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.

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 5 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.

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.

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.

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.

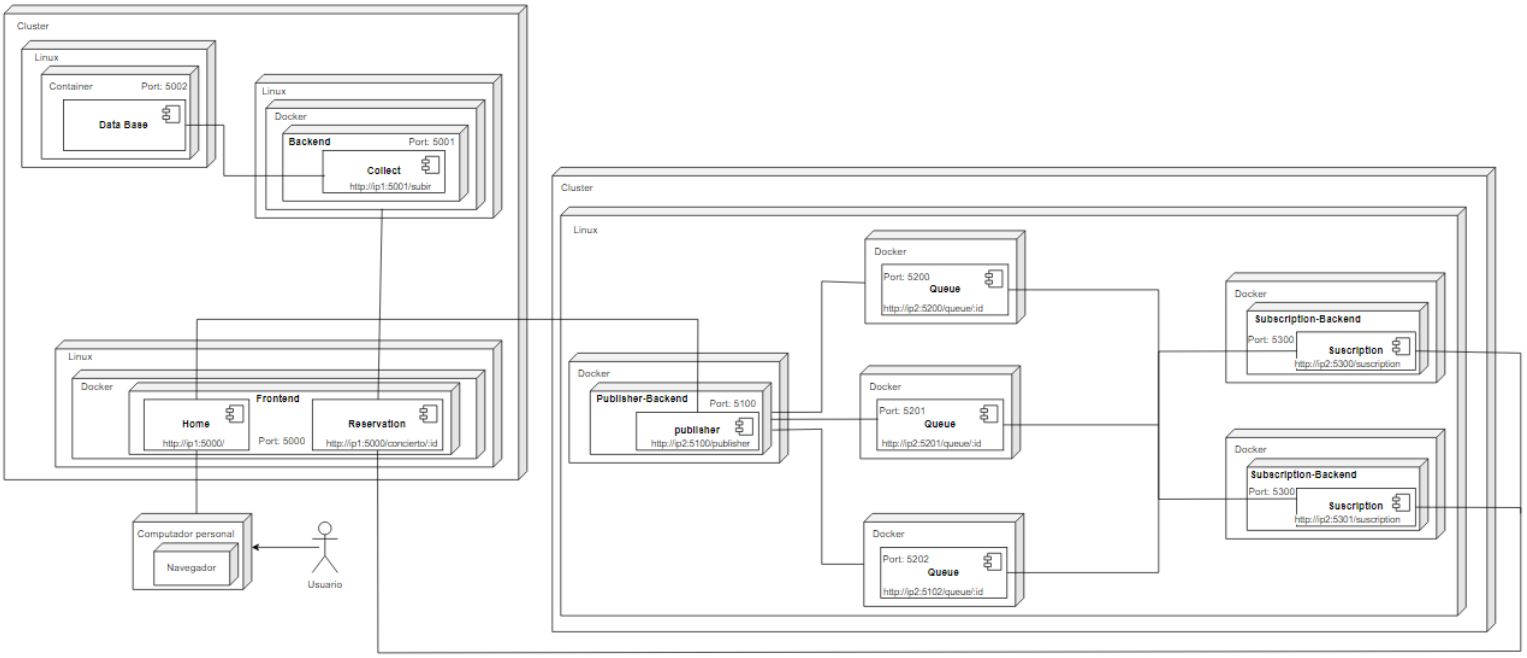
Modelo Fundamental

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 replicación de una cola para un concierto: 3
- Replicado a un uso de: 60%
- Cantidad máxima de conciertos concurrentes: 5
- Tiempo de espera máximo para que el cliente ingrese sus datos y haga el pago: 3 minutos - 180 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 entre dos clusters y cómo se comunican entre ellos , cada una de estas componentes se encuentran contenidas en un contenedor de Docker para un despliegue sencillo 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, Manda requests a Publisher.
  Está hecha con React
- Reservation: Contiene un formulario de inscripción para el cliente, tiene distintas rutas dependiendo del concierto que se seleccionó al principio, envía requests a

Collect.
Está hecha con React

Publisher Backend:

- Publisher: Toma y analiza las request desde Home y las redirige a la cola correspondiente del concierto seleccionado.

Queue: Actúa como cola individual para un concierto, regula el paso hacia Subscription-Backend y su finalidad es descongestionar el sistema.
Está hecho con NATs y posee réplicas dependiendo del número de conciertos disponibles para reservación para ello se pretende utilizar kubernetes y docker.

Subscription Backend:

- Subscription:Analiza las requests provenientes de las múltiples colas de concierto y redirige a los usuarios hacia la ruta correspondiente en Reservation.
  Este componente posee réplicas a modo de procesar mayor cantidad de peticiones, para ello se pretende utilizar kubernetes y docker.

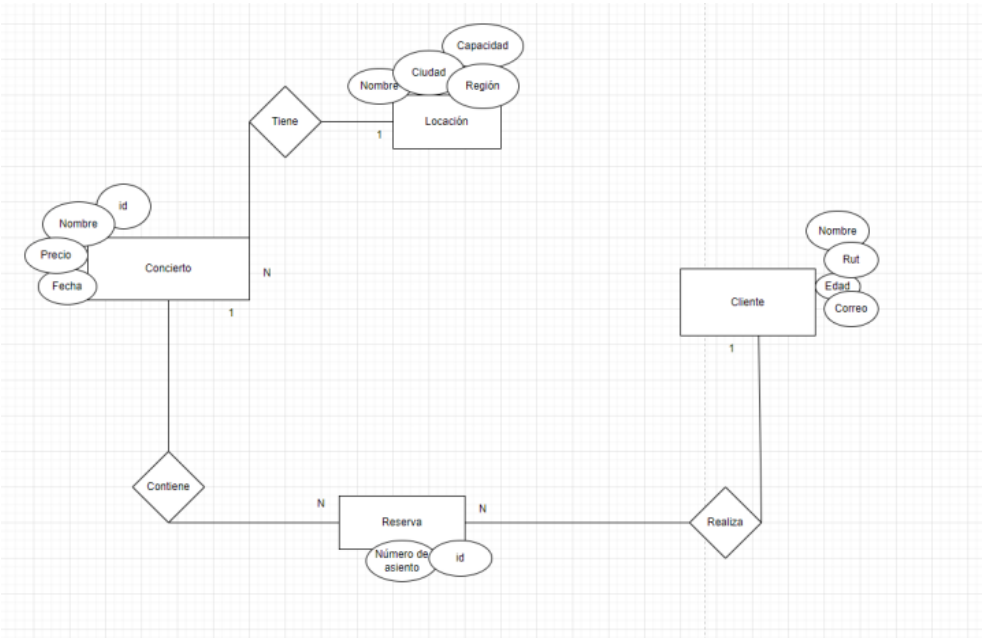
Backend:

- Collect: Toma el formulario rellenado en Reservation, lo formatea y envía hacia la base de datos Data Base.

Database: Nuestro sistema de persistencia de datos, contiene datos de conciertos, reservas y clientes. Utilizamos MariaDB.

Base de datos

En base a lo requerido, creamos el siguiente diagrama UML, que contiene principalmente 4 componentes; Cliente, Reserva, Concierto y Locación.



En base al diagrama UML anterior, se hizo el siguiente diccionario de datos, con sus respectivos nombres, tamaños, tipos de dato y descripciones.

Nombre	Tamaño	Tipo de dato	Descripción
client_name	64	Carácter	Nombre del cliente
rut	9	Numérico	Rut del cliente
age	2	Numérico	Edad del cliente
email	30	Carácter	Correo del cliente
reserve_id	4	Numérico	ID destinado a la reserva hecha por el cliente
seat_number	4	Numérico	Número de asiento que reservó el cliente
concert_id	4	Numérico	ID del concierto
concert_name	30	Carácter	Nombre del Concierto
price	6	Numérico	Precio del concierto
date	8	Fecha	Fecha del concierto
venue	30	Carácter	Nombre del lugar donde se va a realizar el concierto
capacity	4	Numérico	Capacidad del lugar donde se va a realizar el concierto
región	30	Carácter	Región del local
city	30	Carácter	Ciudad donde esta el local

Recursos a utilizar

En base a lo anteriormente descrito, las características de los clusters que se utilizaran para el despliegue del proyecto serán los siguientes:

Cluster 1: En este equipo estará la página principal con su respectivo backend y base de datos.

Especificaciones Cluster 1:

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

Cluster 2: En este equipo backend de publicador, las colas y el backend de suscriptor..

Especificaciones Cluster 2:

- RAM: 16 GB
- Procesador: Intel core i5 8300-h 2.3 GHz 4 cores
- Almacenamiento: 1 TB

Cabe destacar que estos cluster son computadores de los integrantes del proyecto, que se utilizaran con el fin de lograr desplegar la página. Este proyecto no se desplegará en servidores reales, ya que es con fines académicos.

Para los requisitos de cada máquina virtual consideramos los requisitos recomendados de Ubuntu Server: 1GB de memoria RAM, 2.5 GB de almacenamiento y 1GHz de CPU. Además del consumo de las librerías, tecnologías y considerando el número de peticiones simultáneas en el peor caso llegamos a las siguientes especificaciones:

Especificación de máquina virtual 1 (Frontend Home/Reservation):

- Procesador: 2 cores.
- Sistema operativo: Ubuntu Server 22.04 lts
- RAM: 4 GB de ram
- Almacenamiento: 2.5 GB del sistema operativo + 1 GB de scripts e imágenes + 1.5 GB de holgura = 5 GB

Especificación de máquina virtual 2 (Backend):

- Procesador: 1 cores.
- Sistema operativo: Ubuntu Server 22.04 lts
- RAM: 4 GB de ram
- Almacenamiento: 2.5 GB del sistema operativo + 1 GB de scripts + 0.5 GB de holgura = 4 GB

Especificación de máquina virtual 3 (Base de datos):

- Procesador: 1 cores.
- Sistema operativo: Ubuntu Server 22.04 lts
- RAM: 2 GB
- Almacenamiento: 20 GB

Especificación de máquina virtual 4 (Backend Publisher, Colas y Backend Subscription):

- Procesador: 2 cores.
- Sistema operativo: Ubuntu Server 22.04 lts
- Ram: 6 GB
- Almacenamiento: 2.5 GB del sistema operativo + 4 GB de scripts junto con sus réplicas + 0.5 GB de holgura = 7 GB

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.
- Nats versión 2.13.1, para encolar a los usuarios antes de entrar a un concierto.
- Python versión 3.11.2, lenguaje para implementar programación.
- 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.
- Kubernetes versión 1.26.0, para la replicación de componentes.
- Docker 23.0.2, facilitara levantar la aplicación en los servidores.
- Flask 2.2.4, para la API REST de la aplicación.