
Universidad de Buenos Aires
Facultad De Ingeniería

1c2021

# 75.61 - Taller de Programación III

Trabajo práctico n.1: Sitio Institucional
Documento de pruebas de carga

`https://github.com/rozanecm/7561-tp1`

INTEGRANTES:
Rozanec, Matías - `mrozanec@fi.uba.ar` - `#97404`

# Índice

# 1. Introducción

En el presente documento se propone mostrar la arquitectura utilizada en el desarrollo del trabajo práctico. Para cumplir con esta tarea se ha optado por diagramas en de vistas $4 + 1$, presentando así 4 vistas características del sistema, agregando además una vista de casos de uso. Sin más, pasamos a la sección de los diagramas.

# 2. Documentación

## 2.1. Vista física

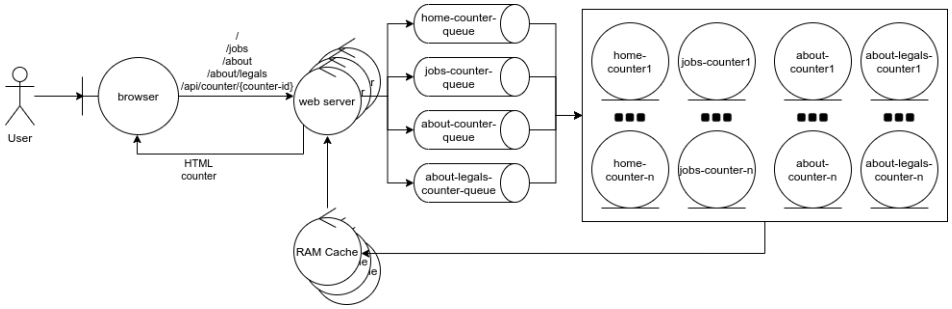


**Figura 2.1:** Diagrama de robustez del sistema implementado

En la vista física se puede ver cómo el cliente interactúa con el Browser que hace de entidad frontera con el sistema. Es por medio de él que se ejecutan las solicitudes a las distintas páginas y contadores.

Notar que la solicitud a un contador figura como un endpoint: esto se debe a que las distintas páginas no devuelven el valor del contador, sino que el mismo se es solicitado por un script en cada una de las páginas. Si bien esto puede parecer como un overhead innecesario, en realidad lo que se busca es liberar la carga del servidor para que pueda responder con la información del sitio lo antes posible, permitiendo que el informe del valor del contador se atrase un poco más, logrando así un sitio con mayor responsiveness. Notar que el uso de templates hizo que el script reciba el contador que deberá pedir de forma programática, por lo que no fue necesario ajustar cada script a mano.

Notar que el web server se comunica además con 4 colas y una cache. Hay una cola por contador, repartiendo así la carga de los distintos contadores. El uso de colas permite que se reponda rápidamente a los clientes, permiento que la actualización del contador se retrase levemente. Como las colas utilizadas vuelven a comunicarse con el mismo web server (pero utilizando métodos de autenticación para evitar que cualquier usuario externo pueda manipularlos), es posible que el la velocidad de atención a una solicitud se vea afectada por la carga de los servidores. Es por ello que ante una carga baja, los contadores se actualizarán de forma cuasi instantánea, mientras que al haber más carga, se tardará proporcionalmente más.

A continuación se puede observar que el almacenamiento de los contadores se realiza en una base de datos con sharding. En la práctica, cada contador cuenta

con 60 entidades en la que se almacenan contadores parciales. Lo que se hace al recibir una solicitud para un sitio es tomar el segudnero del timestamp, y actualizar la entidad según indique dicho segundero. Esto asegura que no habrá más de un segundo de carga sobre una entidad. Si bien Datastore soporta una sola escritura por segundo, según la documentación se pueden recibir pequeños picos de más escrituras. Con el enfoque utilziado, queda asegurado que no habrá picos de larga duración, y además cada pico tendrá un minuto entero para recuperarse y procesar el pico de solicitudes que pudiera llegar a tener.

Por último, la memoria cache se ha implementado en memoria RAM. De esta forma se logra disminuir el impacto de las lecturas, entregando un valor de contador que en la mayoría de las veces estará unos segundos atrasados respecto del valor real. A la hora de tener que actualizar los valores almacenados en la cache, se leen todas las entidades que guardan valores del contador de interés y se suman. Notar que si la cantidad de entidades a utilizar se aumenta demasiado, como se ha considerado en un momento del desarrollo, cada actualización podría terminar resultando sumamente costosa. Leer 60 entidades ha parecido ser un buen punto de equilibrio.

## 2.2. Vista lógica

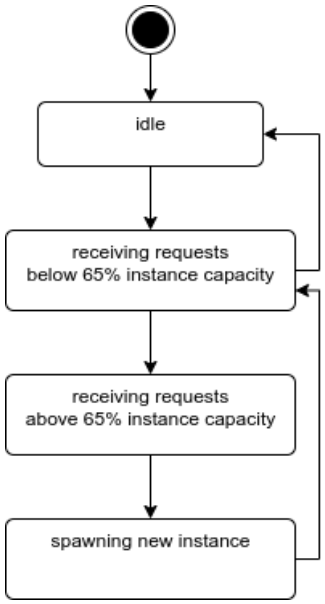


**Figura 2.2:** Diagrama de estados de las instancias deployadas.

Al utilizar AppEngine, se puede configurar el comportamiento de escalamiento de instancias. En este trabajo se ha optado por trabajar con el default propuesto por Google, donde cuando una instancia es iniciada, primero permance en estado idle, lista para recibir requests, luego puede pasar a recibir requests. Si llegara a sobrepasar el 65 % de uso de CPU, entonces se crea una nueva instancia, asegurando de esta forma que el sistema podrá aguantar toda la cantidad necesario de peticiones. Del mismo modo, si llegara a bajar la cantidad de requests a atender, automáticamente se van apagando las instancias que sobren.

## 2.3. Vista de desarrollo

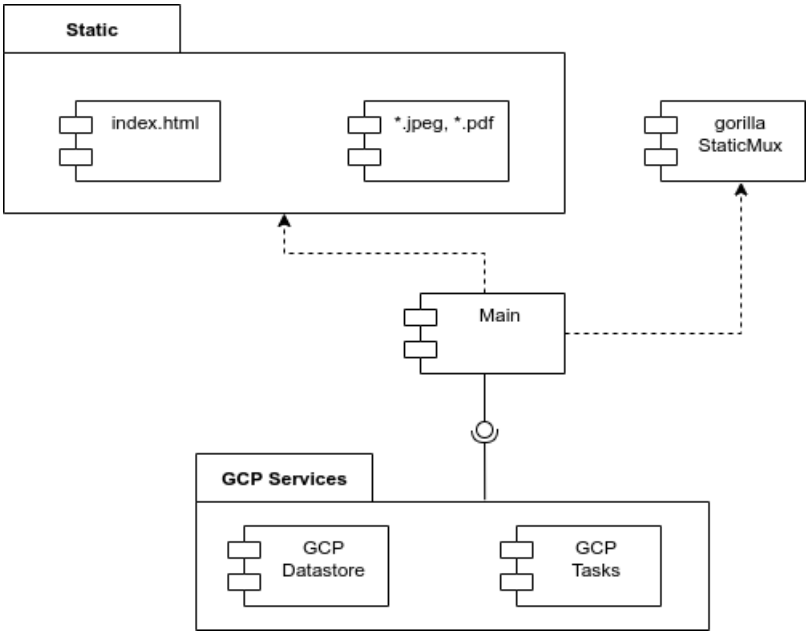


**Figura 2.3:** Diagrama de paquetes.

En este diagrama se puede visualizar que el sistema programado consta básicamente de un paquete principal o main donde se definen los distintos endpoints para servir los recursos estáticos. Las distintas páginas a servir se han modelado a partir de un mismo template de html, que muestra los mismos recursos estáticos, logrando así más uniformidad en las pruebas de carga. Para poder trabajar con templates html, se ha incluido el paquete `gorilla/staticMux`. Como la aplicación ha sido desarrollada en Go, no ha sido necesario incluir bibliotecas externas para implementar un web server.

Finalmente, se han utilizado servicios externos de Google Cloud Computing, a saber¿Datastore para el almacenamiento de datos y Tasks para las colas de tareas.

## 2.4. Vista de proceso

En este apartado se muestran diagramas de secuencia para el acceso a cache, tanto para el caso en que la cache se encuentra actualizada como para cuando la misma se encuentra con datos inválidos.

En ambos casos se puede ver cómo la consulta a una página determinada es independiente de la consulta del valor del contador de dicha página, tal como se ha expuesto en la Sección 2.1.

En la Figura 2.4 se puede ver cómo el contador se actualiza de forma asíncrona, mientras que en la 2.5 se puede ver la consulta adicional a la base de datos en caso que el dato de la cache haya expirado.

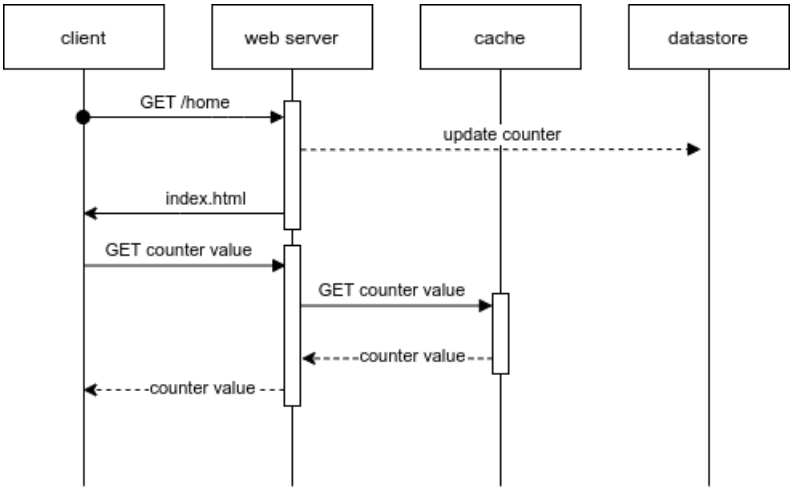


**Figura 2.4:** Diagrama de secuencia de una consulta a cache con datos válidos.

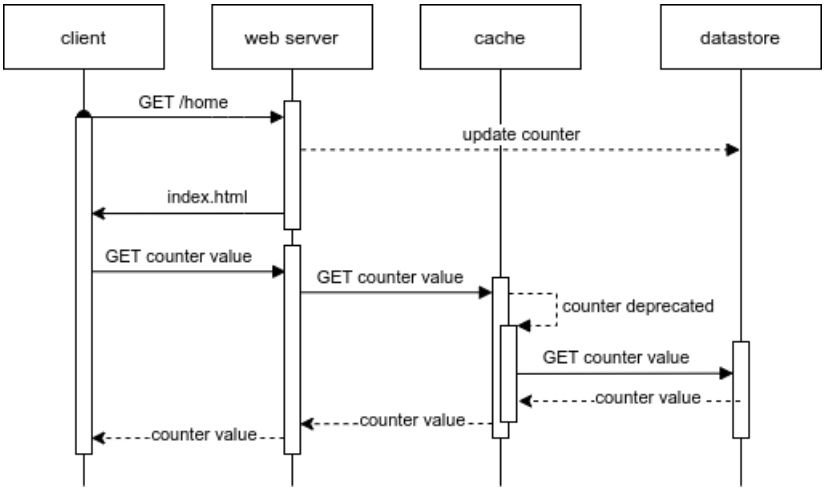


**Figura 2.5:** Diagrama de secuencia de una consulta a cache con datos inválidos.

## 2.5. Escenarios

En la figura 2.6 se presentan los casos de uso posibles: básicamente un usuario puede pedir la vista de alguna de las páginas que provee el instituto. Como el acceso a los contadores se realiza de forma separada, se ha incluido también: por más que sería raro que un usurio quiera acceder a ellos de forma aislada, la informacióne stá expuesta en la API pública y por lo tanto se lista en el diagrama.

# 3. Conclusiones

Partiendo de la experiencia de la materia Sistemas Distribuidos I, en el presente TP se ha podido aplicar dicho conocimiento de la mano de herramientas utilizadas en la industria: en cierta forma se ha logrado obtener una visión más real del desarrollo de los sistemas distribuidos.

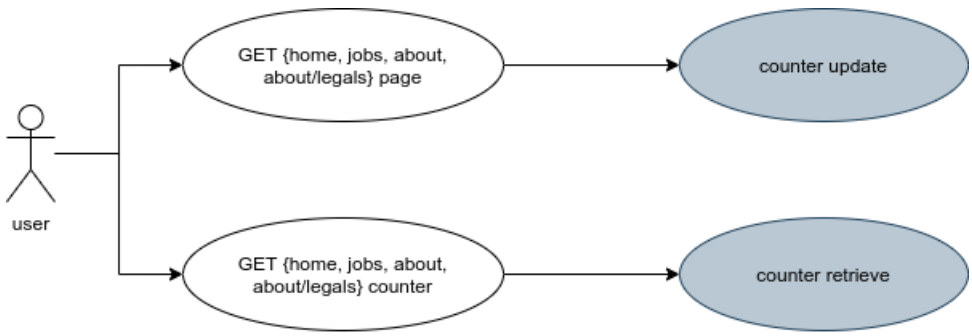


**Figura 2.6:** Diagrama de casos de uso.