

Universidad de los Andes  
Maestría en Ingeniería de Software  
Arquitecturas Ágiles de Software  
Septiembre 12 de 2021

## Informe de Resultados del Experimento

**Por:**

Laura Silva Ayala  
Andrés Fernando Barón  
Cesar Orlando Saavedra  
Félix E. Acero J.

### Presentación:

Este experimento es un MVP que se titula: Implementación Monitor/Elección de líder para garantizar un nivel de disponibilidad. Con él buscamos demostrar que mediante el uso de monitor / elección de líder se puede garantizar una disponibilidad cercana al 99%, ocultando posibles fallas en el sistema que puedan impactar este valor.

## Objetivo del Experimento:

Demostrar que mediante el uso de monitor / elección de líder se puede garantizar una disponibilidad cercana al 99%, ocultando posibles fallas en el sistema que puedan impactar este valor.

## Punto de Sensibilidad:

El uso de monitores para designar un líder que organiza la implementación de una tarea ayuda a ocultar y manejar la carga de un sistema ante posibles fallos o caídas de un sistema distribuido. Con esta táctica se busca demostrar que el sistema puede alcanzar una **disponibilidad del 99%**.

## Historia de Arquitectura Asociada:

HU001 – CONSOLIDACIÓN DEL VALOR A COBRAR A UN PACIENTE
<b>Como</b> empleado administrativo, <b>cuando</b> ingrese a "Generar consolidado", <b>dado que</b> el sistema esta condición de trabajo normal, <b>quiero</b> tener disponible los gastos generados en exámenes, medicamentos, procedimientos realizados y consultas que un paciente haya utilizado, <b>para</b> emitir una factura con los gastos asociados. Esta acción <b>debe</b> estar disponible el 99% de las veces que se solicite.

## Funcionalidad Desarrollada:

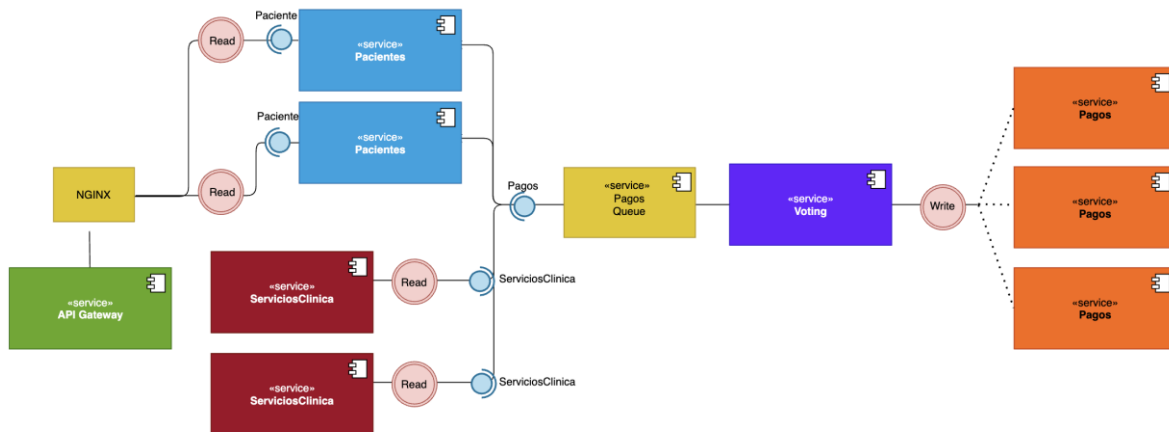
Se diseño un componente de votación y elección de líder, en los cuales se asignaba la carga a un nuevo componente de pagos en el caso de que se presentara una caída. Para simular una pasarela de pagos se simuló con timers y sistemas de persistencia , los registros exitosos asociados a la generación de pagos.

## Táctica de Arquitectura utilizada:

La táctica utilizada por el equipo fue **“Voting”**, también es conocida como redundancia modular triple. Los componentes escogen un líder que se encargara de procesar las respuestas y en caso de que este falle, mediante algoritmos de consenso como Paxos o Raft los servidores seleccionan un nuevo líder encargado de procesar la tarea. Las decisiones de diseño para esta parte fueron el necesitar enmascarar la falla de un posible componente de pagos sin embargo al ser un tema de dinero solo queremos que el pago se procese una vez y no en tres servidores al tiempo.

Esta elección de Voting la realizamos en el sistema propuesto de pagos dado que es el que va a recibir múltiples solicitudes pero que deben atenderse una única vez.

## Vista Funcional:



*Figura 1. Diseño de vista funcional donde se evidencia el sistema de colas propuesto y el componente de voting implementado.*

## Nivel de Incertidumbre del Experimento:

Alta. El equipo no se ha enfrentado de manera regular a la implementación de monitores / elección de líder para favorecer disponibilidad, por lo que este desarrollo experimental se basa principalmente en conocimiento teórico adquirido en la literatura disponible.

## Análisis de los resultados obtenidos:

### Hipótesis de Diseño:

La hipótesis planteada por el equipo de trabajo comprobó satisfactoriamente. Esto basado en los resultados obtenidos al momento de realizar el experimento, allí se evidenció que el hacer uso de la reasignación de líderes en los servidores y monitoreo constante, permiten que se puedan aceptar nuevas solicitudes con la certeza de que estas no lleguen a servidores que no se encuentran disponibles, dado que el monitor estará pendiente de su funcionalidad, lo que permitirá minimizar las fallas en los servicios prestados.

## Pruebas realizadas:

A lo largo del desarrollo del experimento se realizaron diferentes pruebas para verificar el desempeño y evidenciar si se estaba cumpliendo con lo planteado inicialmente, finalmente se realizaron 5 pruebas, para cada una de ellas se planteó lo siguiente:

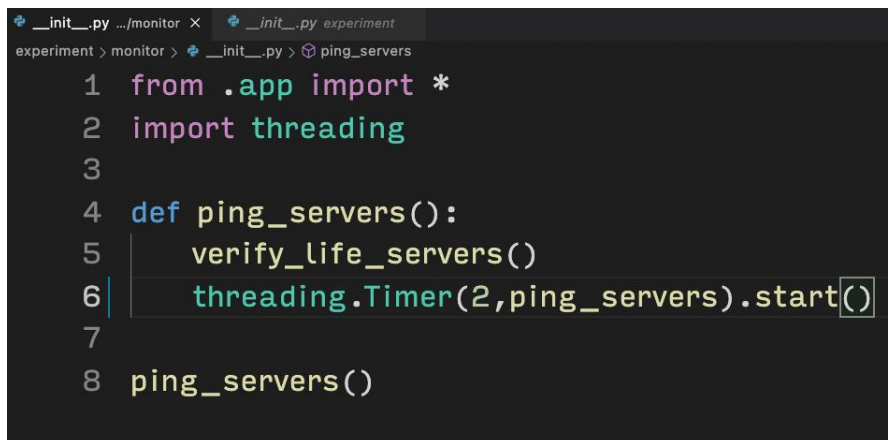
- Realizar el envío de 230 solicitudes al servicio de pagos
- Forzar la falla de manera aleatoria en los servidores designados para completar los procesos de pagos 10 veces en cada intento.



```
21
22 @app.route('/payment',methods=['POST'])
23 def run_experiment():
24     lis = []
25     create_database()
26     for iteration in range(1,230):
27         lis.append(payment(iteration,
28                             data_factory.name().split(" ")[0],
29                             data_factory.name().split(" ")[1],
30                             data_factory.email(),
31                             "+57 " + str(random.randint(1000000, 9000000)),
32                             random.randint(1000000, 9000000),
33                             False))
34     return jsonify(lis)
35
```

Figura 2. Función de simulación de 230 request por cliente simulado.

El servicio fue desplegado de acuerdo a la táctica de arquitectura definida previamente en 3 entornos diferentes e independientes, y ajustamos el tiempo del timer a 2 segundos. Este timer es el tiempo en el que el componente monitor va a validar la vida de los tres servidores de pagos mediante la táctica de disponibilidad PING/ECHO.



```
__init__.py .../monitor x  __init__.py experiment
experiment > monitor > __init__.py > ping_servers
1 from .app import *
2 import threading
3
4 def ping_servers():
5     verify_life_servers()
6     threading.Timer(2,ping_servers).start()
7
8 ping_servers()
```

Figura 3. Set-up de tiempo asociado a la validación de vida de los servidores de pagos.

## Proceso de la prueba:

1. Iniciamos el servidor de pacientes en el puerto 4000 y se espera hasta que este se encuentre disponible:

```
(venv) experiment % flask run -p 4000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:4000/ (Press CTRL+C to quit)
```

*Figura 4. Servidor de pacientes corriendo en el puerto 4000.*

2. Iniciamos el monitor, en este momento se puede evidenciar que los servidores destinados para realizar el servicio de pagos no se encuentran disponibles. Esto:

```
(venv) monitor % flask run
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
I AM OFF from server 1
I AM OFF from server 2
I AM OFF from server 3
The leader is server_2
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
I AM OFF from server 1
I AM OFF from server 2
I AM OFF from server 3
The leader is server_2
```

*Figura 5. Servidor de monitor levantado en el puerto 5000.*

3. Iniciamos los tres componentes de pagos, estos son desplegados en los puertos 5001, 5002, 5003 respectivamente, se espera que hasta que el servidor se encuentre disponible:

### Pagos1

```
(venv) pagos1 % flask run -p 5001
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5001/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Sep/2021 00:16:10] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:12] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:14] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:17] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:19] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:21] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:23] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:25] "POST /ping HTTP/1.1" 200 -
```

*Figura 6. Servidor de pagos 1 recibiendo validaciones de ping/echo.*

## Pagos2

```
(venv) pagos2 % flask run -p 5002
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5002/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Sep/2021 00:16:17] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:19] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:21] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:23] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:25] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:27] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:29] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:31] "POST /ping HTTP/1.1" 200 -
```

Figura 7. Servidor de pagos 2 recibiendo validaciones de ping/echo.

## Pagos3

```
(venv) pagos3 % flask run -p 5003
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5003/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Sep/2021 00:16:25] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:27] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:29] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:31] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:33] "POST /ping HTTP/1.1" 200 -
127.0.0.1 - - [12/Sep/2021 00:16:35] "POST /ping HTTP/1.1" 200 -
```

Figura 8. Servidor de pagos 3 recibiendo validaciones de ping/echo.

4. En este momento se evidencia que el monitor ha asignado un líder al servidor que se encuentra disponible y atendiendo solicitudes, en este caso es el server\_3 (pagos2):

```
I AM OFF from server 1
I AM ALIVE from server 2
I AM ALIVE from server 3
The leader is server_2

I AM ALIVE from server 1
I AM ALIVE from server 2
I AM ALIVE from server 3
The leader is server_2

I AM ALIVE from server 1
I AM ALIVE from server 2
I AM ALIVE from server 3
The leader is server_2
```

Figura 9. Servidor de monitor indicando que servidores respondieron al pin/echo y se encuentran disponibles.

5. Forzamos la caída de la sesión del server\_2 y validamos que aleatoriamente el monitor realizó la asignación de un nuevo líder, que en este caso es el server\_1 (pagos1):

```
The leader is server_1
127.0.0.1 -- [11/Sep/2021 23:46:44] "POST //send_payment HTTP/1.1" 200 -
I AM ALIVE from server 1
I AM OFF from server 2
I AM OFF from server 3
The leader is server_1

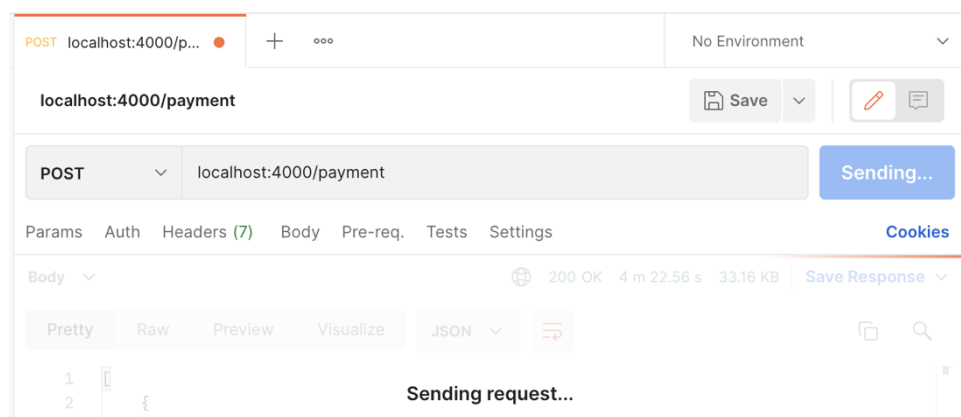
I AM ALIVE from server 1
I AM OFF from server 2
I AM OFF from server 3
The leader is server_1

127.0.0.1 -- [11/Sep/2021 23:46:45] "POST //send_payment HTTP/1.1" 200 -
I AM ALIVE from server 1
I AM OFF from server 2
I AM OFF from server 3
The leader is server_1

I AM ALIVE from server 1
I AM OFF from server 2
I AM OFF from server 3
The leader is server_1
```

*Figura 10. Caída del servidor 2 y respectiva validación en el componente de monitoreo.*

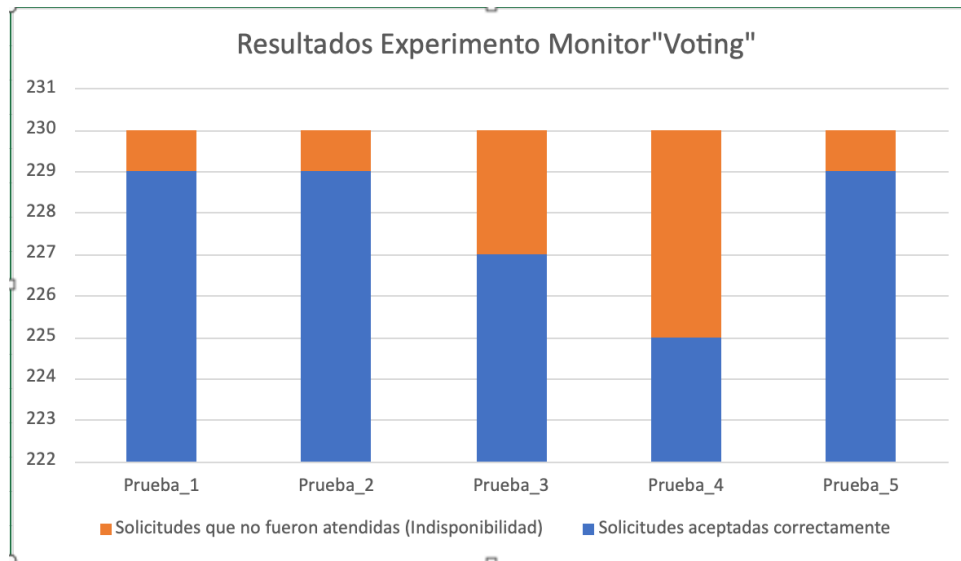
6. Para concluir la prueba, realizamos una solicitud via Postman al servidor de pacientes que se encuentra desplegado en el puerto 4000, invocando el servicio de pagos: Este queda enviando request hasta que termine de procesar las 230.



*Figura 11. Solicitud final para accionar el experimento para un cliente asociado*

## Resultados:

Cada una de las cinco (5) pruebas se realizaron de forma exitosa, es decir, el monitor cumplió su función de estar validando la disponibilidad de los servidores de pagos y realizar reasignación de líder en los momentos en los que se presentó la falla que fue forzada en cada uno de los escenarios de prueba. Esto corresponde con una disponibilidad media de 99,043%.



*Figura 12. Resultados de experimentación asociadas a los 5 grupos de prueba generados.*

En la figura no.13 se pueden evidenciar los resultados porcentuales del experimento , estos fueron obtenidos del total de solicitudes iniciales mandadas (230) y del numero de solicitudes obtenidas al final por los servidores de pagos, cabe aclarar que estas fueron almacenadas en base de datos.

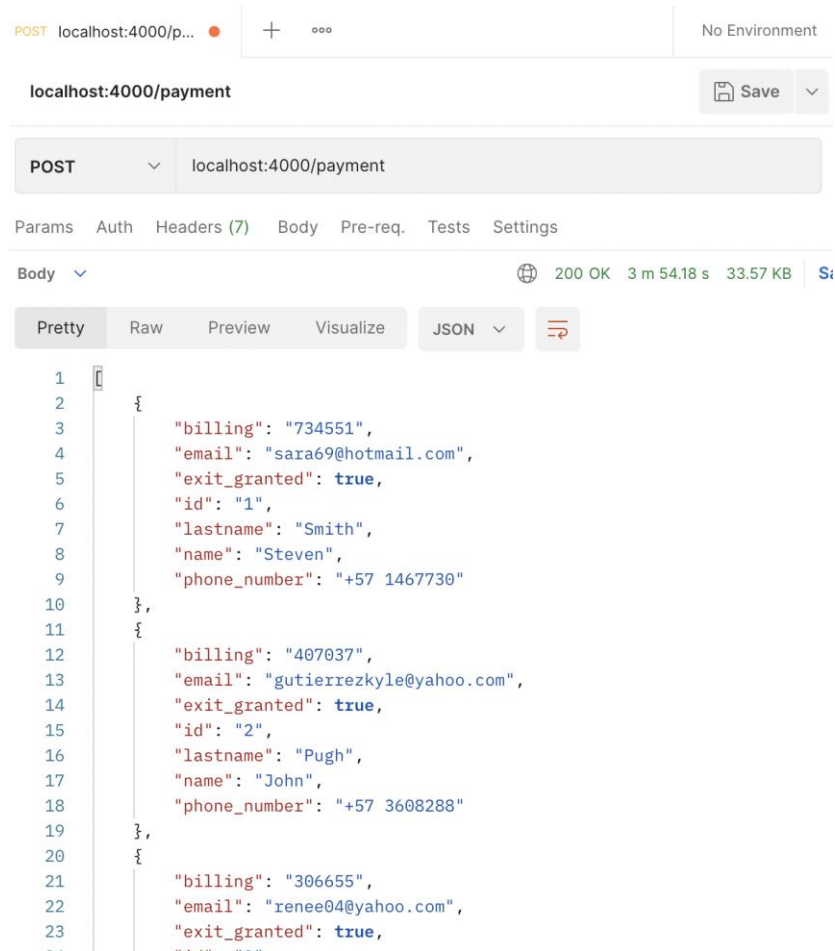
Prueba	Enviados	Exitosos	Disponibilidad
Prueba_1	230	229	99,565%
Prueba_2	230	229	99,565%
Prueba_3	230	227	98,696%
Prueba_4	230	225	97,826%
Prueba_5	230	229	99,565%
<b>Total Requests</b>	<b>1150</b>	<b>1139</b>	<b>99,043%</b>

*Figura 13. Resultados porcentuales de los 5 casos de experimentación.*



Algunas de las validaciones realizadas con postman fueron:

1. Validación de resultados en Postman: Cada objeto JSON con un 'true' en 'exit\_granted'.



*Figura 14. Resultado de las validaciones de pago exitosas informando que un paciente puede salir, debido al éxito en la transacción.*

2. Se evidencia entonces la generación de datos por parte del servicio de pagos de los pacientes que ejecutaron sus pagos de manera exitosa:

New Database Open Database Write Changes Revert Changes

Database Structure Browse Data Edit Pragmas Execute SQL

Table: payments

	id	name	lastname	email	phone_number	billing	exit_granted
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
206	206	Mr.	Baker	thomasmith@hotmail.com	+57 1195422	280948	1
207	207	Danielle	Blevins	smiddleton@church.com	+57 1897723	133561	1
208	208	Christine	Gonzales	briananderson@lane.com	+57 4945180	790895	1
209	209	Juan	Jensen	kristi16@gmail.com	+57 4310053	495627	1
210	210	Christopher	Nelson	joyce11@hotmail.com	+57 4054953	795609	1
211	211	Pedro	Moyer	alex27@hotmail.com	+57 5534425	466765	1
212	212	Jerry	Morris	aaronfisher@clark.info	+57 5083559	274954	1
213	213	Scott	Montgomery	steven00@hotmail.com	+57 8373819	348981	1
214	214	Terri	Graham	henrywang@dennis.com	+57 8441155	627341	1
215	215	Kenneth	Davis	cbarrett@yahoo.com	+57 5794348	897160	1
216	216	Pamela	Horn	brianboyd@gmail.com	+57 4667472	788108	1
217	217	Jessica	Guzman	matthew51@hernandez.com	+57 5022637	217788	1
218	218	David	Brown	dayjoshua@sanders.com	+57 3437578	273588	1
219	219	John	Williams	edecker@cervantes.info	+57 1998498	276069	1
220	220	Adrienne	Lewis	emma10@smith.com	+57 2791316	295259	1
221	221	Beth	Nash	gregorymorris@wheeler-smith.com	+57 6789859	229874	1
222	222	Melissa	Bautista	wwatts@hotmail.com	+57 2656734	546651	1
223	223	Kristin	Gomez	vegadonna@bradley.net	+57 1722438	680349	1
224	224	Jeremy	Mack	kevuncuningham@rogers.biz	+57 3055554	465172	1
225	225	Robert	Grant	cervantesjames@gmail.com	+57 3072763	300038	1
226	226	Jessica	Johnson	qbrooks@litle.org	+57 7435814	102257	1
227	227	Angela	Roman	jfrench@miller.info	+57 3127460	736931	1
228	228	Russell	Velasquez	mejacourtney@hotmail.com	+57 2738228	245169	1
229	229	Sonia	Drake	ujames@gmail.com	+57 5464615	122845	1

206 - 229 of 229 Go to: 1

*Figura 14. Persistencia de la información de pagos exitosos almacenada en base de datos.*

#### Decisiones de arquitectura que favorecieron el experimento:

- La implementación de un monitor que permita continuar con el servicio correctamente aun estando uno o varios servidores abajo.
- La implementación de las colas permitió que los resultados. evita las colisiones entre las solicitudes enviadas, mantiene el orden de llegada permitiendo que los pagos se realicen de acuerdo a la fecha de envío y evita que el servidor líder colapse con el envío de solicitudes en paralelo.
- Dejar un tiempo configurado de ping eco bajo permite la validación constante de la vida de los servidores para saber o descubrir mucho más rápido en que momento estos dejan de estar fuera de línea.

### **Cambios que podría tener el diseño:**

- Agregar al sistema de colas implementado un número de reintento cuando encuentre un servidor disponible para que se cumpla solicitud y evitar un error al cliente.
- Al implementar el sistema real, es decir, implementación a producción , implementar un patrón de maestro esclavo en el sistema de almacenamiento para tener mayor respaldo a la disponibilidad.

### **Evidencias:**

Las evidencias de este experimento se encuentran disponibles para consulta y revisión en los siguientes lugares:

- El código fuente de este experimento se encuentra en el repositorio del equipo.
- Este informe contiene un resumen del experimento y el respectivo análisis de resultados.
- Un archivo en Excel “Datos de Prueba Experimento” que contiene los datos de los resultados y las gráficas que fueron incluidas en este informe junto con sus datos base. Este documento se encuentra dispuesto en la Wiki del proyecto:
- Un video de la presentación del experimento realizado con los miembros del equipo realizando las respectivas explicaciones, para ver este video seguir el link:
- [https://uniandes-edu-co.zoom.us/rec/share/XYGrDpo7a-tlqHpQE0IIVEi\\_1u7zs2xGciUrQeeiMCBj9nraH1L7VRWDihsq--h0.xT5gsMicLdfEF1JL?startTime=1631457212000](https://uniandes-edu-co.zoom.us/rec/share/XYGrDpo7a-tlqHpQE0IIVEi_1u7zs2xGciUrQeeiMCBj9nraH1L7VRWDihsq--h0.xT5gsMicLdfEF1JL?startTime=1631457212000)

### **Conclusiones:**

- Con el experimento se logra evidenciar la importancia de implementación de un monitor que verifique la prestación correcta de los servicios correspondientes a los servidores a cargo, esto permite disminuir el impacto negativo en el servicio al cliente y, por consiguiente, tener una buena imagen y confianza.
- Este tipo de pruebas pueden realizar una comprobación muy efectiva y temprana sobre nuestra arquitectura elegida para tener un feedback rápido que nos permita verificar si efectivamente el trabajo que se realizó en el proceso de diseño se acopla a los objetivos del negocio y además cumple con lo esperado.
- Realizar este tipo de pruebas sobre diseño en etapas tempranas permite tener información del comportamiento de nuestro diseño de manera previa a ponerlo en producción o en un ambiente de pruebas grande que nos haga incurrir en fallas que luego desencadenen nos lleven a temas monetarios.
- Aunque hay muchos puntos únicos de posibles fallas de disponibilidad y difíciles de controlar como por ejemplo una falla en la base de datos, realizar este tipo de pruebas a la arquitectura, permite tomar acciones que minimicen el impacto y

mejoren el desempeño de la arquitectura disminuyendo posibles fallas en los demás puntos en los que si se pueden tomar acciones.