Arquitectura

Dado el problema presentado en la sección anterior, y sabiendo que se registran varios millones de puntuaciones en forma mensual se necesita responder las siguientes preguntas:

1) Sabiendo que la **escalabilidad** y la **performance** son los dos atributos de calidad más importantes, proponga una arquitectura para resolver el problema, explicando cómo la misma tiene en cuenta los atributos de calidad mencionados.
2) Explique cómo almacenaría los datos para resolver el problema. Tenga en cuenta que puede seleccionar más de un tipo de almacenamiento para resolver lo mejor posible cada uno de los requerimientos funcionales.
3) Mencione cuál le parece la mejor opción para comunicar los eventos mencionados a otros servicios en las condiciones mencionadas.
4) En base a la arquitectura propuesta en la pregunta anterior, seleccione qué tecnologías utilizaría para implementar la solución, tanto a nivel de programación, almacenamiento y comunicación entre servicios. Como dato adicional se sabe que la empresa favorece las tecnologías libres y de código abierto frente a las propietarias y con costo de licenciamiento.

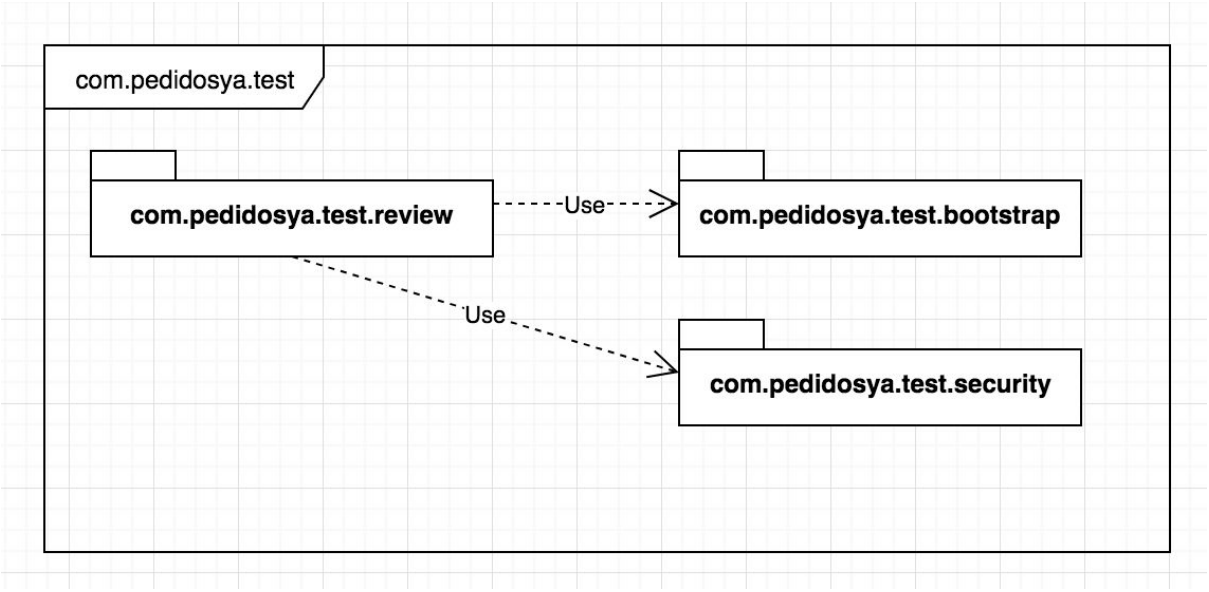
Nota: En caso de asumir aspectos del problema no mencionados en los requerimientos, aclararlos como parte de la respuesta.

Supuestos

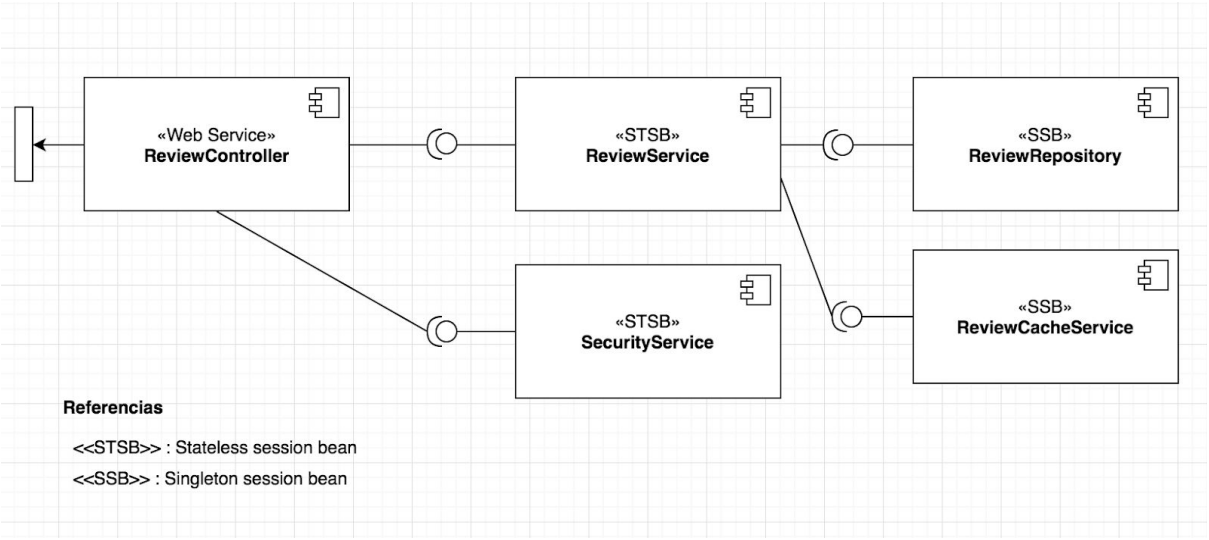
- Suponiendo que a escalabilidad se refiere a escalabilidad horizontal (agregar más recursos a las unidades lógicas, como por ejemplo agregar otro server a un pool de web servers).
- Suponiendo Performance como lo detalla el libro Software Architecture In Practice, referente al tiempo.
- Se supone que los reads de la API son mayores que los writes (por ejemplo 1 a 10).

Diagramas

Vistas de módulos - vista de uso



Vista de componentes y conectores - vista de diagrama de componentes y conectores



Preguntas

1) Para resolver el problema se me ocurrió utilizar las siguiente tácticas:

Táctica reducción de overhead: Hice todo el proceso de una review con la menor cantidad de pasos posibles e intentando reducir al máximo los intermediarios.
Introducción de concurrencia: En los pasos que pudiesen ejecutar en paralelo fueron implementados así.

Incrementar recursos: Para lograr esto podemos tener servidores web dedicados para read y para write. Además suponiendo que siempre leemos n veces más de las que escribimos podemos tener escalar en relación a este supuesto colocando un balanceador de carga en un pool de servidores web, es decir, tengo dos pooles de instancias uno para read y otro para write con determinada cantidad de instancias (respetando la relación de los supuestos) y eso recibe tráfico mediante un balanceador de carga.
Mantener múltiples copias de cómputo: Al igual que la explicación anterior.
Mantener múltiples copias de la data: Se decidió utilizar un caché, que replica datos de la base para poder tener un acceso rápido a los mismos.

2) Para poder cumplir con los requerimientos de escalabilidad y performance recomiendo utilizar a nivel de almacenamiento

- Una base de datos no relacional para el manejo de las entidades del sistema, es bien conocido que los motores no relacionales son más fáciles de escalar.
- Un cache de acceso rápido para las consultas de READ (en específico los GET por keys)
- Para las búsquedas utilizar un motor especializado en búsquedas sobre documentos (si bien no hay que detallar tecnologías, algo como solr, elasticsearch)
- [opcional] En las grandes empresas normalmente existen equipos de BI dedicados a lo que es minería de datos, si bien la letra no lo especifica creo que es una buena práctica manejar una réplica en algún motor relacional para poder explotarlo desde ese equipo en alguna herramienta de BI.

3) Para esta pregunta claramente utilizaría un servicio de colas de mensajes para poder utilizar el patrón conocido como publicador-suscriptor, es bien sabido que si mi servicio (publicador) no quiere conocer a todos los interesados debe utilizar esto.

4) Como tecnologías propongo:

- Backend: Java 8 + Spark + Guice (para manejar la inyección de dependencias y evitar utilizar enumerados como singletons de la JVM)
- Almacenamiento:
    - Motor no relacional MongoDB me inclino por este ya que es de los que tengo un poco mas de conocimiento, y se que funciona muy bien.
    - Para el cache se puede ir por una solución muy simple a nivel de cada web server como Guava que es muy fácil de configurar y funciona bastante bien. Sin embargo si queremos algo más distribuido y no dependiente del servidor en el cual estamos corriendo podemos utilizar Redis (link a vs con Memcached y Cassandra)
    - Motor de búsquedas por excelencia ElasticSearch tiene la flexibilidad y rápida respuesta que necesita el sistema. Y en comparación con Solr mucho más soporte. Otra opción sería usar Lucene pero realmente desconozco la complejidad ponerlo andando y además entiendo que ElasticSearch se basa en el mismo para su motor de búsqueda y le agrega muchas capabilities copadas.

- Para BI utilizaría algo como Teradata combinado con Alation y Tableau tal vez lo cual da un combo muy interesante ya que Teradata nos permite almacenar toda la información Alation nos permite publicar ese datasource para que pueda ser explotado por toda la compañía y Tableau nos permite explotar la data mediante gráficos.

Diseño

Dado que se decide implementar el servicio como un único api rest, describa que endpoints tendría dicha api, mencionando el verbo http, la url, query string y body de los requests, así como el formato las responses.

ReviewController

A continuación se detallan los endpoints con su verbo http, la url, los params, query params y body de ser necesario.

POST /v1/reviews

Body

```
{
        “user_id”:”82033667-faf5-4625-bac0-8e83911541b6”,
        “shop_id”:”9f9d4e4d-1e01-4f9b-a6cd-2c4689435908”,
        “purchase_id”:”105311fe-692e-4b15-8917-8ea9da809b1d”,
        “comment”:”Este comentario es un comentario de test para opinar”,
        “score”:4
}
```

Tipos de datos:

user_id: UUID
shop_id: UUID
purchase_id: UUID
comment: String
score: int entre 1 y 5

Retorno:


```
{
   "id": "92ff9301-3c63-4907-8e71-de56c82315f7",
   "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
   "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
   "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
   "timestamp": {
      "date": {
         "year": 2018,
         "month": 11,
         "day": 19
      },
      "time": {
```

```
            "hour": 19,
            "minute": 46,
            "second": 21,
            "nano": 350000000
        }
    },
    "comment": "Este comentario es un comentario de test para opinar",
    "score": 4
}
```


GET /v1/reviews/purchases/:purchase_id

Body

No Aplica

Tipos de datos:

purchase_id: UUID

Retorno:


```
{
    "id": "92ff9301-3c63-4907-8e71-de56c82315f7",
    "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
    "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
    "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
    "timestamp": {
        "date": {
            "year": 2018,
            "month": 11,
            "day": 19
        },
        "time": {
            "hour": 19,
            "minute": 46,
            "second": 21,
            "nano": 350000000
        }
    },
    "comment": "Este comentario es un comentario de test para opinar",
    "score": 4
}
```

GET /v1/reviews/shop/:shop_id

Body

No Aplica

Tipos de datos:

shop_id: UUID

Retorno:


```
[
        {
            "id": "92ff9301-3c63-4907-7a71-de56c82315f7",
            "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
            "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
            "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
            "timestamp": {
                "date": {
                    "year": 2018,
                    "month": 11,
                    "day": 19
                },
                "time": {
                    "hour": 19,
                    "minute": 46,
                    "second": 21,
                    "nano": 350000000
                }
            },
            "comment": "Este comentario es un comentario de test para opinar",
            "score": 4
        },
        {
            "id": "92ff9301-3c63-4907-8e71-de56c82315f7",
            "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
            "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
            "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
            "timestamp": {
                "date": {
                    "year": 2018,
                    "month": 11,
                    "day": 19
                },
                "time": {
```

```
                "hour": 19,
                "minute": 46,
                "second": 21,
                "nano": 350000000
            }
        },
        "comment": "Este comentario es un comentario de test para opinar",
        "score": 4
    }
]
```


GET /v1/reviews/:review_id

Body

No Aplica

Tipos de datos:

review_id: UUID

Retorno:


```
{
   "id": "92ff9301-3c63-4907-8e71-de56c82315f7",
   "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
   "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
   "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
   "timestamp": {
      "date": {
         "year": 2018,
         "month": 11,
         "day": 19
      },
      "time": {
         "hour": 19,
         "minute": 46,
         "second": 21,
         "nano": 350000000
      }
   },
   "comment": "Este comentario es un comentario de test para opinar",
   "score": 4
}
```


DELETE /v1/reviews/purchases/:purchase_id

Body

No Aplica

Tipos de datos:

purchase_id: UUID

Retorno:


```
{
   "id": "92ff9301-3c63-4907-8e71-de56c82315f7",
   "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
   "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
   "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
   "timestamp": {
      "date": {
         "year": 2018,
         "month": 11,
         "day": 19
      },
      "time": {
         "hour": 19,
         "minute": 46,
         "second": 21,
         "nano": 350000000
      }
   },
   "comment": "Este comentario es un comentario de test para opinar",
   "score": 4
}
```


PUT /v1/reviews/:review_id

Body


```
{
   "id": "92ff9301-3c63-4907-8e71-de56c82315f7",
   "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
   "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
   "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
   "timestamp": {
      "date": {
         "year": 2018,
         "month": 11,
         "day": 19
      },
      "time": {
         "hour": 19,
         "minute": 46,
         "second": 21,
         "nano": 350000000
      }
   },
   "comment": "Este comentario es un comentario de test para opinar",
   "score": 4
```

}

Tipos de datos:
review_id: UUID
id: UUID
user_id: UUID
shop_id: UUID
purchase_id: UUID
comment: String
score: int entre 1 y 5
timestamp: formato standard de json para localdatetime de Java 8

Retorno:


```
{
   "id": "92ff9301-3c63-4907-8e71-de56c82315f7",
   "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
   "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
   "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
   "timestamp": {
      "date": {
         "year": 2018,
         "month": 11,
         "day": 19
      },
      "time": {
         "hour": 19,
         "minute": 46,
         "second": 21,
         "nano": 350000000
      }
   },
   "comment": "Este comentario es un comentario de test para opinar",
   "score": 4
}
```


PUT /v1/reviews/:purchase_id

Body


```
{
   "id": "92ff9301-3c63-4907-8e71-de56c82315f7",
   "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
   "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
   "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
   "timestamp": {
      "date": {
```

```
            "year": 2018,
            "month": 11,
            "day": 19
        },
        "time": {
            "hour": 19,
            "minute": 46,
            "second": 21,
            "nano": 350000000
        }
    },
    "comment": "Este comentario es un comentario de test para opinar",
    "score": 4
}
```


Tipos de datos:
purchase_id: UUID
id: UUID
user_id: UUID
shop_id: UUID
purchase_id: UUID
comment: String
score: int entre 1 y 5
timestamp: formato standard de json para localdatetime de Java 8

Retorno:


```
{
    "id": "92ff9301-3c63-4907-8e71-de56c82315f7",
    "user_id": "82033667-faf5-4625-bac0-8e83911541b6",
    "shop_id": "9f9d4e4d-1e01-4f9b-a6cd-2c4689435908",
    "purchase_id": "105311fe-692e-4b15-8917-8ea9da809b1d",
    "timestamp": {
        "date": {
            "year": 2018,
            "month": 11,
            "day": 19
        },
        "time": {
            "hour": 19,
            "minute": 46,
            "second": 21,
            "nano": 350000000
        }
    },
    "comment": "Este comentario es un comentario de test para opinar",
```

```
    "score": 4
}
```