Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Arquitectura de software

Obligatorio

Integrantes:

Sebastián del Arca 202096

Docentes:

Gerardo Quintana
Marco Fiorito
Juan Possayman

DESCRIPCIÓN DE ARQUITECTURA DE PhiTV

GitHub

Grupo M7B - Junio 2023

ÍNDICE

1. INTRODUCCIÓN **3**

2. ANTECEDENTES **3**

2.2.1 Resumen de Requerimientos Funcionales *4*

2.2.2 Resumen de Requerimientos de Atributos de Calidad *5*

3. DOCUMENTACIÓN DE LA ARQUITECTURA **6**

3.1.1 Vista de Descomposición *7*

3.1.1.1 Representación primaria 7

3.1.2 Vista de Uso *7*

3.1.2.1 Representación primaria 7

3.1.2.2 Decisiones de diseño 8

3.1.3 Vista de Layers *8*

3.1.3.1 Representación primaria 8

3.1.4 Catálogo de elementos *9*

3.1.5 Comportamiento *10*

3.1.6 Decisiones de diseño *11*

3.2.1 Representación primaria 12

3.2.2 Relación con elementos lógicos 12

3.2.3 Decisiones de diseño 12

3.3.1 Vista de Despliegue *14*

3.3.1.1 Representación primaria 14

3.3.1.2 Catálogo de elementos 14

3.3.2 Vista de Instalación *16*

3.3.2.1 Representación primaria 16

3.3.2.2 Catálogo de elementos 16

3.3.2.3 Decisiones de diseño 17

4. INSTRUCCIONES PARA CORRER LA APLICACIÓN **18**

5. ANEXO **19**

Módulos NPM *20*

Archivos .env *20*

Pasarela de Pagos 20

Unidad Reguladora 20

phiTV 21

1. Introducción

El presente documento intenta brindar una descripción del sistema phiTV. Para esto se intenta describir cada elemento según las diferentes vistas arquitectónicas. El documento presenta sobre cada visión general mostrando diagramas y luego una descripción sobre las decisiones de diseño en cada apartado.

1.1 Propósito

El propósito del presente documento es proveer una especificación completa de la arquitectura propuesta para el sistema phiTV. Para la descripción de la misma haremos uso de múltiples vistas arquitectónicas, así como diagramas que representen de la mejor forma posible las principales decisiones de diseño que buscan cumplir los atributos de calidad planteados.

2. Antecedentes

2.1 Propósito del sistema

La empresa ÁureaUY necesita construir una plataforma de streaming de eventos de video bajo demanda llamada phiTV. La plataforma permitirá a distribuidores de contenido, como por ejemplo a empresas de TV cable y a streamers, poder publicar sus propios eventos controlando las suscripciones, los pagos por los eventos y los usuarios que se suscriben.
Los objetivos de la plataforma son los siguientes:
• Permitir a las empresas proveedoras de contenido publicar en pocos minutos eventos con su marca propia (white-label product: producto de marca blanca). Además, se quiere que puedan controlar las suscripciones de los usuarios, los precios que le otorguen a los eventos y a los pagos que realicen los usuarios. De esta forma, las empresas no tienen que invertir esfuerzo en configurar y mantener la infraestructura necesaria para el streaming de video de eventos.
• Permitir soportar las altas demandas provocadas por el masivo ingreso de usuarios que consumen los eventos que proveen las empresas.
• Brindar un marco de interacción confiable entre los interesados mediante el intercambio de información seguro.

Los principales usuarios y entidades que utilizarán la solución son:
• Proveedores: empresas que se dedican a la generación de contenido, que utilizarán la plataforma para gestionar las transmisiones y ofrecer el contenido mediante suscripciones.
• Clientes: usuarios que compran los productos que ofrecen los proveedores mediante suscripción y acceden a las transmisiones de los eventos que compraron.
• Administradores: son los técnicos de la empresa que desarrolló la plataforma encargados de gestionar el acceso de los proveedores.

La plataforma interactúa con los siguientes sistemas:
• Pasarelas de Pago: servicio de terceros que permite realizar los pagos de los Proveedores a la empresa por la creación de un canal y permite a los Clientes que se suscriben a los eventos de los Proveedores pagar por ellos.
• Unidad Reguladora: Organismo encargado de autorizar a los proveedores a transmitir eventos. Expone un servicio de consulta que brinda información sobre la autorización de los proveedores para transmitir eventos.

2.2 Requerimientos significativos de Arquitectura

En esta sección se describen los requerimientos funcionales, atributos de calidad y restricciones del sistema.

2.2.1 Resumen de Requerimientos Funcionales

ID Requerimiento	Descripción	Actor
RF 1	*Se debe proveer la capacidad para dar de alta un proveedor para que pueda operar en la plataforma. Los datos de los proveedores son los siguientes: Nombre, Dirección (País y Ciudad), Dirección de Correo Electrónico, Teléfono (opcional), Código de moneda, Nombre de moneda, Símbolo de moneda, Precio predeterminado.*	*Administradores*
RF 2	*Los eventos publicados por los Proveedores deben ser autorizados previo a su comienzo, de forma tal que los usuarios tengan tiempo para poder suscribirse a los eventos.*	*Administradores*
RF 3	*Los administradores deben poder consultar la bitácora de la actividad de la plataforma en un determinado periodo*	*Administradores*
RF 4	*La plataforma debe permitir crear un evento.*	*Proveedores*
RF 5	*Dado un evento y un proveedor, se debe poder modificar si el evento no está aprobado aún. Se deben realizar las mismas validaciones que se realizan en la creación de un evento.*	*Proveedores*
RF 6	*Se necesita que a través de la plataforma se pueda enviar un correo electrónico con información relevante de un evento a cada uno de los Clientes que se suscribieron y solicitaron que se les envíe información.*	*Proveedores*
RF 7	*Los usuarios deben poder obtener una lista de eventos disponibles que hayan sido aprobados y cuya fecha de fin no sea anterior a la fecha de consulta, con el objetivo de poder elegir uno.*	*Clientes*
RF 8	*Para realizar el pago del evento, se debe utilizar el proveedor que esté utilizando la Pasarela de Pagos. Como se mencionó en REQ 2 – Autorizar/Desautorizar un evento, se debe definir un protocolo de intercambio con proveedores de pago. Tener en cuenta que los proveedores de pago pueden cambiar y un cambio de proveedor debe ser fácil de realizar.*	*Clientes*
RF 9	*La plataforma debe permitir acceder a un video del evento en el horario estipulado solamente a los usuarios que pagaron por él. Es importante realizar un diseño utilizando mecanismos que permitan dotar a la plataforma con la capacidad de que el acceso a los eventos esté disponible para los clientes cuando lo necesiten y que el tiempo de acceso sea el menor posible.*	*Clientes*
RF 10	*Se debe retornar una lista con todos los eventos de un proveedor*	*Proveedores*
RF 11	*Para un evento de un proveedor se debe retornar información*	*Proveedores*
RF 12	*Para cada uno de los eventos que se están ejecutando actualmente (eventos activos) en la plataforma, se necesita obtener información*	*Proveedores*
RF 13	*Dado un período, se debe listar cierta información*	*Proveedores*
RF 14	*Dado un usuario, se debe listar cierta información tanto para los eventos a los que se suscribió como para los que aún puede suscribirse*	*Clientes*

RF 15	*En el caso de ocurrir una falla o cualquier tipo de error, es imprescindible que el sistema provea toda la información necesaria que permita a la empresa hacer un diagnóstico rápido y preciso sobre las causas.*	*Sistema*
RF 16	*Los endpoints disponibles deben estar protegidos y accesibles solamente a los usuarios mencionados en cada una de las definiciones de los requerimientos.*	*Sistema*
RF 17	*Debido a que la plataforma, una vez puesta en producción, puede verse saturada de solicitudes acceso o procesamiento en un momento dado del tiempo, se deberá lograr la mayor capacidad de procesamiento posible, sin pérdida de datos y logrando la mejor latencia posible.*	*Sistema*
RF 18	*La plataforma debe registrar información que permita realizar auditorías de acceso de forma de identificar los accesos autorizados y no autorizados al sistema.*	*Sistema*

2.2.2 Resumen de Requerimientos de Atributos de Calidad

ID Requerimiento	ID Requerimiento de Calidad o restricción	Descripción
RF1	*AC1 Usabilidad*	*Se espera que la API tenga una interfaz intuitiva y clara para que los administradores puedan completar la información requerida para dar de alta un proveedor de manera eficiente y sin dificultades.*
RF2	*AC2 Seguridad* *AC3 Disponibilidad* *AC5 Interoperabilidad*	*La seguridad sería importante para que solo los responsables de esta parte del sistema puedan acceder al mismo y se tiene que haber disponibilidad de los eventos publicados previamente por los proveedores para su uso. Ocurren interacciones entre varias apis.*
RF3	*AC3 Disponibilidad*	*Es fundamental que la bitácora esté siempre disponible para los administradores que necesiten consultarla.*
RF4	*AC3 Disponibilidad* *AC2 Seguridad*	*Se necesita asegurar que sean solamente los administradores los que operen en este RF y hay que corroborar que los campos sean válidos. Tiene que estar disponible cuando queramos crear.*
RF5	*AC2 Seguridad* *AC4 Modificabilidad*	*Se busca que el sistema sea capaz de permitir cambios en la información de un evento por parte de los proveedores, siempre y cuando el evento aún no haya sido aprobado por un administrador.*
RF6	*AC1 Usabilidad*	*Se busca que la plataforma sea fácil de usar para enviar mensajes relevantes a los clientes suscritos a un evento en particular.*
RF7	*AC3 Disponibilidad* *AC2 Seguridad*	*Es importante que la plataforma esté disponible y que el sistema de consulta de eventos funcione correctamente en todo momento. Proteger la información de los eventos y de los usuarios que acceden a la plataforma evitando amenazas o vulnerabilidades*
RF8	*AC1 Seguridad* *AC5 Interoperabilidad* *AC6 Performance*	*El sistema debe ser capaz de comunicarse con la Pasarela de Pagos y con los proveedores de manera eficiente y sin errores.*
RF9	*AC3 Disponibilidad* *AC6 Performance*	*La consulta debe estar disponible para permitir el acceso a los eventos a los clientes en el momento en que lo necesiten y el tiempo de acceso debe ser el menor posible.*

RF10	*AC3 Disponibilidad* *AC5 Interoperabilidad* *AC1 Seguridad* *AC6 Performance*	*Se debe realizar una consulta en un tiempo inferior a los 5 segundos y solamente los Proveedores harán uso de esta funcionalidad*
RF11	*AC3 Disponibilidad* *AC5 Interoperabilidad* *AC1 Seguridad* *AC6 Performance*	*Se debe realizar una consulta en un tiempo inferior a los 5 segundos y solamente los Proveedores harán uso de esta funcionalidad*
RF12	*AC3 Disponibilidad* *AC5 Interoperabilidad* *AC1 Seguridad* *AC6 Performance*	*Se debe realizar una consulta en un tiempo inferior a los 5 segundos y solamente los Administradores harán uso de esta funcionalidad*
RF13	*AC3 Disponibilidad* *AC5 Interoperabilidad* *AC1 Seguridad* *AC6 Performance*	*Se debe realizar una consulta en un tiempo inferior a los 5 segundos y solamente los Administradores harán uso de esta funcionalidad*
RF14	*AC3 Disponibilidad* *AC5 Interoperabilidad* *AC1 Seguridad* *AC6 Performance*	*Se debe realizar una consulta en un tiempo inferior a los 5 segundos y solamente los Administradores harán uso de esta funcionalidad*
RF15	*AC1 Usabilidad* *AC4 Modificabilidad*	*El sistema debe ser capaz de proporcionar información detallada y permitir la fácil modificación de herramientas o librerías para producir esta información, sin tener un impacto significativo en el código.*
RF16	*AC1 Usabilidad* *AC4 Seguridad*	*Se refiere a la protección de los datos y acceso a la plataforma por parte de los usuarios*
RF17	*AC1 Usabilidad* *AC2 Testeabilidad* *AC3 Disponibilidad* *AC6 Performance*	*Es necesario que la plataforma pueda manejar cargas de solicitudes y procesamiento en momentos de alta demanda, sin pérdida de datos ,con la menor latencia posible y sin perder ningún tipo de dato.*
RF18	*AC3 Disponibilidad* *AC4 Seguridad* *AC5 Interoperabilidad*	*Se busca registrar información que permita identificar accesos autorizados y no autorizados al sistema*

3. Documentación de la arquitectura

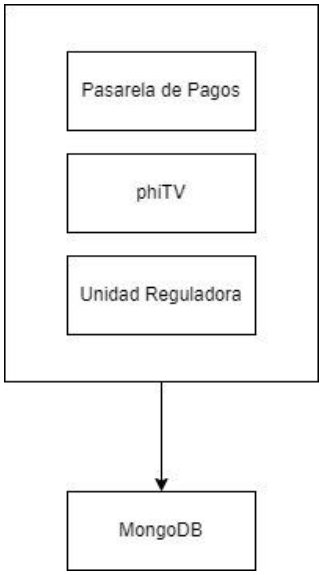
3.1 Vistas de Módulos

3.1.1 Vista de Descomposición

A continuación se muestran las vistas de Descomposición del sistema.

3.1.1.1 Representación primaria

Como podemos apreciar, nuestro sistema fue creado como un monolito. Hemos decidido esto porque los requerimientos de nuestro sistema implican una construcción sencilla y básica, por lo que decidimos crear un monolito por su simplicidad. Además, consideramos que era lo más adecuado ya que al momento de utilizar el sistema, tanto los clientes como los proveedores como los administradores utilizarian algunas de las 3 apis.

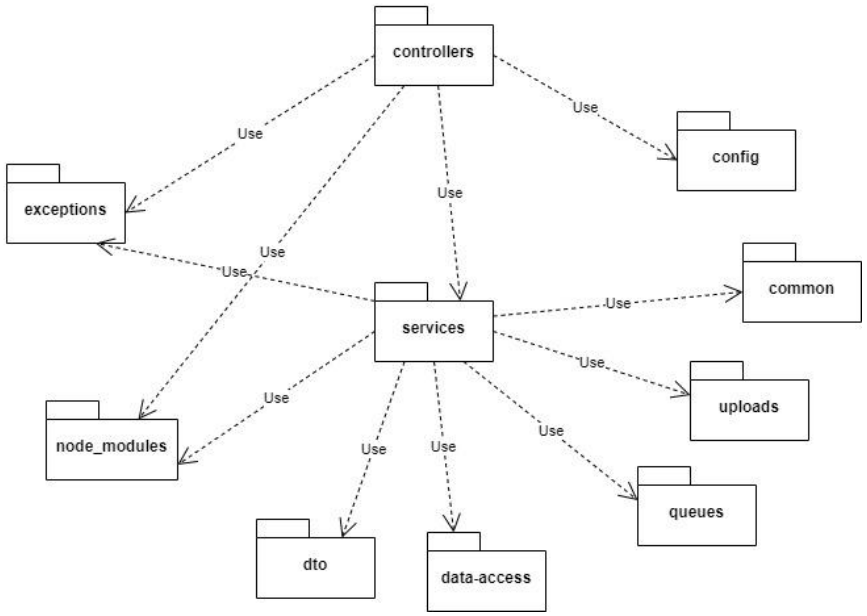


3.1.2 Vista de Uso

A continuación se describen las Vistas de Uso del sistema

3.1.2.1 Representación primaria
3.1.2.1.3 phiTV

Modulos
comunes



3.1.2.2 Decisiones de diseño

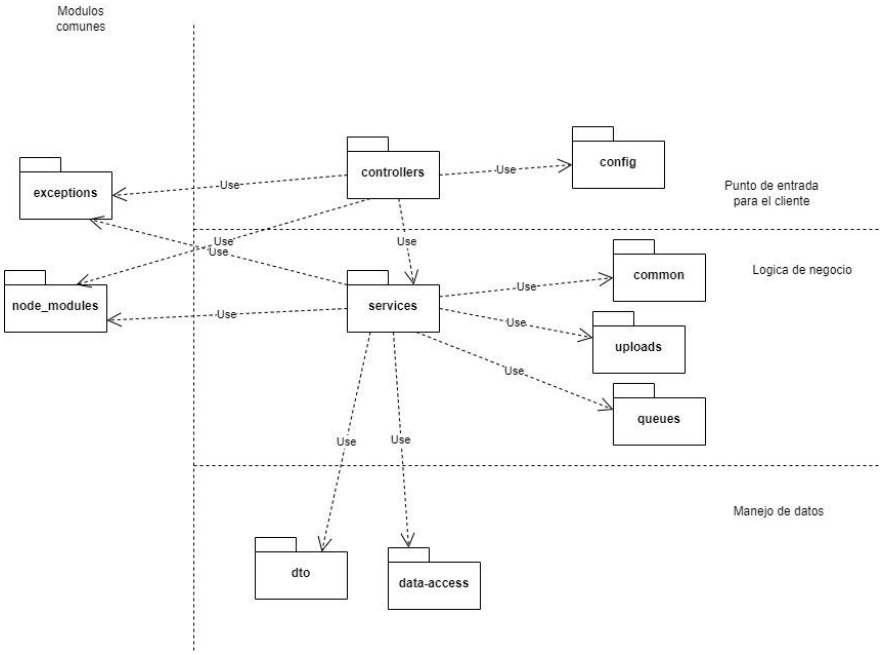
Para realizar nuestra aplicación decidimos desarrollar una aplicación monolítica basada en dos módulos: el sistema que se compone de 3 Apis a las cuales el cliente puede acceder y por último el sistema de DataAccess donde se van a guardar los datos, que es MongoDB. Dentro del sistema, el punto de inicio es un archivo llamado app.js, ahí se realizan las configuraciones iniciales, como pasarle la lógica a los controllers, la cual serían los services dentro de la carpeta services e iniciar los controllers y la base de datos. No se indica en el diagrama, pero dentro de services se encuentra implementado un Pipes & Filters para resolver el Req 9, así como las queues utilizadas por los logs para satisfacer el Req 3.

El motivo de uso de una base de datos como MongoDB fue debido a su alto rendimiento. Se pedía que las consultas no duraran más de 5 segundos en realizarse y por este motivo se utilizó esta base de datos

3.1.3 Vista de Layers

3.1.3.1 Representación primaria

Utilizamos el patrón de arquitectura layers debido a que necesitábamos tener una separación de responsabilidades a la hora de operar entre los controladores, el servicio y la base de datos.



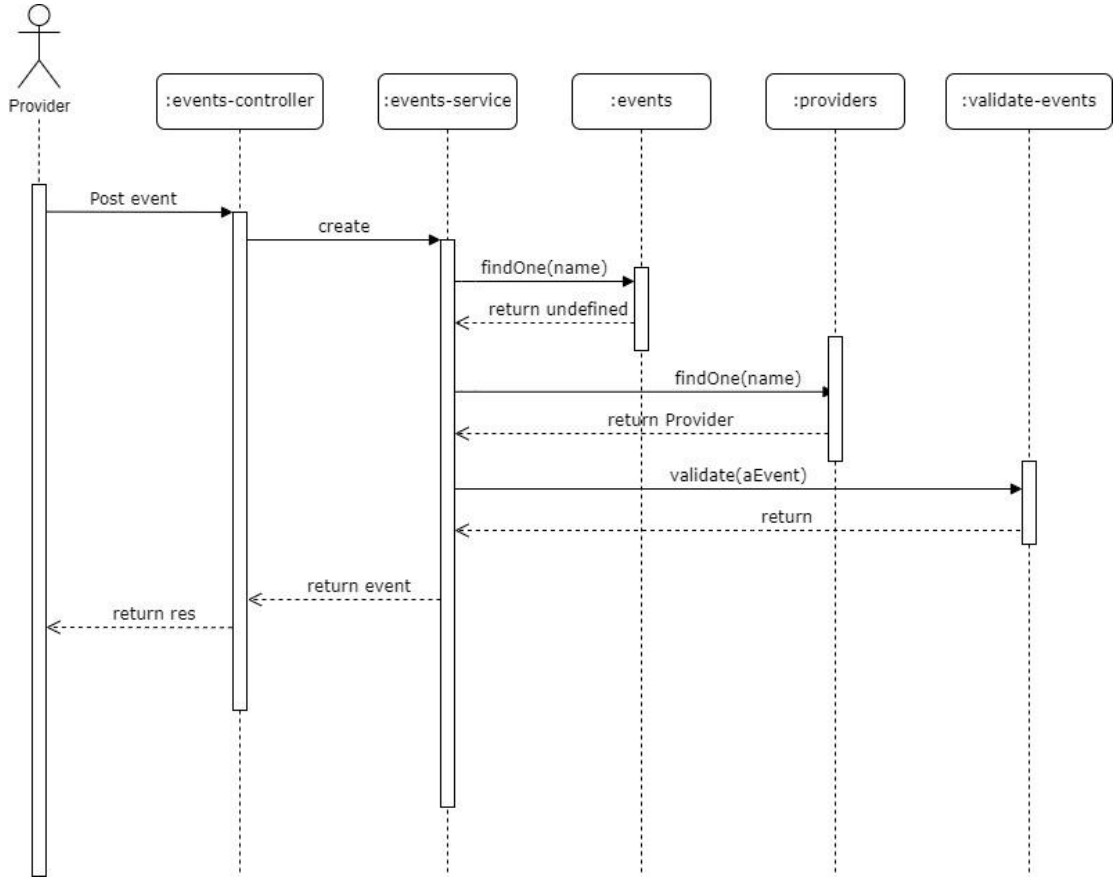
3.1.4 Catálogo de elementos

Elemento	Responsabilidades
controllers	***En este elemento se encuentran los controllers que reciben las consultas de los endpoints, es el punto de acceso del cliente***
common	***Aquí se encuentran archivos de validación y tokens usados por los servicios de la aplicación para efectuar tareas de autenticación y validación***
node_modules	***En esta carpeta están los módulos npm utilizados dentro de la aplicación, todas las librerías, componentes y recursos utilizados se encuentran aquí***
config	***Aquí se encuentra el archivo default.json, donde se especifica la ruta a la API así como las diferentes rutas dentro de cada módulo para poner realizar las consultas a los controllers***
services	***Aquí se realiza la lógica de negocio, recibiendo las consultas de los controllers, validándolas y obteniendo la información necesaria de la base de datos de nuestro sistema, y generando la response de la consulta***
exceptions	***Aquí están las excepciones que podrían ocurrir con nuestro sistema, usadas para informar al usuario de que ocurrió un error***
queues	***Aquí se encuentra la implementación de una cola de mensajes en bull, utilizada por la lógica para guardar información según requiera***

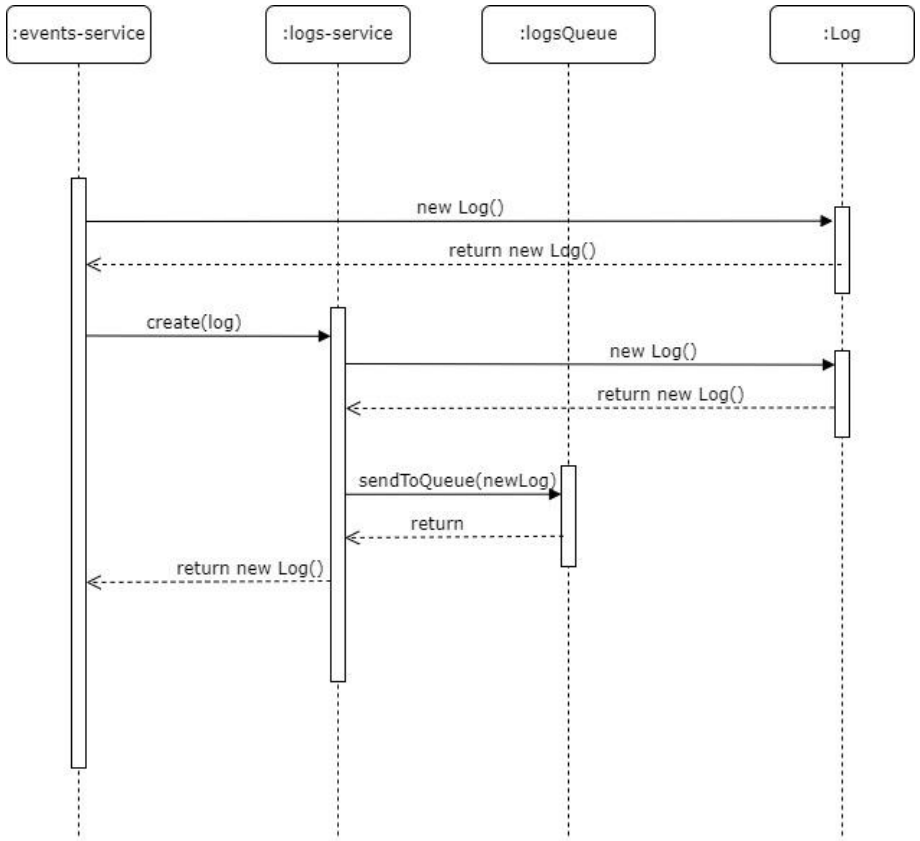
dto	***Este módulo se usa para retornar una response de un get con más de un elemento***
data-access	***En esta carpeta están los modelos de los elementos del sistema, así como la conexión con la base de datos MongoDB***
uploads	***En esta carpeta se ubicaran los videos que se le dará a los clientes que pagaron, al cual pueden acceder en cualquier momento***

3.1.5 Comportamiento

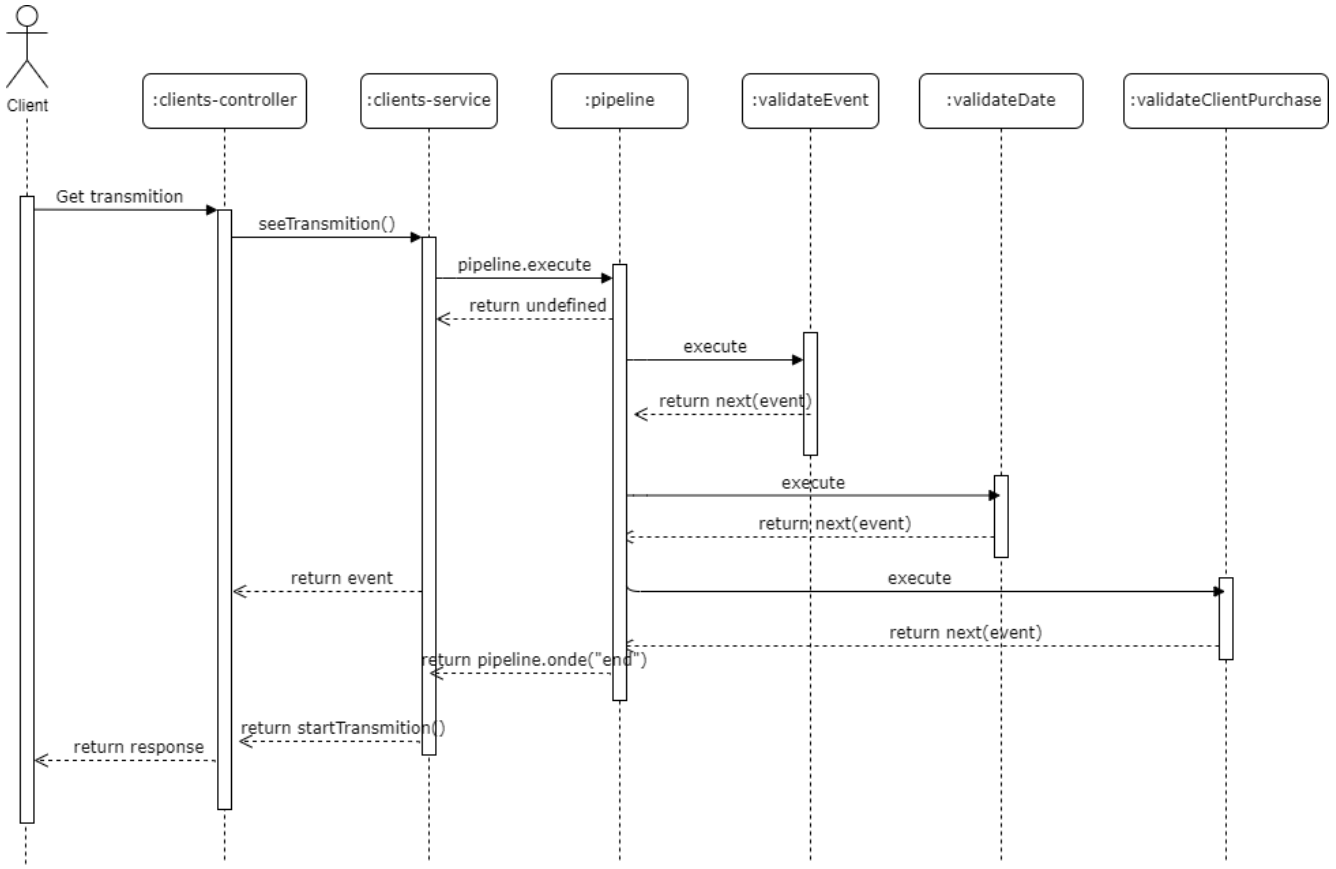
Agregar un evento:



Agregar una entrada a la bitácora al crear un evento:



Ver transmision de evento:



3.1.6 Decisiones de diseño

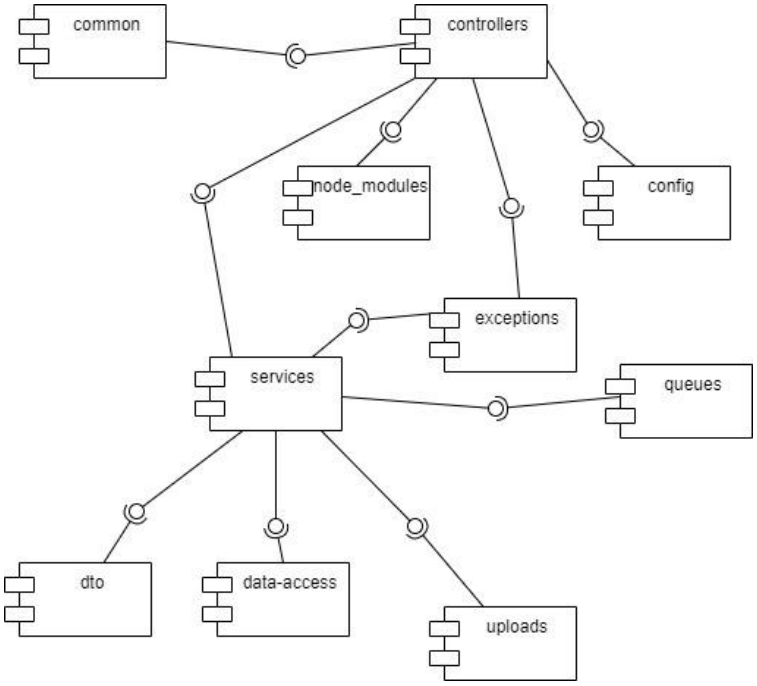
Se decidió utilizar el patrón de arquitectura pipes and filters porque en el requerimiento 9 se necesitaba hacer varias validaciones y la cantidad de clientes concurrentes podía crecer rápidamente. Entonces decidimos usar este patrón para favorecer la performance y la modificabilidad

También utilizamos una cola de mensajes para el requerimiento 3 debido a que este pedía crear una bitácora de la plataforma, la cual no era necesario acceder inmediatamente después de realizar las determinadas acciones. Esto nos ayudó a mejorar la performance de la aplicación.

3.2 Vistas de Componentes y conectores

En esta sección procederemos a mostrar nuestra solución desde el punto de vista del Runtime, es decir, cómo se comporta el sistema en tiempo de ejecución, cuáles son sus componentes en dicho momento y cómo éstos se conectan/comunican.

3.2.1 Representación primaria



3.2.2 Relación con elementos lógicos

Componente	Paquetes
controller	*exceptions* *config* *node_modules* *services*
services	*exceptions* *data-access* *dto* *queues* *uploads*

3.2.3 Decisiones de diseño

El punto de inicio de la aplicación es app.js, como se mencionó anteriormente, decidimos implementarlo de esta manera de tener todo en un mismo lugar, y poder iniciar la conexión en un solo lugar, y que después cada componente vaya llamando a los recursos que necesita. Elegimos así ya que consideramos que era la mejor manera de no dejar componentes sin uso y maximizar el rendimiento de nuestra aplicación. También de esta manera no acoplamos los endpoints a la lógica, sino que dejamos que app.js realice las

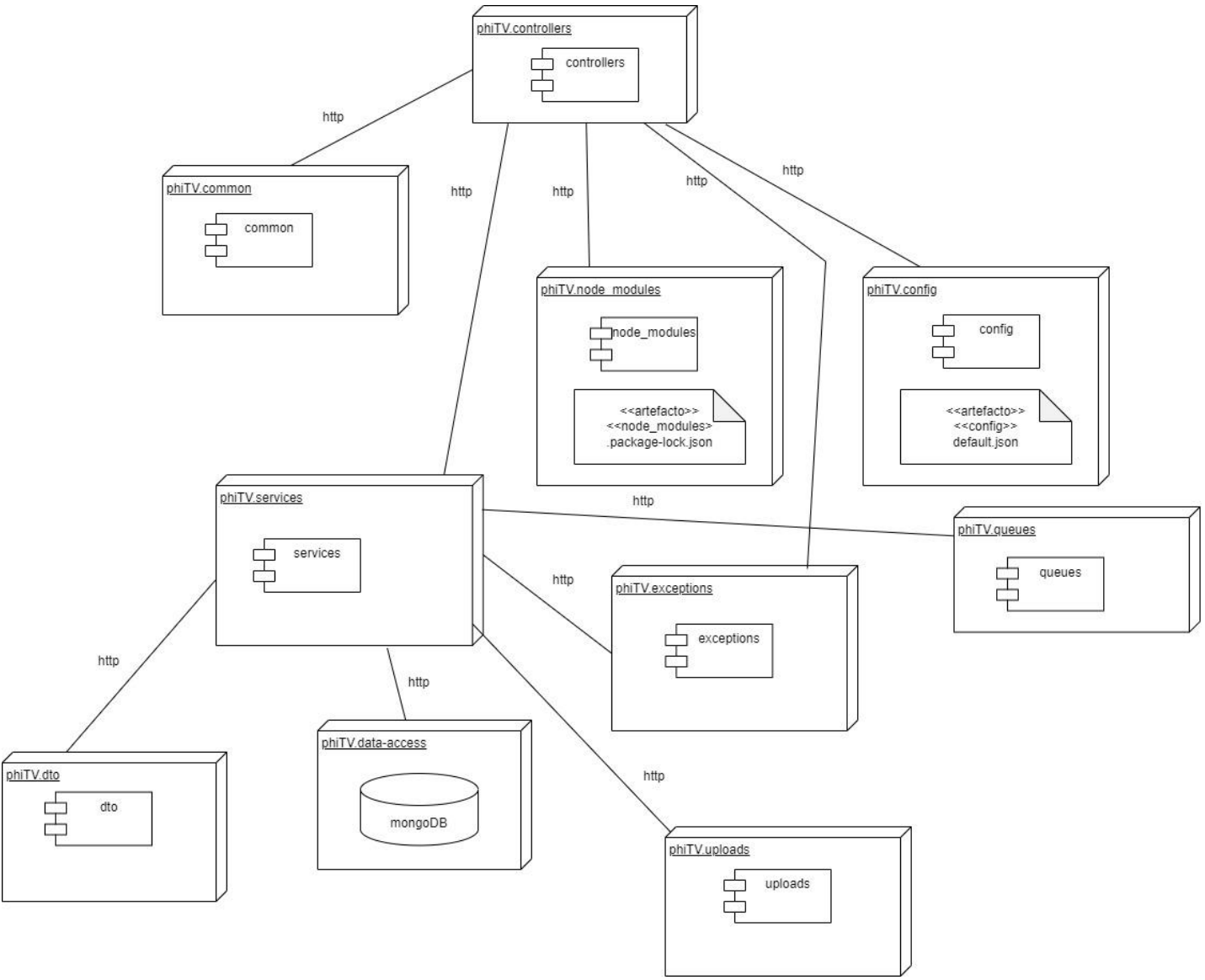
inicializaciones y dependencias, de forma que si las mismas cambian solo cambiamos el parámetro dentro de app.js, permitiendo una arquitectura mucho menos rígida y fácil de modificar.

3.3 Vistas de Asignación

En esta vista se intenta mostrar la asignación de los distintos componentes del sistema a los nodos correspondientes en la arquitectura.

3.3.1 Vista de Despliegue

3.3.1.1 Representación primaria



3.3.1.2 Catálogo de elementos

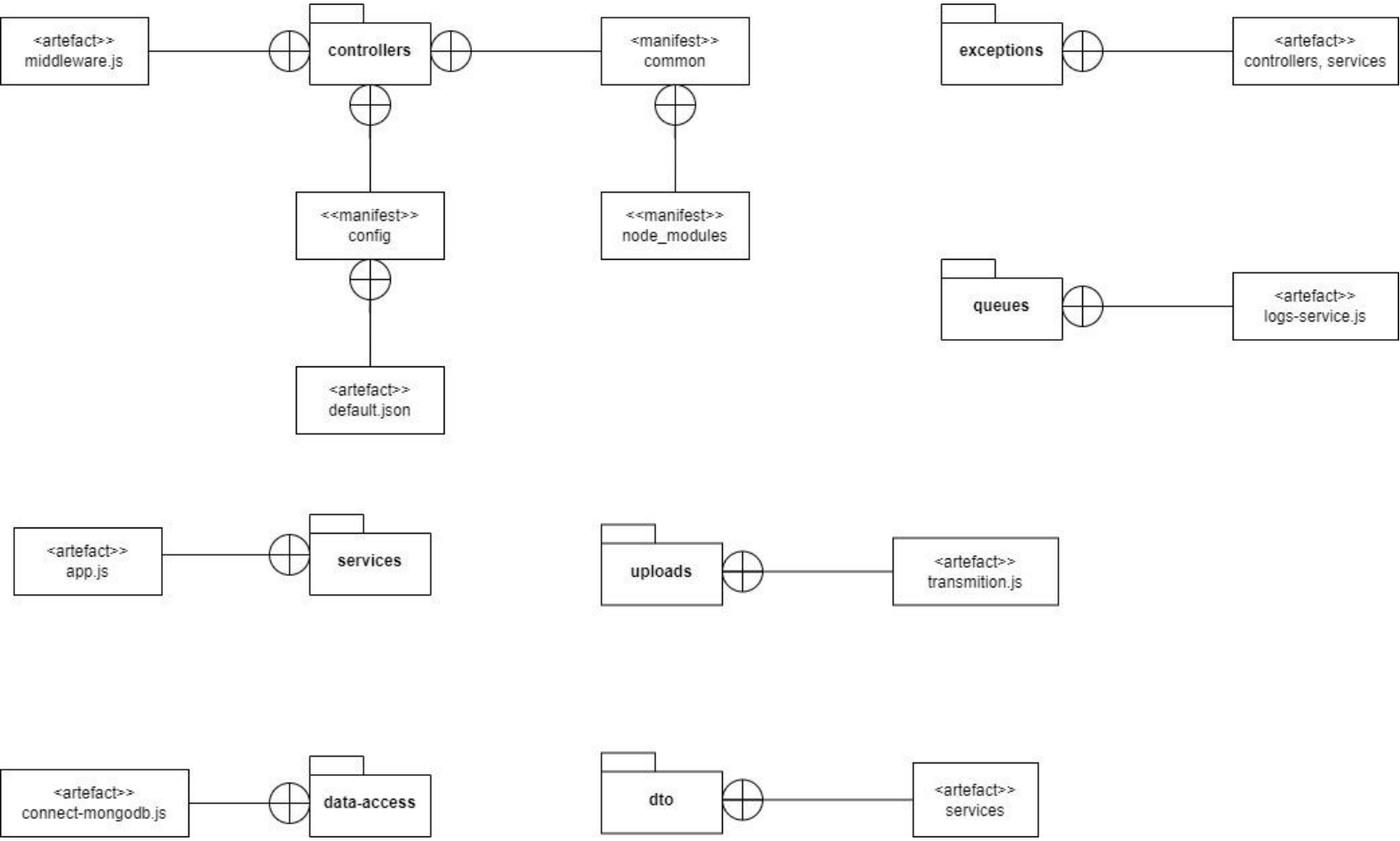
Nodo	Descripción
phiTV.common	Nodo que guarda los módulos comunes al elemento
phiTV.controllers	Nodo que almacena los controllers
phiTV.node_modules	Nodo que guarda los módulos de node.js utilizados
phiTV.config	Nodo que guarda los archivos de configuración del sistema

phiTV.services	Nodo que guarda los servicios del sistema
phiTV.queues	Nodo que guarda las queues utilizadas
phiTV.exceptions	Nodo que guarda las excepciones que puedan ocurrir
phiTV.dto	Nodo que guarda el esquema de las response
phiTV.data-access	Nodo que contiene la base de datos del sistema
phiTV.uploads	Nodo que contiene los videos a ser devueltos al cliente
default.json	Artefacto que contiene las rutas de los endpoints de las APIs
.package-lock.json	Artefacto que contiene todos los packages npm usados por el sistema

Conector	**Descripción**
http	HyperText Transfer Protocol - Es el protocolo utilizado para comunicarse con los módulos de nuestra aplicación

3.3.2 Vista de Instalación

3.3.2.1 Representación primaria



Se puede visualizar todos los archivos de configuración, archivos para iniciar cada servicio y el package.json que contiene todas las dependencias de cada servicio.

3.3.2.2 Catálogo de elementos

Nodo	Descripción
app.js	Punto de inicio de la aplicacion, desde aqui se inician los servicios de la misma
middleware.js	El middleware se encarga de asignar la lógica a los controllers
common	Aquí hay archivos comunes a toda la aplicación
config	Aquí se encuentran todos los archivos relacionados a la autorización y autenticación de los datos
node_modules	Aquí están los módulos propios de node.js que se usaron en la aplicación
default.json	En este archivo estan las rutas de los endpoints de los controllers

events-service.js	Este es el servicio que maneja la lógica de los eventos, que es el único que usa los logs
transmition.js	Aquí se maneja el envío de los videos a los clientes que lo pagaron y solicitaron
connect-mongo.js	Esta clase se encarga de realizar la conexión a la base de datos mongo
services	Aqui estan ubicados todos los servicios de nuestro sistema

3.3.2.3 Decisiones de diseño

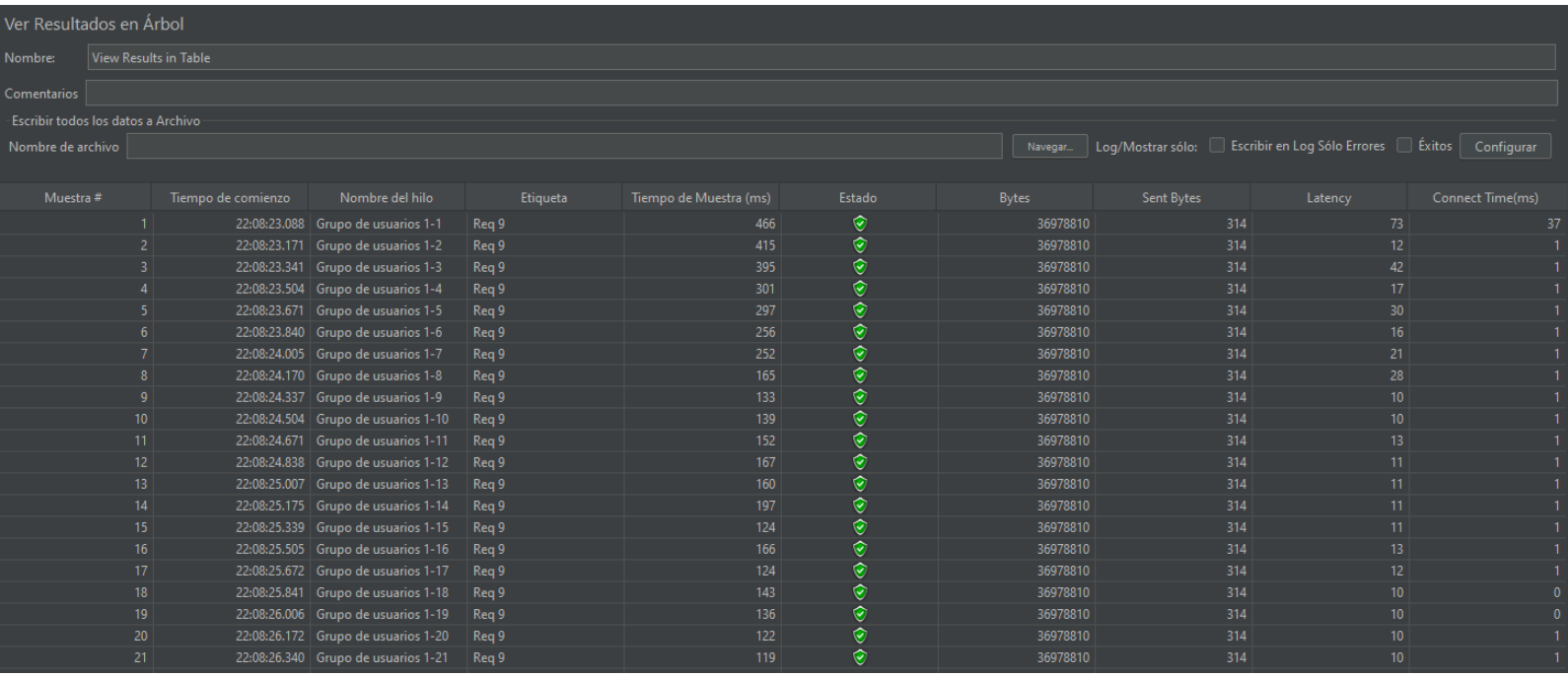
Como nuestra aplicación sigue una arquitectura monolítica tiene un único punto de inicio, que es app.js, la cual es la que se encarga de iniciar los servicios de la misma y de pasarlos al middleware de los controllers. Asi como app.js inicia los services, middleware.js hace lo mismo con los controllers, y también usa los elementos de config, node_modules y common para funcionar. Exceptions tiene a controllers y a service como iniciadores porque son los elementos que lo usan. Queues solo es usado por logs-service, por lo que es el único que lo llama, así cómo uploads solo es usado por transmition.

4. Instrucciones para correr la aplicación

1. En los componentes de Pasarela de Pagos, Unidad Reguladora y phiTV correr el comando **npm install** para descargar las dependencias de los proyectos.
2. Tener instalado Docker Desktop
3. Tener instalado mongo en el puerto 27027 version 4.18.0 en docker
4. Utilizar el comando **node index** para las apis de Pasarela de pagos y Unidad reguladora.
5. Utilizar el comando **node app** para phiTV

Se dejan en el anexo los archivos .env necesarios para las apis.

5. Anexo


Ver Resultados en Árbol

Nombre: View Results in Table

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo Navegar... Log/Mostrar sólo: Escribir en Log Sólo Errores Éxitos Configurar

Muestra #	Tiempo de comienzo	Nombre del hilo	Etiqueta	Tiempo de Muestra (ms)	Estado	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	22:08:23.088	Grupo de usuarios 1-1	Req 9	466		36978810	314	73	37
2	22:08:23.171	Grupo de usuarios 1-2	Req 9	415		36978810	314	12	1
3	22:08:23.341	Grupo de usuarios 1-3	Req 9	395		36978810	314	42	1
4	22:08:23.504	Grupo de usuarios 1-4	Req 9	301		36978810	314	17	1
5	22:08:23.671	Grupo de usuarios 1-5	Req 9	297		36978810	314	30	1
6	22:08:23.840	Grupo de usuarios 1-6	Req 9	256		36978810	314	16	1
7	22:08:24.005	Grupo de usuarios 1-7	Req 9	252		36978810	314	21	1
8	22:08:24.170	Grupo de usuarios 1-8	Req 9	165		36978810	314	28	1
9	22:08:24.337	Grupo de usuarios 1-9	Req 9	133		36978810	314	10	1
10	22:08:24.504	Grupo de usuarios 1-10	Req 9	139		36978810	314	10	1
11	22:08:24.671	Grupo de usuarios 1-11	Req 9	152		36978810	314	13	1
12	22:08:24.838	Grupo de usuarios 1-12	Req 9	167		36978810	314	11	1
13	22:08:25.007	Grupo de usuarios 1-13	Req 9	160		36978810	314	11	1
14	22:08:25.175	Grupo de usuarios 1-14	Req 9	197		36978810	314	11	1
15	22:08:25.339	Grupo de usuarios 1-15	Req 9	124		36978810	314	11	1
16	22:08:25.505	Grupo de usuarios 1-16	Req 9	166		36978810	314	13	1
17	22:08:25.672	Grupo de usuarios 1-17	Req 9	124		36978810	314	12	1
18	22:08:25.841	Grupo de usuarios 1-18	Req 9	143		36978810	314	10	0
19	22:08:26.006	Grupo de usuarios 1-19	Req 9	136		36978810	314	10	0
20	22:08:26.172	Grupo de usuarios 1-20	Req 9	122		36978810	314	10	1
21	22:08:26.340	Grupo de usuarios 1-21	Req 9	119		36978810	314	10	1

La cantidad máxima de clientes concurrentes que pudimos obtener fue de 21 de 30 por las capacidades de nuestro sistema, si se quisieran utilizar mas tendriamos que escalar con nuestros componentes

Obtuvimos con jmeter los tiempos de espera de todas las consultas de los clientes.

Aca vemos un ejemplo de como calculamos el tiempo de espera de un cliente

Tiempo de espera 1 = Tiempo de carga 1 + Connect Time 1 + Latencia 1 = 466 + 37 + 73 = 576 ms

Sumamos todos los tiempos de espera e hicimos el promedio

Suma de los tiempos de espera = 576 + 428 + 438 + 319 + 328 + 273 + 274 + 194 + 144 + 150 + 166 + 179 + 172 + 209 + 136 + 180 + 137 + 153 + 146 + 133 + 130 = 4672 ms

Promedio de tiempo de espera = 4672 ms / 21 = 222.476 ms

Por lo tanto, el promedio de tiempo de espera entre los 21 datos que obtuvimos es de aproximadamente 222.476 ms.

Tambien hicimos el calculo del promedio de compra con jmeter.

De 12 compras se calculó el promedio en que se demoraba en hacer una compra y los resultados fueron los siguientes:

Esta es la suma de todos los valores de "Load time":
115 + 75 + 111 + 111 + 109 + 109 + 70 + 92 + 113 + 103 + 102 + 139 = 1209

Y este es el promedio de todas esas muestras
1209 / 12 = 100.75

Estos valores fueron hardcodeados al sistema para utilizarse al momento de realizar las querys.

Módulos NPM

Elemento	Responsabilidad
axios	Cliente http para node
bull	Implementación de colas de mensajes utilizando un servidor de mongo como backend
currency-codes	Validar los códigos de las monedas
currency-iso	Validar nombres y símbolos de monedas
dotenv	
express	Servicio http extensible para publicar endpoints de una api.
jsonwebtoken	Una implementación de JSON Web Tokens.
moment	Una biblioteca de fechas de JavaScript para analizar, validar, manipular y formatear fechas
mongoose	Object document mapper para node
node-cron	Utilizado para las autorizaciones automáticas y agendar cuándo ejecutar código
nodemailer	Nodemailer es un módulo para aplicaciones Node.js que permite enviar correos electrónicos de forma sencilla
pipes-and-filters	Utilizado para hacer uso del patrón pipes and filters

Archivos .env

Pasarela de Pagos

```
MYSQL_DB = "mysql://root:secret@localhost:3306/mysql-db"
PORT = 5000
BASE_URL = "/api"
PAGING_DEFAULT_RECORD_OFFSET = 0
PAGING_DEFAULT_RECORD_LIMIT = 10
PAGING_DEFAULT_MAX_RECORD_LIMIT = 500
MONGO_CONNECTION_STRING =
"mongodb://admin:secure@localhost:27017/admin"
QUEUE = "providerPaymentGateways"
```

Unidad Reguladora

```
MYSQL_DB = "mysql://root:secret@localhost:3306/mysql-db"
PORT = 4000
BASE_URL = "/api"
PAGING_DEFAULT_RECORD_OFFSET = 0
```

```
PAGING_DEFAULT_RECORD_LIMIT = 10
PAGING_DEFAULT_MAX_RECORD_LIMIT = 500
MONGO_CONNECTION_STRING =
"mongodb://admin:secure@localhost:27017/admin"
QUEUE = "providerRegulatoryUnits"
```

phiTV

```
MYSQL_DB = "mysql://root:secret@localhost:3306/mysql-db"
PORT = 3000
TRANSMITION_PORT = 3300
BASE_URL = "/api"
PAGING_DEFAULT_RECORD_OFFSET = 0
PAGING_DEFAULT_RECORD_LIMIT = 10
PAGING_DEFAULT_MAX_RECORD_LIMIT = 500
MONGO_CONNECTION_STRING =
"mongodb://admin:secure@localhost:27017/admin"
QUEUE = "events"
JWT_EXPIRES = "12h"
PRIVATE_KEY =
"MIIEowIBAAKCAQEA00g5GZEe9vLQjwMmwaKWec8ODr3ems0quqWkYjlLUS6lQvOy
5GkEtnyrYCY8pZpv3syUFcX2h7trKgPycMVtfqpNJoV7bQh/1gkcev7i5ramthK0
clyS/k6PCFXDpvGPUombs+1lt/g0xp9OMqJmIETOuYbQE1IzoPRwDrLmuP0cVN//
E9Sl6k4+BtYOBTuAfGHnMEVTXlNFYRQOxAAl98J3yifJPNlsqZeyy7+uK8z+HmbW
qp3sWjXAJPJ4cYtT6ioBpeoPIiwCuRjpsF2wT8incCJPFKkL96+aB803YENDfkLt
FjxxZW23JPylgUAWziFw31gdMaXYtLgHloU+PQIDAQABAoIBADGPdn1t0jKhQJTb
a2LujZejHywME8SCLey8YDP+NpGjfHaZDbSQgd+AnP6f2YgLEwTTPkVhO3QZXYjs
hhk35nhY4GIR26L25G3Dk1+ynGz8GV0KRDOzTXPD379Xxt5JOZJl3xU0sbCW02Jo
CgLMn2IgYdGZMwPlO2EdFZANHRv1LgxddMbjo8Jyid662lPfBW4lDEJRgsJyNuqd
37LVTt8EHvw5LyZRDeKCa7ybdxonVhaJHAy905Rj4evu3A1TvsIRkqijQFgbnevD
jJCIx+tbA1W/gq4njfCbAUnpmj78als3wZFtSnSjgwJPF5dv/FWVukPrgQmFOHSS
6WmiauECgYEA/MQrV8RD67nsnEAMR8EZNaCJnqooqSveEWjto4/xeW4JfieJXtbO
/cJF04Zv2fHo/wtyNKQHdr5UGJSFVMfRpSTLxQ12shp7dmIQE0vVMHHOFWrc02VR
wkNr6M7V6ChMyxWd50B4VkRfZd+7nKWGCR8kE6xgPovMGrPgNvkVwxkCgYEA1fww
i9ySiBi60V13HDKxYglQqHQH9uIGvY5Cheb9NdmFldSj6+FzF15H3B95THsPvRlD
fWA7CAxLl9Bb9EAAY1rE/NRNLen3J5XoJJvDl4c0D2sfAjF+Wm/03FqjrXZdmBJI
Ukzzavq45UgVUsDsVNSbWXmLsgB8eLch8882fMUCgYEAv1eU4/Sr4ec16TZG1Y6k
kcb8RsLS0vVlPtKqihWuG1kkEH7Ha6a6vLfy+eQcCeUpluBITPcau1Xq/Ux4/ivh
VN1c78d4YezQJ1amiqTuNqT/95CUp+ZcFO3K40qn/P183IN5rBROxCcv/EN4l2Vp
6L/NJKNpnA9hq5kdEyBNzAkCgYBLEj610pxVwtdHvFKdQeuZSUtSzeI9au6SceZE
s9UBn16OQhuAFYgbGqh7QU981W+gF8zln20WVbnveNnJxupb8OxLBvBMu1hKOv1G
Wp8z+er5vcsTKen6MGznIaVqpawQPdnBhd2gGnXjenaAoc5HtI8LxOiNej4w2469
ymSDQQKBgGoIKGvIaaUxeq9uH7HjsX5PYvHt4s2cCWU7CKdznW79N8xurF0s9/0H
iH5C7hEnsuRWtMcVlPDHF/ZzMfLRKQuUGxexuFNYx6R2qho6EeSCQ8V1aKA4XSpH
```

```
TUrCo6ifEeXNawZL8/Ey31cpbs1ZtpmjfB7576nYpkv0UqzJTczG"
PUBLIC_KEY                                                    =
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA00g5GZEe9vLQjwMmwaKW
ec8ODr3ems0quqWkYjlLUS6lQvOy5GkEtnyrYCY8pZpv3syUFcX2h7trKgPycMVt
fqpNJoV7bQh/1gkcev7i5ramthK0clyS/k6PCFXDpvGPUombs+1lt/g0xp9OMqJm
IETOuYbQE1IzoPRwDrLmuP0cVN//E9Sl6k4+BtYOBTuAfGHnMEVTXlNFYRQOxAAl
98J3yifJPNlsqZeyy7+uK8z+HmbWqp3sWjXAJPJ4cYtT6ioBpeoPIiwCuRjpsF2w
T8incCJPFKkL96+aB803YENDfkLtFjxxZW23JPylgUAWziFw31gdMaXYtLgHloU+
PQIDAQAB"
```