FACULTAD DE INGENIERIA
Universidad de Buenos Aires

# Trabajo Práctico 1

**Grupo**

[75.73/TB034] Arquitectura del Software
Segundo cuatrimestre de 2025


| Alumno | Padrón | Email |
|---|---|---|
| CASTRO MARTINEZ, Jose Ignacio | 106957 | jcastrom@fi.uba.ar |
| DEALBERA, Pablo Andres | 106858 | pdealbera@fi.uba.ar |
| FIGUEROA RODRIGUEZ, Andrea Emperatriz | 110450 | afigueroa@fi.uba.ar |
| RICALDI REBATA, Brayan Alexander | 103344 | bricaldi@fi.uba.ar |

# Índice

# 1. Introducción

## 1.1. Contexto (startup arVault).

## 1.2. Objetivos del TP.

## 1.3. Alcance del análisis.

# 2. QAs identificados

## 2.1. QAs clave y justificación.

### 2.1.1. Disponibilidad

Al ser un servicio de exchange de monedas, asumimos que es un servicio que se utiliza durante todos los dias habiles de la semana en horario cambiario. Por lo tanto, es importante que el servicio se encuentre disponible durante esos horarios para no perder clientes.

Tambien asumimos que los reclamos pueden venir que los clientes no puedan acceder al servicio o que no puedan realizar operaciones.

### 2.1.2. Escalabilidad (Elasticidad)

El servicio debe poder escalar para poder atender la demanda de los clientes, que asumimos debe tener picos de demanda importantes en ciertos momentos del dia (por ejemplo, en la apertura y cierre del horario cambiario) y momentos de baja o nula demanda, por lo que tambien seria deseable que el servicio pueda escalar a cero para evitar costos innecesarios.

### 2.1.3. Visibilidad

Es importante poder monitorear el estado del sistema para detectar posibles problemas y poder solucionarlos antes de que afecten a los clientes. Ademas, es importante poder monitorear el rendimiento del sistema para poder escalarlo adecuadamente y evitar problemas de disponibilidad.

### 2.1.4. Seguridad

Es fundamental que el sistema sea seguro para evitar posibles ataques que puedan comprometer la integridad del sistema, la privacidad de los datos de los clientes o perdida/robo de dinero. Incluso pensando que debemos tambien tener en cuenta marcos regulatorios sobre el manejo de datos personales y financieros. Tambien entendemos del enunciado que es importante la reputacion del sistema, y esto podria verse muy dañado en caso de que haya una brecha de seguridad.

## 2.2. Caso base y como cumplen estos QAs claves

### 2.2.1. Disponibilidad

### 2.2.2. Escalabilidad (Elasticidad)

Actualmente hay un Nginx que actua como reverse proxy y potencialmente balanceador de carga, pero en este momento solo tiene configurado una sola instancia de la app de Node.js. De todas formas, notamos varios problemas con esto. En principio, la app es stateful porque guarda el estado en memoria y guarda cada tantos segundos el estado de la memoria en distintos archivos json en la carpeta `state/`. Esto hace que no se pueda escalar horizontalmente la app sin perder el estado, ya que cada instancia tendria su propio estado en memoria y no habria forma de sincronizarlos.

#### 2.2.3. Visibilidad

Actualmente hay un contenedor de Graphite y otro de Grafana para monitorear el sistema, y tienen algunas metricas en un dashboard creado por la catedra que permite visualizar algunas metricas como Scenarios launched, Request state, Response time y Resources. Faltarian metricas mas especificas del negocio como por ejemplo, volumen de transacciones por moneda, cantidad de clientes activos, etc.

#### 2.2.4. Seguridad

# 3. Arquitectura base

## 3.1. Diagrama C&C inicial.