Problema de venta de boletos

INFO 288- Sist. Distribuidos

Instituto de Informática, Universidad Austral de Chile.

Integrantes: Sebastián Luarte
Matias Martinez
Tomas Banegas
Tomas Manriquez
Sebastian Paredes
Javier Mansilla

Fecha: 07 de Abril de 2023

Introducción

El objetivo de este trabajo es presentar un sistema distribuido que maneja la alta demanda de solicitudes de una página web que se dedica a la venta de boletos, similar a la plataforma ticketmaster. Para esto, se establecieron supuestos previos basados en las necesidades específicas del cliente. Se decidió que el sistema debe ser escalable, para permitir un aumento en la demanda, y tener una alta disponibilidad, para garantizar que los clientes puedan comprar boletos en cualquier momento.

En este informe, se presentarán en detalle los elementos clave del diseño del sistema distribuido, incluyendo los diagramas de arquitectura,despliegue y fundamental, además de las especificaciones de hardware y software donde se pretende desplegar para realizar los procesos de prueba y validación. Además, se muestra el modelamiento de la base de datos, junto con todas las librerías y herramientas necesarias para el desarrollo de este Software.

Supuestos

Los supuestos de este sistema se presentan mediante preguntas a los clientes, junto con sus respectivas respuestas, estas son las siguientes:

1. **¿Cuántos servidores tiene actualmente en su empresa y qué sistemas operativos utilizan?**

   R: Contamos con 2 clusters en los cuales mantenemos un total de 4 servidores con ubuntu 22.04 lts, de ser necesario también se pueden usar estos servidor con windows.

2. **¿Cuál es el nivel de tráfico usual que recibe su sitio web? ¿Ha notado algún pico en la demanda en algún momento en particular?**

   R: El tráfico usual del sitio es de menos de 1000 personas en un día normal, con un peak de hasta 4500 personas cuando existen conciertos de gran envergadura durante el día, habiendo momentos que llegan a las 2000 personas simultáneas.

3. **¿Cuántos eventos y conciertos están disponibles simultáneamente para reserva?**

   R: Cómo máximo existen 6 conciertos disponibles para reserva en cualquier momento dado.

4. **¿Con qué frecuencia necesita realizar actualizaciones o mantenimiento en sus servidores o en su sitio web?**

   R: Necesitamos realizar actualizaciones y mantenimiento en nuestros servidores y sitio web con bastante frecuencia para garantizar su funcionamiento correcto y eficiente. Estamos en constante monitoreo de cualquier problema o inconveniente que pueda surgir y solucionarlo de manera oportuna. De forma estimada, ocurre un

concierto importante de gran tráfico por mes por lo que se espera realizar tareas de mantenimiento antes y después de este concierto.

5. **¿Qué nivel de seguridad necesita para proteger sus datos y sistemas? ¿Tiene alguna política de seguridad en su empresa?**

   R: La seguridad de nuestros datos y sistemas es una prioridad clave en nuestra empresa. Contamos con políticas de seguridad y medidas de protección para garantizar que la información personal y financiera de nuestros clientes esté segura en todo momento.

6. **¿Cuál es su presupuesto para el desarrollo y mantenimiento de su sitio web de venta de boletos y cuánto está dispuesto a invertir en tecnología?**

   R: El presupuesto disponible cuenta con un cupo limitado por lo que el uso de cluster y la eficiencia que presentan debe ser optimizada para obtener el máximo beneficio por los recursos que otorgan.

7. **¿Requiere una solución de alta disponibilidad o redundancia en sus servidores?**

   R: Dado que somos un servicio web de boletería que funciona las 24 horas del día, los 7 días de la semana, requerimos una solución de alta disponibilidad y redundancia en nuestros servidores para garantizar que nuestros clientes puedan acceder al sitio web y realizar compras en cualquier momento, sin interrupciones. (mencionar infraestructura)

8. **¿Cómo se asegura de que su sitio web de venta de boletos esté diseñado para ser escalable y capaz de manejar grandes volúmenes de tráfico durante eventos de alto perfil?**

   R: Utilizando tecnologías de escalado como un orquestador de contenedores que permita replicar componentes de ser necesarios.

   Utilizando un sistema de colas replicable que permita a 1000 usuarios por cola de forma de poder agregar más colas para las fechas en donde exista un alto tráfico para los conciertos pertinentes.

9. **¿Está dispuesto a considerar soluciones de software de código abierto para reducir los costos de licencia y mantenimiento de software?**

   R: Sí, cualquier manera de abaratar costos sería ideal, nuestro presupuesto es limitado.

10. **¿Qué tipo de soluciones de alojamiento está utilizando actualmente y qué tan satisfecho está con su rendimiento y seguridad?**

R: Disponemos con alojamiento en servidores locales , de capacidad limitada por lo que el rendimiento no es el óptimo y nos gustaría obtener soluciones sobre cómo optimizarlo.

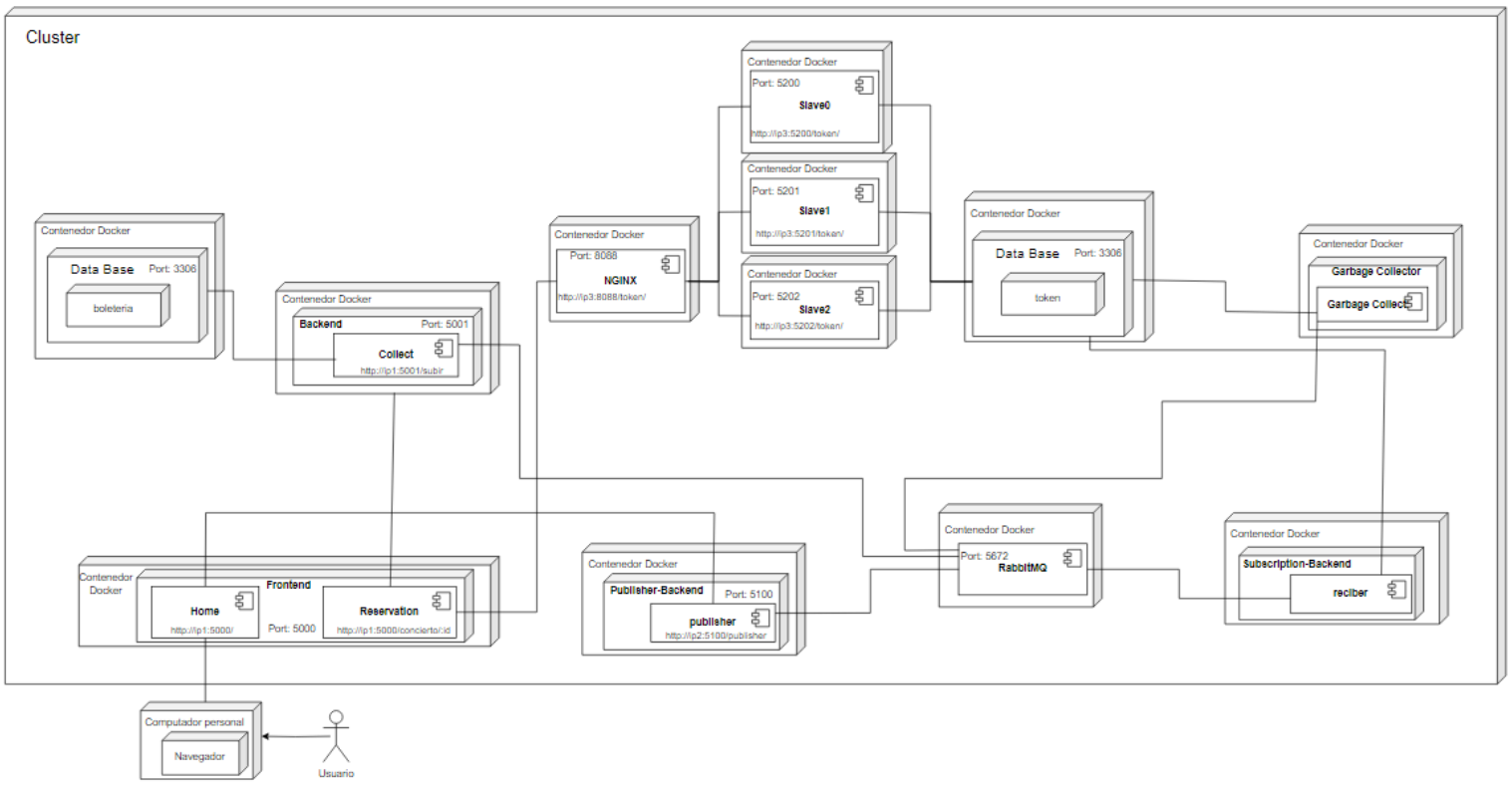
Modelo Fundamental

En base a los supuestos descritos anteriormente, se definieron las siguientes políticas del sistema que describe el modelo fundamental:

- Capacidad de clientes simultáneos en un concierto (No encolados): 1000
- Capacidad de la cola para cada conciertos: 5000
- Capacidad total de clientes en página principal: 2500
- Cantidad máxima de conciertos concurrentes: 6
- Tiempo de espera máximo para que el cliente ingrese sus datos y haga el pago: 5 minutos - 300 segundos
- Tiempo para almacenar información del cliente en la base de datos: 300ms
- Tiempo de espera máximo en ir desde la página principal hacia administrador de colas (Publisher Backend-Colas): 400ms
- Tiempo de espera máximo en que se guarde la información en la base de datos, luego de que el usuario complete la reserva: 400ms

Modelo Físico y diagrama de despliegue UML

En este diagrama se muestran los distintos componentes distribuidos en un cluster, cada una de estas componentes se encuentran contenidas en un contenedor de Docker para un despliegue sencillo a modo de prueba y que no presente problemas de incompatibilidad.



La descripción de cada componente de nuestra solución son las siguientes:

Frontend:

- Home: Contiene páginas con todos los conciertos listados, es la página principal con la que interactúa el cliente. Al seleccionar uno, se asigna una cookie llamada “token” con un string aleatorio, este mismo string es enviado a Publisher-Backend.
  Está hecha con React
- Reservation: Tiene distintas rutas dependiendo del concierto que se seleccionó al principio. cada 5 segundos a NGINX, si el token almacenado en la cookie se encuentra en la base de datos “token”, si no se encuentra, se muestra una pantalla de carga. En caso de que el token se encuentre, se mostrará un formulario de inscripción para el cliente, junto con información proveniente de Backend, la información del formulario se envía nuevamente a Backend.
  Está hecha con React

NGINX: Actúa como balanceador de carga, distribuye las peticiones provenientes de Reservation hacia los distintos slaves

Slave: Consulta si un string se encuentra almacenado en la base de datos “token”. Retorna “SI” en caso de que el string se encuentre y “No” en caso contrario.

Publisher Backend:

- Publisher: Toma el token enviado desde Home y las redirige a la cola correspondiente del concierto seleccionado.

RabbitMq: Cola de mensajes, maneja colas individuales para cada concierto, regula el paso hacia Subscription-Backend y su finalidad es descongestionar el sistema.

Subscription Backend:

- Reciber: Analiza el token provenientes de las múltiples colas de concierto y sube el token a la base de datos “token”.

Backend:

- Collect: Toma el formulario rellenado en Reservation, lo formatea y envía hacia la base de datos Boletería. Además, entrega la información almacenada en la base de datos, sobre los conciertos y su respectivo lugar, además de los asientos disponibles.

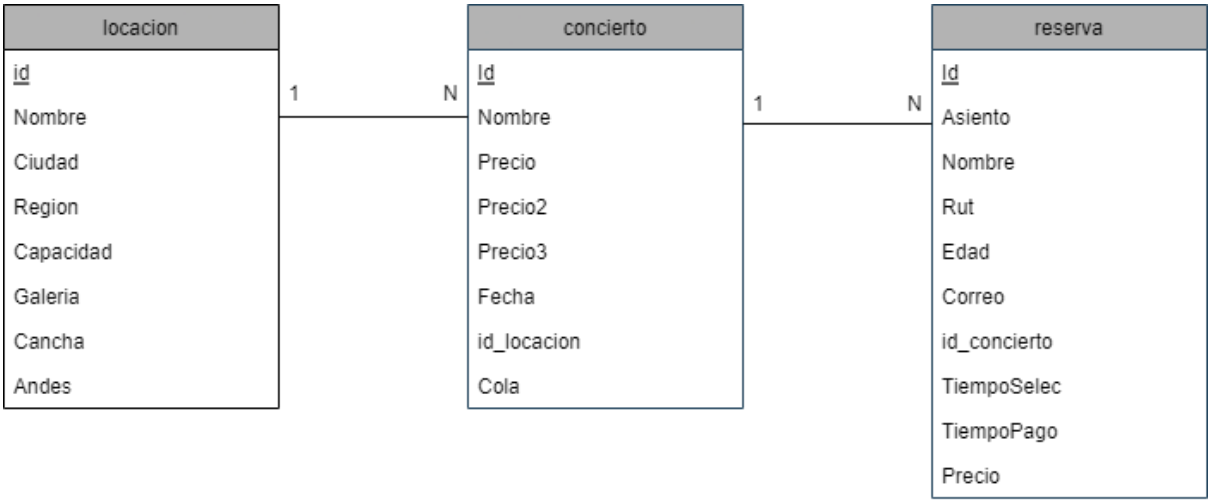
Database:

- Boletería: Nuestro sistema de persistencia de datos, contiene datos de conciertos, reservas y clientes. Está compuesta por las tablas: Conciertos, Reserva y Locación. (Utilizamos MariaDB)
- Token: La base de datos token, está compuesta por dos tablas, la primera llamada token, que guardara las cookies del navegador del usuario, y las borrará después del tiempo de expiración definido (5 minutos), y la tabla all_tokens que guardará todas las cookie. (Utilizamos MariaDB)

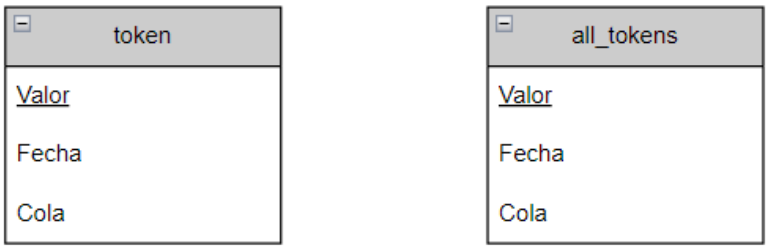
Garbage Collector: Se encarga de limpiar la base de datos Token revisando que no existan tokens expirados debido a un cierre inesperado por parte del cliente.

Base de datos

El sistema contiene dos bases de datos llamadas “boletería” y “token”. El modelo relacional que representa la primera base de datos mencionada se encuentra a continuación:



El modelo relacional que representa la base de datos “token” se presenta en la siguiente figura:



En base a los modelos anteriores, se realizaron los siguientes diccionarios de datos, con sus respectivos nombres, tamaños, tipos de datos y descripciones.

Tabla Reserva

Nombre	Tamaño (bytes)	Tipo de dato	Descripción
id	4	Numérico	Id de la reserva.
Asiento	4	Numérico	Número de asiento que reservó el cliente.
Nombre	30	Carácter	Nombre del cliente.
Rut	30	Carácter	Rut del cliente.
Edad	4	Numérico	Edad del cliente.
Correo	30	Carácter	Correo del cliente.
#id_concierto	4	Numérico	ID del concierto
TiempoSelec	8	Fecha	Fecha en la cual se apreto el boton de pago
TiempoPago	8	Fecha	Fecha en la cual se realizó el pago.
Precio	4	Numérico	Precio total del concierto

Tabla concierto

Nombre	Tamaño	Tipo de dato	Descripción
id	4	Numérico	Id del concierto.
Nombre	30	Carácter	Nombre del concierto.
Precio	4	Numérico	Precio de sección galería
Precio2	4	Numérico	Precio de sección cancha
Precio3	4	Numérico	Precio de sección andes
Fecha	8	Fecha	Fecha del concierto.
#id_locacion	4	Numérico	ID de la localización.
Cola	30	Carácter	Cola a la que pertenece.

Tabla locación

Nombre	Tamaño (bytes)	Tipo de dato	Descripción
id	4	Numérico	Id de locación.
Nombre	30	Carácter	Nombre del lugar donde se hará el concierto.
Ciudad	30	Carácter	Nombre de la ciudad donde se hará el concierto.
Región	30	Carácter	Nombre de la región donde se hará el concierto.
Capacidad	4	Numérico	Capacidad del lugar donde se hará el concierto.
Galeria	4	Numérico	Capacidad de personas que pueden estar en sección galería.
Cancha	4	Numérico	Capacidad de personas que pueden estar en sección cancha.
Andes	4	Numérico	Capacidad de personas que pueden estar en sección andes.

Tabla token (Pertenece a la base de datos token, almacena los tokens y los borra luego de la fecha de expiración)

Nombre	Tamaño	Tipo de dato	Descripción
valor	30	Carácter	Valor de la cookie que se almacenará.
Fecha	4	Numérico	Tiempo de expiración.
Cola	30	Carácter	Cola a la que pertenece.

Tabla all_tokens (Pertenece a la base de datos token, guardará todos los tokens)

Nombre	Tamaño	Tipo de dato	Descripción
valor	30	Carácter	Valor de la cookie que se almacenará.
Fecha	4	Numérico	Tiempo de expiración.
Cola	30	Carácter	Cola a la que pertenece.

Recursos a utilizar

En base a lo anteriormente descrito, las características del dispositivo que se utilizará para probar el despliegue del proyecto es el siguiente:

Especificaciones Cluster:

- RAM: 16 GB
- Procesador: Intel core i5-11400-h 2.7 GHz 6 cores
- Almacenamiento: 512 GB

Espacio requerido por los contenedores a ocupar:

Contenedor 1:

-cluster1-backend: 68.29 MB

-cluster1-frontend: 1.15 GB

-db-boleteria: 403.33 MB

Contenedor 2:

-cluster2-publisher-backend: 307.72 MB

-cluster2-suscription-backend: 320.23 MB

-cluster2- rabbitmq: 246.85 MB

Contenedor 3:

-cluster3-db-token: 403.33 MB

-cluster3-slave0: 68.28 MB

-cluster3-slave1: 68.28 MB

-cluster3-slave2: 68.28 MB

-cluster3-garbagecollector: 61.55 MB

-cluster3-nginx : 142.15 MB

Contenedor 1: En este contenedor, estará la página principal con su respectivo backend y base de datos.

Contenedor 2: En este equipo backend de publicador, y backend de publisher.

Contenedor 3: En esta máquina el recolector de basura, la base de datos persistente, el backend de los tokens y el sistema de colas.

Cabe destacar que estos contenedores son ejecutadas en un solo cluster/computador, correspondiente a un integrante de este proyecto, que se utilizará con el fin de lograr desplegar la página. Este proyecto no se desplegará en servidores reales, ya que es con fines académicos.

Software a ocupar

Se establecieron los siguientes softwares libres para el desarrollo de la página, enfocados en las fases de desarrollo y testeo del software.

- Visual Studio code, editor de texto que se utilizará por familiaridad.
    - Ventajas: Gratuito, familiaridad por parte de los programadores, compatible.
    - Desventajas: Un poco lento cuando se tienen muchos archivos (proyectos grandes).
- Postman, se utilizará para realizar peticiones y pruebas a nuestro sistema.
    - Ventajas: Gratuito, familiaridad por parte de los programadores además de presentar una interfaz simple. Gran utilidad para probar la API del sistema.
    - Desventajas: Colaboración con usuarios limitada (hasta 3) y llamadas limitadas (1000 al mes).
- Jmeter, lo utilizaremos para realizar pruebas de carga a nuestro sistema.
    - Ventajas: Gratuito
    - Desventajas: Puede tener un alto consumo de memoria, no permite visualizar los resultados de forma clara, alta curva de aprendizaje al principio además los programadores no tienen familiaridad con la herramienta.

Librerías y herramientas

Se establecieron las siguientes librerías y herramientas que el equipo de desarrollo necesitará para realizar el software.

- Nginx versión 1.22.1, balanceador de carga para distribuir las peticiones.
- RabbitMQ versión 3.11.16, para encolar a los usuarios antes de entrar a un concierto.
- Python versión 3.11.2, lenguaje para implementar programación con las siguientes librerías.
    - requests
    - json
    - Flask
    - pika
    - install
    - mariadb
- React versión 18.2.0, framework para el frontend, escogido por familiaridad.
- TailwindCss versión 3.2.7, framework de css para dar estilo a la página web.
- Mariadb versión 10.11.2, para la base de datos.
- Docker 23.0.2, facilitara levantar la aplicación en los servidores.
- Flask 2.2.4, para la API REST de la aplicación.