Facultad de Ingeniería Universidad de Buenos Aires

75.61 Taller de Programación III

TP N°3 - Event Managment (Google App Engine)

Profesor: Andrés Veiga
JTP: Pablo Roca

Integrantes:

Padrón	Nombre	Email
89579	Torres Feyuk, Nicolás R. Ezequiel	ezequiel.torresfeyuk@gmail.com

Índice

1. Introducción

El presente trabajo práctico consiste en diseñar e implementar un sistema de recepción de eventos a través de la plataforma de cloud computing de google App Engine. El presente sistema debe permitir el ingreso de personas invitadas en diferentes eventos, como asi también tener la posibilidad de poder consultar quienes son los invitados anotados en cada uno de ellos.

2. Desarrollo

Google App Engine permite realizar el desarrollo de las aplicaciones web a través de los siguientes lenguajes de programación:

- Java
- Python
- Go
- PHP

Debido a la simplicidad del lenguaje y la falta de conocimientos en las otras opciones, se decidió implementar el sistema en Python. App Engine permite realizar el desarrollo de la parte Servidor a través de cualquier web framework de Python que implemente la interfaz WSGI(Web Service Gateway Interface). Se lista a continuación algunos de los frameworks que cumplen con esta especificación:

- Django
- CherryPy
- Pylon
- web.py
- web2pyi
- webapp2

Para realizar la presente aplicación se decidió utilizar webapp2. El motivo de esta elección es que la complejidad del presente proyecto era baja y los tutoriales de google respecto a las aplicaciones de Python para App Engine están desarrolladas con este framework.

Para el desarrollo y diseño de la página web se utilizó HTML, scripts en Javascript y JQuery y el template language Jinja2. Este último fue utilizado la generación estática del página principal del proyecto, la cual nuevamente se realizó en función del tutorial de App Engine.

3. Arquitectura 4 + 1

3.1. Casos de Uso

En la figura 1 se exhibe el diagrama de casos de uso del sistema. Se detalla a continuación cada una de las entradas del mismo:

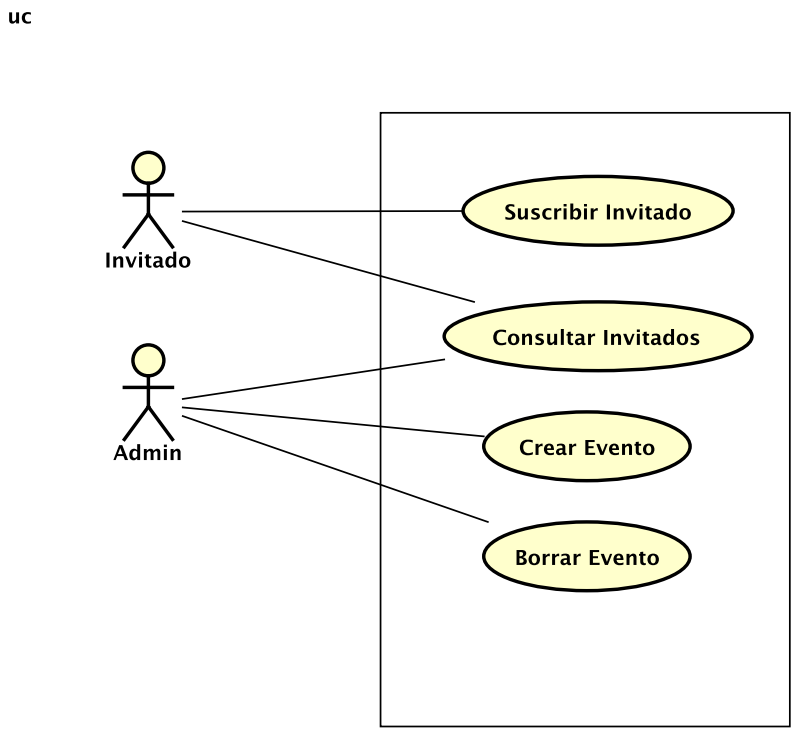

Figura 1: Diagrama de Casos de Uso

- **Suscribir Invitado:** Un invitado ingresa sus datos pertinentes a la página web e intenta registrarse. El sistema toma como identificador único del usuario su email, por lo cual en el caso de que ya exista una persona en el mismo evento con el mismo email, la invitación será rechazada.
- **Consultar Invitados:** Un usuario administrador o invitado elige el evento a consultar. Al hacerlo, la página web muestra en una tabla la lista de invitados actual del evento.
- **Crear Evento:** Un usuario administrador crea un evento e ingresa el nombre y la máxima cantidad de invitados del mismo.
- **Borrar Evento:** Un usuario administrador posicionado sobre un evento específico elimina al mismo. Al hacer esto, se borrará del storage de la

aplicación el evento junto con todos los invitados registrados en el mismo.

Cabe destacar que no existe el caso de uso **Desuscribir Invitado** debido a que el enunciado no especificaba este caso. El caso **Borrar Evento** fue agregado solamente para poder eliminar fácilmente aquellos eventos creados a partir de pruebas de carga.

3.2. Vista Lógica

En la figura 2 se exhibe el diagrama de clases, el cual muestra la vista lógica del sistema. A continuación se adjunta una breve explicación de cada una de las clases:

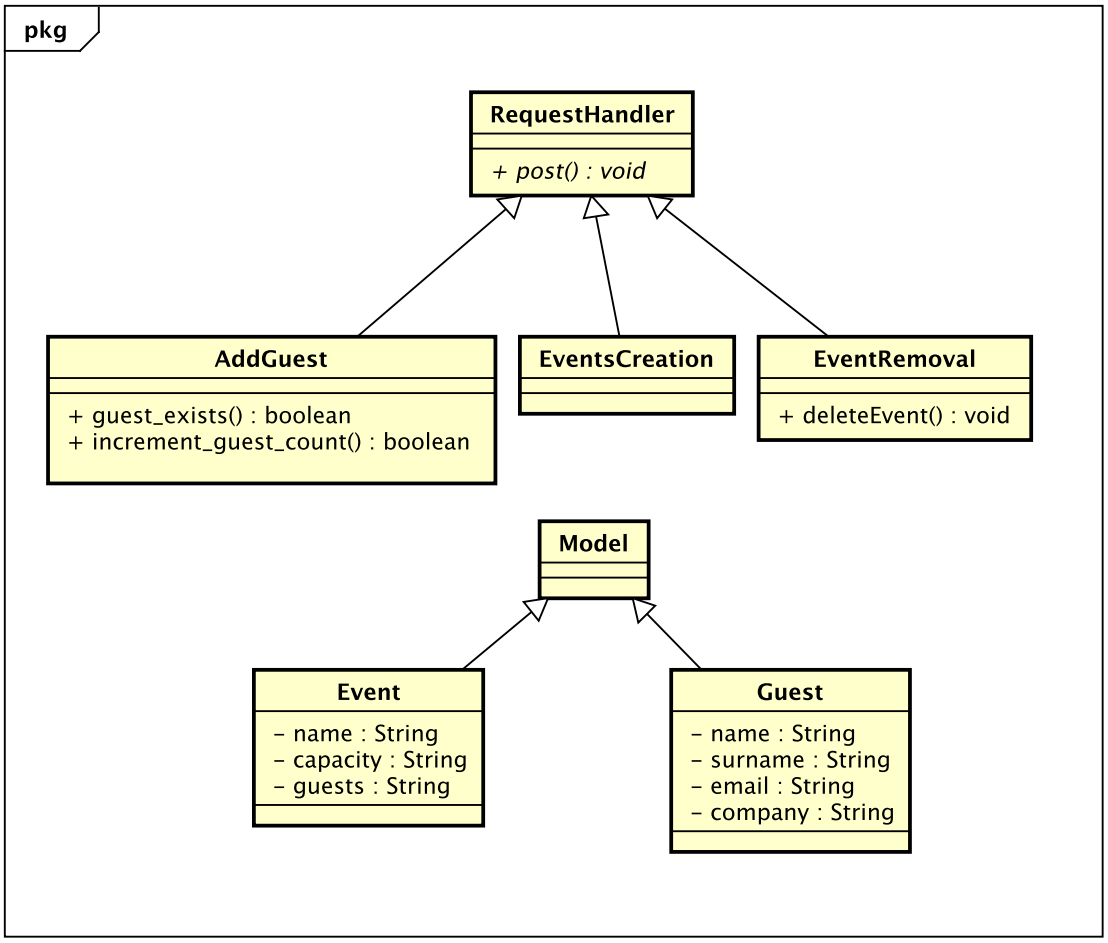

Figura 2: Diagrama de Clases

- **AddGuest:** Clase derivada de RequestHandler. Al recibir un post, se encarga de validar que el invitado a ingresar no exista y luego agrega al mismo. Además, chequea si el evento permite ingresar más invitados anter de dar el alta.
- **EventsCreation:** Clase que deriva de RequestHandler. Al recibir un post, crea un evento nuevo en el datastorage
- **EventRemoval:** Clase que deriva de RequestHandler. Al recibir un post, obtiene el evento pasado por parámetro y elimina al mismo junto con todos los invitados asociados.

- **Event:** Objeto que represeta a un evento en el datastorage.
- **Guest:** Objeto que representa a un invitado en el datastorage.

4. Pruebas de Carga

La aplicación en la nube se puede acceder a través del siguiente link. Para probar la performance de la misma, se investigaron diferentes simuladores de carga. Los requisitos del simulador a buscar son los siguientes:

- **Simplicidad:** La aplicación a testear es muy sencilla. Solo se desea enviar un POST al servidor con ciertos datos y analizar como responde el mismo.
- **Flexibilidad:**El simulador debe tener la capacidad de poder recibir un conjunto de tests y ejecutar los mismos de forma automática.
- **Concurrencia:** El simulador debe poder lanzar los tests desde N procesos/threads diferentes, para poder probar la concurrencia del sistema Web.

Los primeros simuladores a probar fueron Jmeter y The Grinder. Jmeter fue descartado inmediatamente al abrir la documentación la cual posee más de 500 hojas. Claramente este simulador no cumple el requisito de *Simplicidad* que se estaba buscando. Luego se realizaron pruebas con *The Grinder*. Este simulador resultó ser más ameno debido a que los tests se programan en Jython. Se intentó utilizar el mismo pero luego de estar una hora sin obtener resultados concretos, se decidió buscar otra alternativa.

Buscando se encontró Pylot. El mismo es un simulador de carga que recibe los tests a través de un xml y con el cual se modificar parámetros de carga a través de un archivo de configuración. Se muestra a continuación un XML de ejemplo de la simulación de un HTTPRequest.

```
<testcases>
  <case>
    <url>http://localhost:8080/add_guest</url>
    <method>POST</method>
    <body><![CDATA[actualEvent=Mondongo]]></body>
  </case>
</testcases>
```

El simulador posee los siguientes parámetros a configurar:

- **Agents:** Cantidad de threads a lanzar para realizar las simulaciones.
- **Duration:** Duración (en segundos) de la simulación.
- **RampUp:** Tiempo (en segundos) a esperar entre el lanzamiento de un agente y el proximo.
- **Interval:** Periodicidad con la cual un agente realiza un test en función de las pruebas almacenadas en el XML.

4.1. Resultados

Se muestra a continuación los resultados de las pruebas de cargas realizadas. En las mismas se muestra como se fueron modificando los parámetros del simulador, junto con la respuesta de la WebApp.

Agentes	**Duración (seg)**	**RampUp(ms)**	**Interval(ms)**	**Resp. Time (ms)**	**Requests**	**Errors**
1	30	0	500	452	57	0
1	30	0	100	478	61	0
5	30	0	100	567	237	0
10	30	0	100	1194	226	37
15	30	0	100	1390	256	78