3.3 Decisiones de diseño (opcional) 6](#_Toc391389991)

[3.1.3.4 Otras vistas de módulos 6](#_Toc391389992)

[3.1.4 Catálogo de elementos 6](#_Toc391389993)

[3.1.5 Interfaces 7](#_Toc391389994)

[3.1.6 Comportamiento 7](#_Toc391389995)

[3.2 Vistas de Componentes y conectores 8](#_Toc391389996)

[3.2.1 Vista de Tiers 8](#_Toc391389997)

[3.2.1.1 Representación primaria 8](#_Toc391389998)

[[Diagramas ] 8](#_Toc391389999)

[3.2.1.2 Catalogo de elementos 8](#_Toc391390000)

[3.2.1.3 Interfaces 8](#_Toc391390001)

[3.2.1.4 Comportamiento 8](#_Toc391390002)

[3.2.1.5 Relación con elementos lógicos 9](#_Toc391390003)

[3.2.1.6 Decisiones de diseño 9](#_Toc391390004)

[3.2.1.7 Otras vistas de C&C 9](#_Toc391390005)

[3.3 Vistas de Asignación 10](#_Toc391390006)

[3.3.1 Vista de Despliegue 10](#_Toc391390007)

[3.3.1.1 Representación primaria 10](#_Toc391390008)

[3.3.1.2 Catalogo de elementos 10](#_Toc391390009)

[3.3.1.3 Relación con componentes 11](#_Toc391390010)

[3.3.1.4 Decisiones de diseño 11](#_Toc391390011)

# Introducción

Este documento presenta información sobre la arquitectura y características del sistema TePagoYa.

El documento se estructura de la siguiente forma: una primera sección donde se identifican y especifican los requerimientos funcionales, no funcionales y restricciones; una segunda enfocada al detalle de la arquitectura, donde se utilizan distintas vistas para representar los aspectos más relevantes del sistema y reflejar las decisiones tomadas a nivel de diseño de la solución.

Es importante destacar que la siguiente lectura se elaboró para un público familiarizado con leguaje técnico referente a desarrollo de software.

Propósito

El propósito del documento es proveer una especificación completa de la arquitectura de TePagoYa, dejando una idea clara de las decisiones tomadas, sin entrar en detalla en los aspectos de desarrollo.

# Antecedentes

Propósito del sistema

El propósito del sistema es simplificar lo más posible la integración de los siguientes sistemas:

* Comercio.
* Gateway.
* Red.
* Emisor.

La propuesta es crear una herramienta (TePagoYa) que provea la funcionalidad de actuar de intermediario entre los cuatro diferentes actores resolviendo la mayor parte posible los desafíos de integración, como pueden ser los diferentes formatos de fechas y números, campos obligatorios y opcionales, formatos de representación de datos, etc.; permitiendo que la comunicación sea siempre hacia y desde el nuevo sistema, identificando el actor de destino de cada comunicación.

Requerimientos significativos de Arquitectura

Se describen a continuación los requerimientos: funcionales, no funcionales y restricciones que llevaron a la arquitectura desarrollada.

### Resumen de Requerimientos Funcionales

|  |  |  |
| --- | --- | --- |
| **ID Requerimiento** | **Descripción** | **Actor** |
| REQ01: Compra en Comercio | En función de la categoría del producto a comprar, el comercio deberá́ seleccionar el ​Gateway ​que procesará la compra y enviarle la solicitud. | Comercio |
| REQ02: Compra en Gateway | Cuando un ​comercio ​envía la información pertinente para que el ​Gateway procese un pago, este debe decidir a qué ​red enviar el pago según a cuál de ellas pertenece la tarjeta. | Gateway |
| REQ03.1: Compra en Red | Se debe identificar el emisor de la tarjeta. | Red |
| REQ03.2: Compra en Red | Para el análisis de fraude, se revisa la cantidad de transacciones que hizo esa tarjeta en las últimas 72 horas y si supera un límite determinado la transacción se rechaza. | Red |
| REQ04.1: Compra en Emisor | Cuando la ​red envía una transacción a ser aprobada en el ​emisor,​ este debe confirmar que la tarjeta sea válida. | Emisor |
| REQ04.2: Compra en Emisor | Cuando la ​red envía una transacción a ser aprobada en el ​emisor debe validar que el saldo de la cuenta del titular sea suficiente para el monto a aprobar. | Emisor |
| REQ04.3: Días de devolución | El emisor podrá definir la cantidad máxima de días que pueden transcurrir entre que se realiza una compra y la devolución de la misma. | Emisor |
| REQ5: Devolución | Deberá́ ser posible realizar la devolución de cualquier compra realizada previamente que no haya superado el límite de días definido en el ​emisor. Realizando todos los controles pertinentes, notificando y asegurando la consistencia de todas las aplicaciones involucradas. | Comercio/ Gateway /Red/ Emisor |
| REQ6: Chargeback | El usuario puede realizar un desconocimiento de compra en su ​emisor.​ El emisor deberá informar al comercio de este evento para que ejecute las acciones administrativas pertinentes. | Emisor |
| REQ7: Cierre de Lotes | El ​comercio ​le solicita diariamente al ​Gateway la transacción de cierre de lotes, en donde se informa al máximo detalle los movimientos de dinero que hubo en su cuenta en ese día. | Gateway |
| REQ8: Registro de Interfaz | Para cada prestador de servicios podrá registrar una “interfaz de servicio” que permita acceder a su funcionalidad. | TePagoYA |
| REQ9: Especificar formatos | Los prestadores de servicios podrán especificar propiedades sobre la forma de usar cada interfaz (por ejemplo, el formato de las estructuras de datos que se intercambian por la interfaz, o el tipo de comunicación). | TePagoYA |
| REQ10: Localización de Interfaces | Para cualquier consumidor de servicios de debe poder localizar una interfaz de un servicio registrado. | TePagoYA |
| REQ5: Validez de tarjeta de crédito. | Para que una tarjeta sea válida, debe haber sido emitida por el emisor, pasar el algoritmo de Luhn, no estar vencida, ni bloqueada (por falta de pago), ni denunciada (por robo o pérdida). | Emisor |
| REQF6: ChargeBack | Se debe poder hacer un Chargeback de hasta 6 meses atrás | Emisor |
| REQF6.2: Hora de cierre de lote | El sistema permitir cerrar los lotes a una hora acordad | Gateway |

Tabla 2.1 – Requerimientos funcionales

### Resumen de Requerimientos No Funcionales (Ver si hay que agregar un requermiento no funcional sobre el repositorio, importante que atributo de caldiad )

|  |  |  |  |
| --- | --- | --- | --- |
| **ID Atributo** | **Descripción** | **Atributo de calidad** | **Actor** |
| REQNF01: Selección de Gateway | La selección del Gateway debe ser configurable sin necesidad de modificar el código de los comercios. | Modificabilidad | Comercio |
| REQNF02: Tiempos de identificación de red | La identificación de la red debe realizarse en tiempos menores a 10 ms en promedio bajo cargas de 1000 solicitudes por minuto. | Eficiencia | Gateway |
| REQNF03: Configuración de límites de compras | Este límite de operaciones debe poder ser alterado en tiempo de ejecución sin necesidad de interrumpir la operación de la red. | Modificabildad | Red |
| REQNF04: Tiempos de identificación de emisor | La identificación del emisor debe realizarse en tiempos menores a 10 ms en promedio bajo cargas de 1000 solicitudes por minuto. | Eficiencia | Red |
| REQNF05: Tiempo de procesamiento del cierre de lotes. | Esta transacción debe responder en un tiempo promedio menor a 50 ms bajo cargas de 5000 rpm. | Eficiencia | Gateway |
| REQNF06: Cumplimiento de PCI | Toda aplicación debe cumplir con los estándares de seguridad de PCI, pudiendo demostrar y justificar a cada momento el nivel de cumplimiento. | Seguridad | N/A |
| REQNF07: Mecanismo de integración de aplicaciones | Desarrollar la aplicación ​TePagoYa para resolver la integración e interoperabilidad de las aplicaciones actuales y futuras. Toda aplicación que se registra en TePagoYa puede actuar como prestador de servicios y/o consumidor de servicios. | Modificabildad | N/A |
| REQNF08: Gestión de errores y fallas | El sistema debe proveer suficiente información, de alguna forma, que permita conocer el detalle de las tareas que se realizan. En particular, en el caso de ocurrir una falla o cualquier tipo de error. |  | N/A |
| REQNF-11: Autenticación de operaciones hacia prestadoras de servicios | Para toda aplicación “prestadora de servicios” que se registre en ​TePagoYa,​ se requiere que todas sus operaciones publicadas deben ser invocadas únicamente a través de ​TePagoYa.​ | Seguridad | N/A |
| REQNF-12: Autenticación de operaciones hacia TePagoYA | Toda invocación a ​TePagoYa debe provenir de una aplicación consumidora de servicios legítima. | Seguridad | N/A |

Tabla 2.2 – Requerimientos no funcionales

### Resumen de Restricciones

|  |  |
| --- | --- |
| **ID Restricción** | **Descripción** |
| RES1: Backend | La implementación del Backend debe desarrollarse en Node js utilizando las tecnologías vistas en el curso. |
| RST2: APIs | Las APIs expuestas por los servicios de comercios deben ser REST. |
| RST3: GIT | Todo el código fuente, documentación, archivos de configuración o cualquier otro artefacto referido al desarrollo de los prototipos debe gestionarse en el repositorio GIT asignado. |

Tabla 2.2 – Restricciones

# Documentación de la arquitectura

En las secciones de este capítulo se documenta la arquitectura del sistema por medio de distintas vistas (Módulos, Componentes y Conectores, Asignación).

* **Módulos**: se detallan los módulos identificados en las distintas aplicaciones desarrolladas, los cuales fueron agrupados de acuerdo a sus responsabilidades. Para dicha representación se utilizaron dos vistas:
  + **Vista de descomposición:** muestra las agrupaciones de acuerdo a responsabilidades utilizando la relación “*is-part-of*”.
  + **Vista de uso:** muestra las relaciones existentes entre los distintos módulos utilizando la relación “*depends-on*”.
* **Componentes y Conectores**: se detallan los elementos que tienen presencia en tiempo de ejecución, tales como: procesos, servidores, BD, etc, y las interacciones entre ellos representadas como conectores. Para dicha representación se utilizó:
  + **De cliente-servidor:** para representar los distintos servicios consumidos y prestados entre los distintos componentes.
* **Asignación**: se detalla el mapeo entre los elementos de software y de entorno, para dicha representación se utilizó:
  + **De Despliegue:** describe el mapeo entre componente-conectores y el hardware.

Vistas de Módulos

Vista de Descomposición - Representación primaria (Alinear titulos de applicaciones)

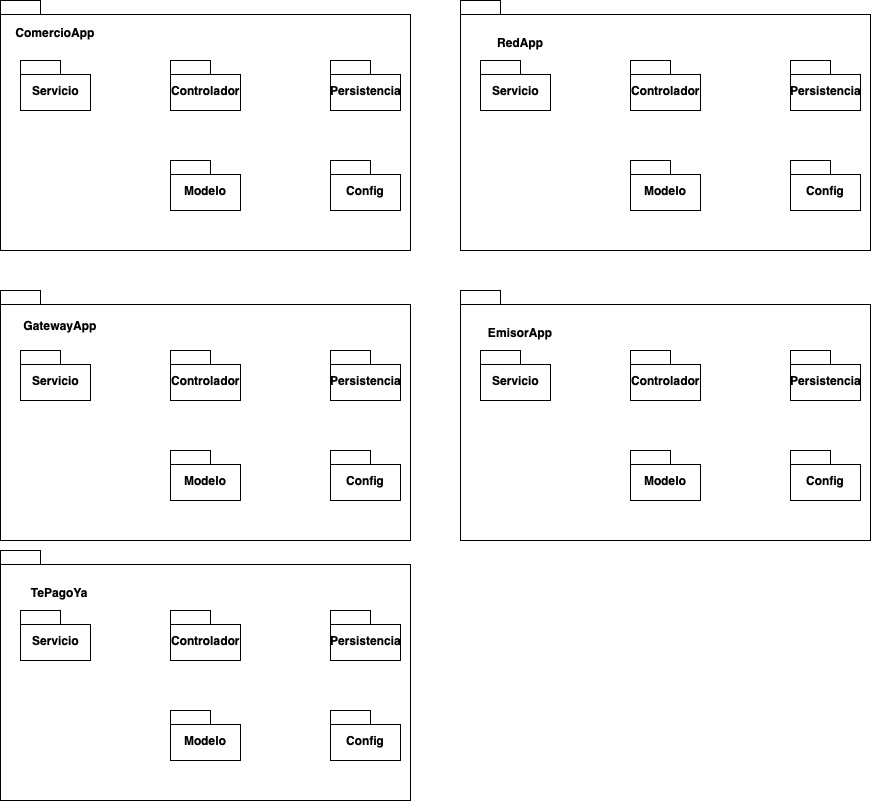


Ilustración 3.1 – Diagrama de Módulos.

### Catálogo de elementos

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| ComercioApp | Representa la aplicación con la lógica de negocio del comercio. Este ofrece las funcionalidades especificadas en la tabla de requerimientos bajo el actor Comercio. |
| GateWayApp | Agrupa todo lo relacionado con la entidad Gateway. |
| RedApp | Contiene todo lo relacionado con la entidad Red. |
| EmisorApp | Agrupa todo lo relacionado con la entidad Emisor. |
| TePagoYA | Agrupa todo lo relacionado con la entidad TePagoYA. |
| Servicio | Agrupa toda la logica encargada de las peticiones. |
| Controlador | Agrupa toda la logica de negocio propiamente dicha |
| Persitencia | Encapsula la logica encargada de persistir los datos |

Tabla 3.1 – catálogo de elementos

### Decisiones de diseño

En el diagrama de representación primaria se puede ver la división en módulos de cada uno de los aplicativos, la cual se realizó bajo los mismos criterios para todos.

Para su agrupación se pensó en posibles cambios, analizando los costos y tiempos que un cambio podría tener, para esto nos apoyamos en las siguientes preguntas:

* ¿Qué puede cambiar?
* ¿Cuál es la probabilidad de un cambio?
* ¿Cuál es el costo del cambio?

Se buscó minimizar los costos de mantenimiento, por lo que fue necesario reducir el acoplamiento entre módulos de los distintos aplicativos, así como también aumentar la cohesión de los mismos.

Por otro lado, se consideró importante escribir código limpio, de forma que sea fácil la lectura de cada pieza de código, sobre todo para cualquier persona ajena al autor de la misma. Para alcanzar dicho objetivo, se utilizó las convenciones de Node js, donde se prestó suma atención a:

* nombres de métodos.
* nombre de variables.
* nombre de módulos.
* evitar la necesidad de comentarios.

De alguna forma, se consideró pertinente lograr que a simple vista se pueda tener una idea de las responsabilidades asociados a cada pieza de código.

A modo de ejemplo, a la hora de pensar en persistencia, agrupamos las funciones cuyas responsabilidades estaban fuertemente relacionadas a la misma, buscando aumentar la coherencia semántica y reducir el impacto que puede llegar a tener un cambio, evitando que este se propague a módulos que poco tienen que ver con la persistencia. Además, al tener las funcionalidades agrupadas por responsabilidad, se facilita la búsqueda de las piezas relacionadas al cambio.

Otro ejemplo referente a la persistencia fue separar los esquemas - modelo de datos - en distintos módulos pensando en el tamaño de los mismos, teniendo un módulo por cada esquema a persistir.

### Vista de Uso

Tener claridad en las dependencias existentes entre los distintos módulos, facilita el entendimiento del impacto de cualquier cambio solicitado, pudiendo identificar qué partes del software pueden verse afectadas por el mismo.

### Representación primaria

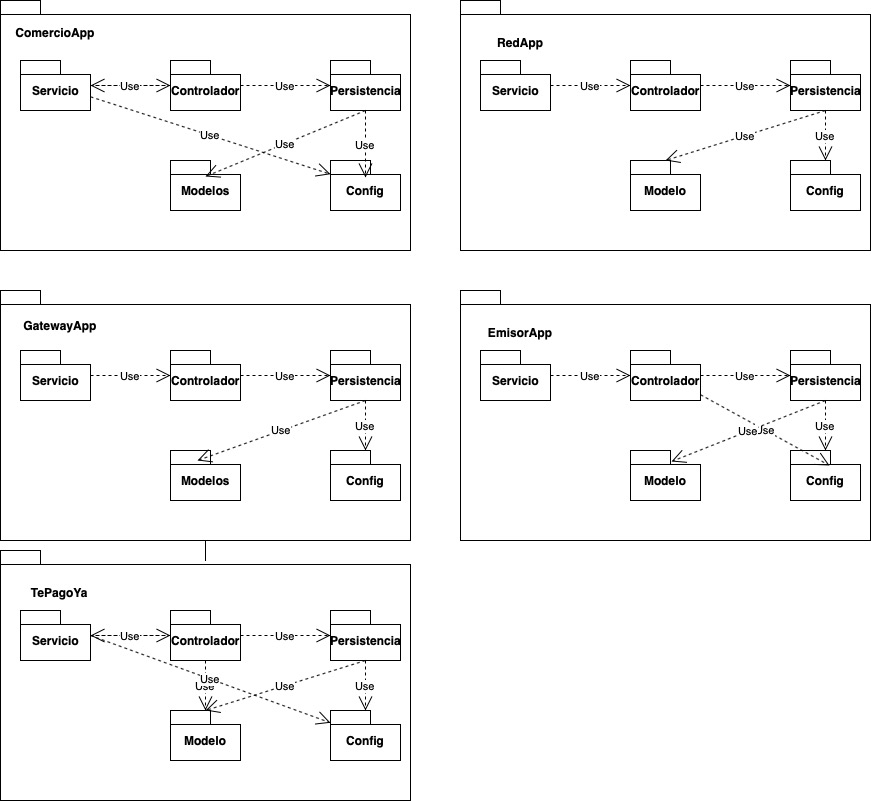


Ilustración 3.2 – Diagrama de Módulos y sus relaciones de uso.

### Decisiones de diseño

Agregar explicación de para que sive vista de usa

### Componente y conectores

En esta sección se describe las vistas de componentes y conectores que el arquitecto consideró relevantes para comunicar la visión del sistema en tiempo de ejecución. En particular se describen los componentes, las formas de conexión y la interacción entre los componentes para explicar la implementación de funcionalidades o de mecanismos claves.

### Representación primaria

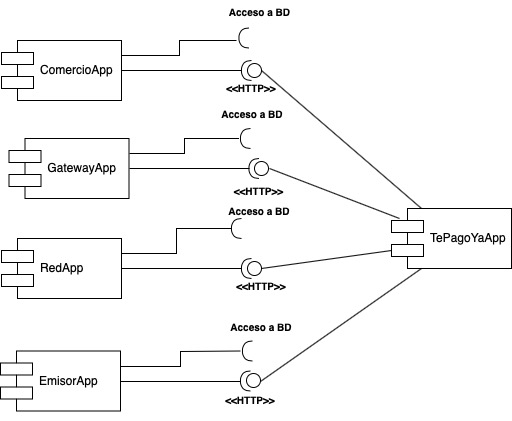


Ilustración 3.3 – Diagrama de componentes y conectores

### Decisiones de diseño

Como se puede observar en la Ilustración 3.3, se buscó desacoplar los componentes por medio de interfaces API REST por medio de comunicaciones HTTP, de esta forma se minimiza las dependencias en la comunicación de los aplicativos. Ayudando a futuro a bajar costos de redespligue

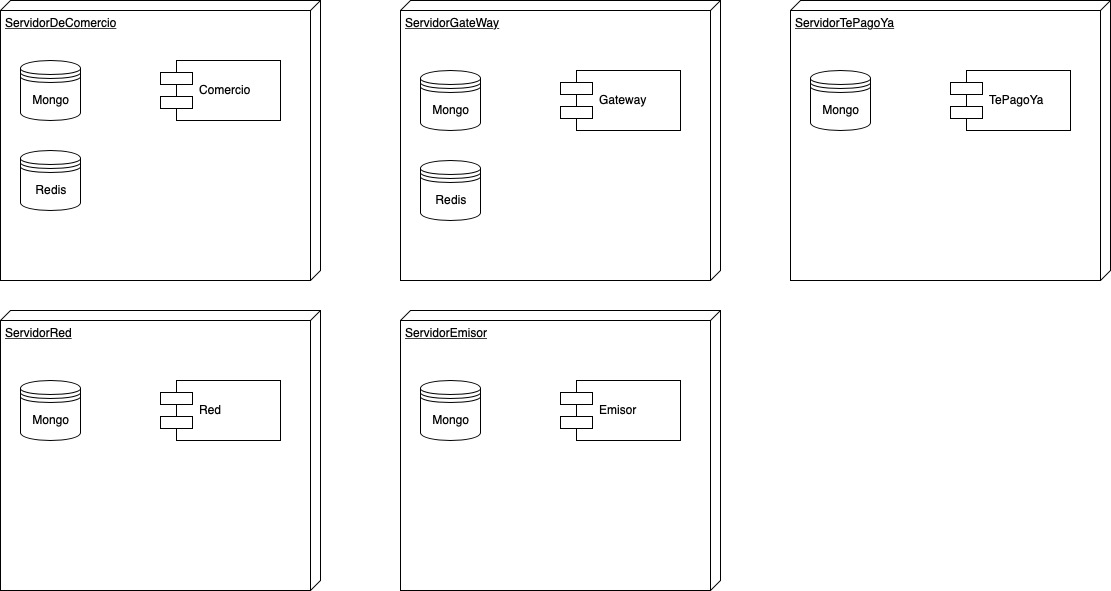
El formato de datos utilizado mayormente para el intercambio de datos, fue JSON, donde cada API tiene la responsabilidad de transformar los parámetros a una representación interna de datos.

### Vista de Asignación

A continuación desarrollamos una vista de asignación mostrando la forma óptima de despliegue del sistema aprovechando las ventajas de la arquitectura.

Se decidio desplegar al menos 5 servidores diferentes.

### Representación Primaria (Ver esquema de redis y cambiar nombre de las app)



### Decisiones de diseño

Como forma de representar los aplicativos en un escenario lo más acercado a la realidad, se divide su despliegue en servidores físicos ya que cada uno representa una empresa distinta. Permitiendo una mejor capacidad de mantenimiento y re-despliegue.

Se utilizó una Base de datos no relacional para cada sistema tratando de favorecer la eficiencia en cada subsistema.

### Principales decisiones de diseño

REQNF01: Selección de Gateway.

REQNF03: Configuración de límites de compras.

Modificabilidad

Con el objetivo de promover el atributo de calidad asociado a los requerimientos no funcionales REQNF01 y REQNF03 se utilizó la táctica de diferir el tiempo de enlazado.

Por un lado, en reglas generales de la solución, se parametrizaron varios aspectos de configuración (DB, aplicación, autenticación, etc.), para esto se creó un paquete con módulos referentes a la configuración, promoviendo la modificabilidad y aumentando la flexibilidad en cientos aspectos de los aplicativos.

En lo que refiere a los atributos de calidad mencionados previamente, se implementó la táctica por medio de MongoDB.

Para cumplir con el REQNF01, el comercio cuenta con una tabla que tiene la relación Gateway - Categoría, por tanto si se requiere modificar la categoría asociada a un Gateway, simplemente se modifica el registro de la tabla para la base de datos del comercio. De esta forma, se puede modificar fácilmente la selección del Gateway sin tocar código existente, y por otro lado permitiendo su configuración en tiempo de ejecución.

Para cumplir con el REQNF03, se creó una tabla referente a configuraciones en la base de datos el emisor, por tanto razonando análogamente al punto anterior, se provee flexibilidad para la modificación del límite de compras y su configuración puede ser en tiempo de ejecución.

Es importante destacar que se eligió como manejador MongoDB por su flexibilidad para manejar los datos y la librería Mongoose ya que simplificó todas las operaciones relacionadas a su manipulación (creación de esquemas, salvado, funciones de agregación, etc).

Para los esquemas definidos, se realizan controles previó al salvado: nombre de campos, tipo de datos, campos requeridos, etc.; pero para este primer Release, por temas de tiempo y alcance, el sistema no contempla un control frente al salvado de un esquema que contemple más campos de los definidos dentro de cada esquema, mostrando una vulnerabilidad a nivel de base de datos.

REQNF02: Tiempos de identificación de red

REQNF04: Tiempos de identificación de emisor

REQNF05: Tiempo de procesamiento del cierre de lotes.

Eficiencia

Con el objetivo de promover el atributo de calidad asociado a los requerimientos no funcionales REQNF02 y REQNF04, se utilizó la táctica de mantener múltiples copias de datos

Para esto fue necesario replicar los datos en repositorios distintos con velocidades de acceso distintas: cache y base de datos en Mongo.

De esta forma se optimizan los tiempos de respuesta incrementando la eficiencia, ya que se accede a memoria cache en lugar de utilizar la memoria secundaria de la base de datos.

Se muestra a continuación las pruebas realizadas que reflejan la mejora en términos de eficiencia al implementar la táctica mencionada.

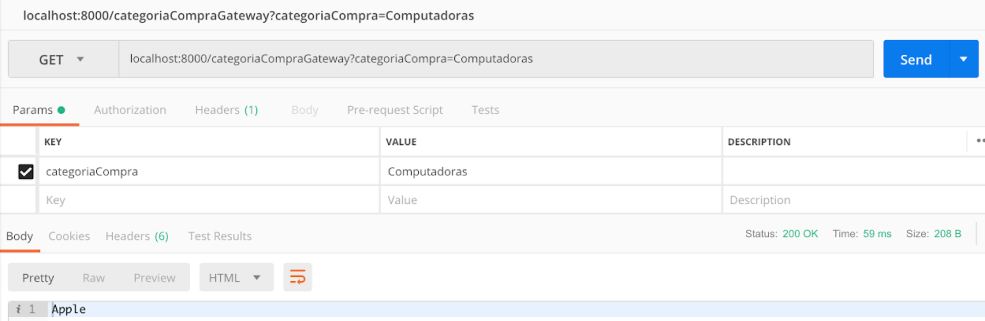


Ilustración 3.x – Prueba consulta Gateway en base de datos.

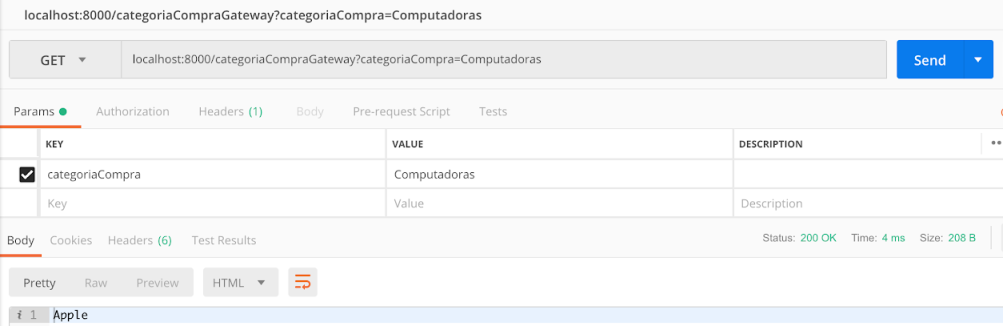


Ilustración 3.x – Prueba consulta Gateway en cache.

Como se puede observar, se mejora 14 veces los tiempos de procesamiento.

REQNF-8: Mecanismo de integración de aplicaciones

Modificabildad: con el fin de mejorar este atributo utilizamos varias tácticas:

Incrementar la cohesión: Generamos módulos en función de responsabilidades. Permitiendo dejar encapsuladas las probabilidades del cambio, haciendo que los cambios no se propaguen a otros módulos. Ejemplo: todo lo relacionado con comunicarse con la BD de datos quedo encapsulado en el módulo de Persistencia.

Abstracción de servicios comunes: Dado que todas las aplicaciones consumirán servicios de una forma similar de TePagoYa, vimos la oportunidad brindar un servicios mas general que varios especializados para cada tipo de consumidor, ayudando que cualquier modificación a realizar sobre el servicio se haga en un solo lugar reduciendo los costos.

**REQNF-9: Gestión de errores y fallas**

Hablar de que se usan try cach para capturar los errores

Para log vamos a guardar en la base de datos el detalle del error con la fecha

Autenticacion de login cuando se carga la app se le hace un req al modulo de autencticacion y se devuelve un token