Universidad de Buenos Aires

Facultad de Ingeniería

Propuesta de Trabajo Profesional

Qin – Cluster

Análisis de la arquitectura

Barrabino, Diego	dbarrabino@gmail.com	80183
Moreyra, Martín	moreyramj@gmail.com	84394
http://code.google.com/p/qin-cluster/		

Tutora: Licenciada Adriana Echeverría
Co Tutora: Ingeniera Julia Garibaldi

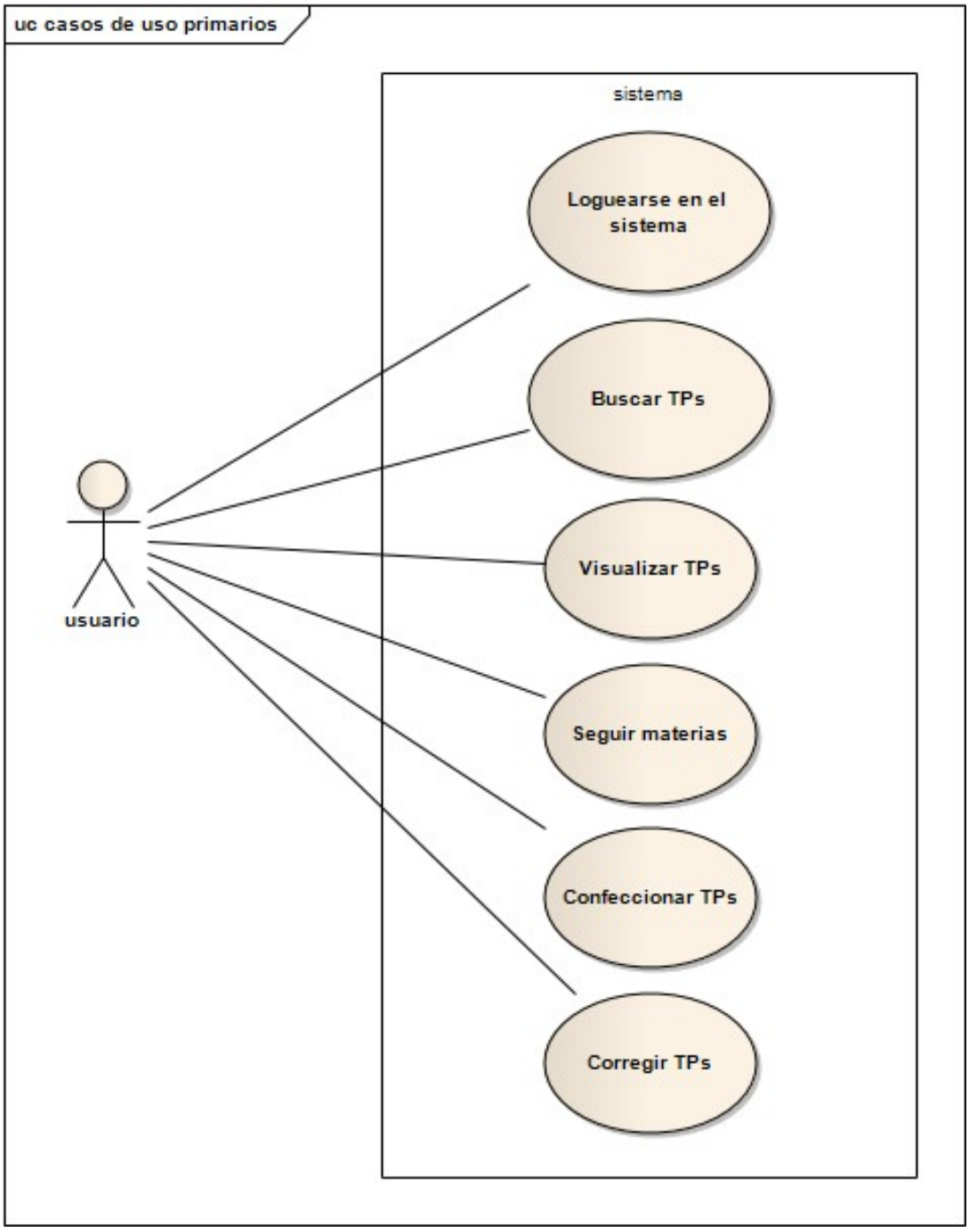
Observaciones:

Índice

01.- Análisis general

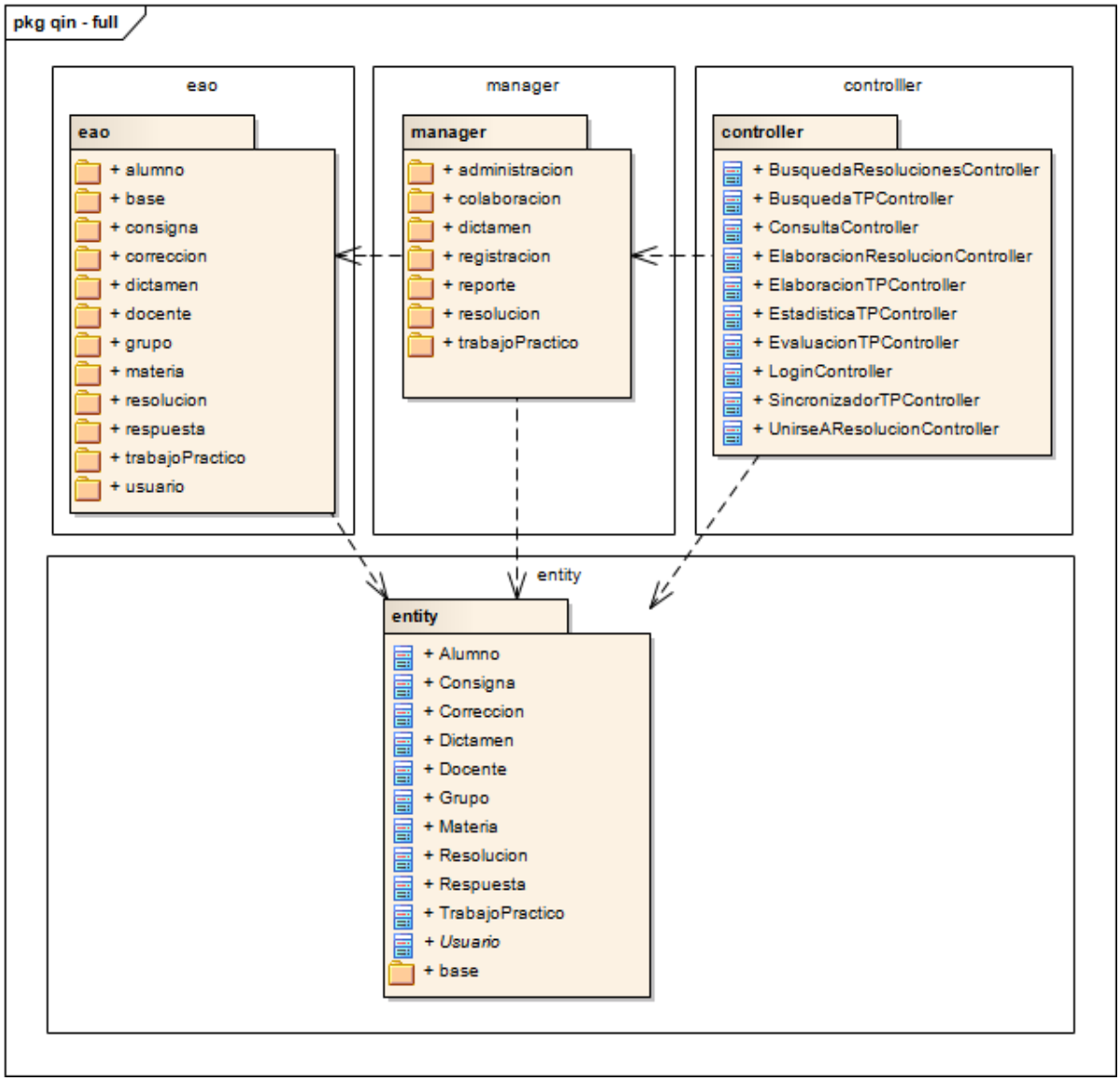
01.01.- Casos de uso

Los casos de uso contemplados son:



01.02.- Vista Lógica

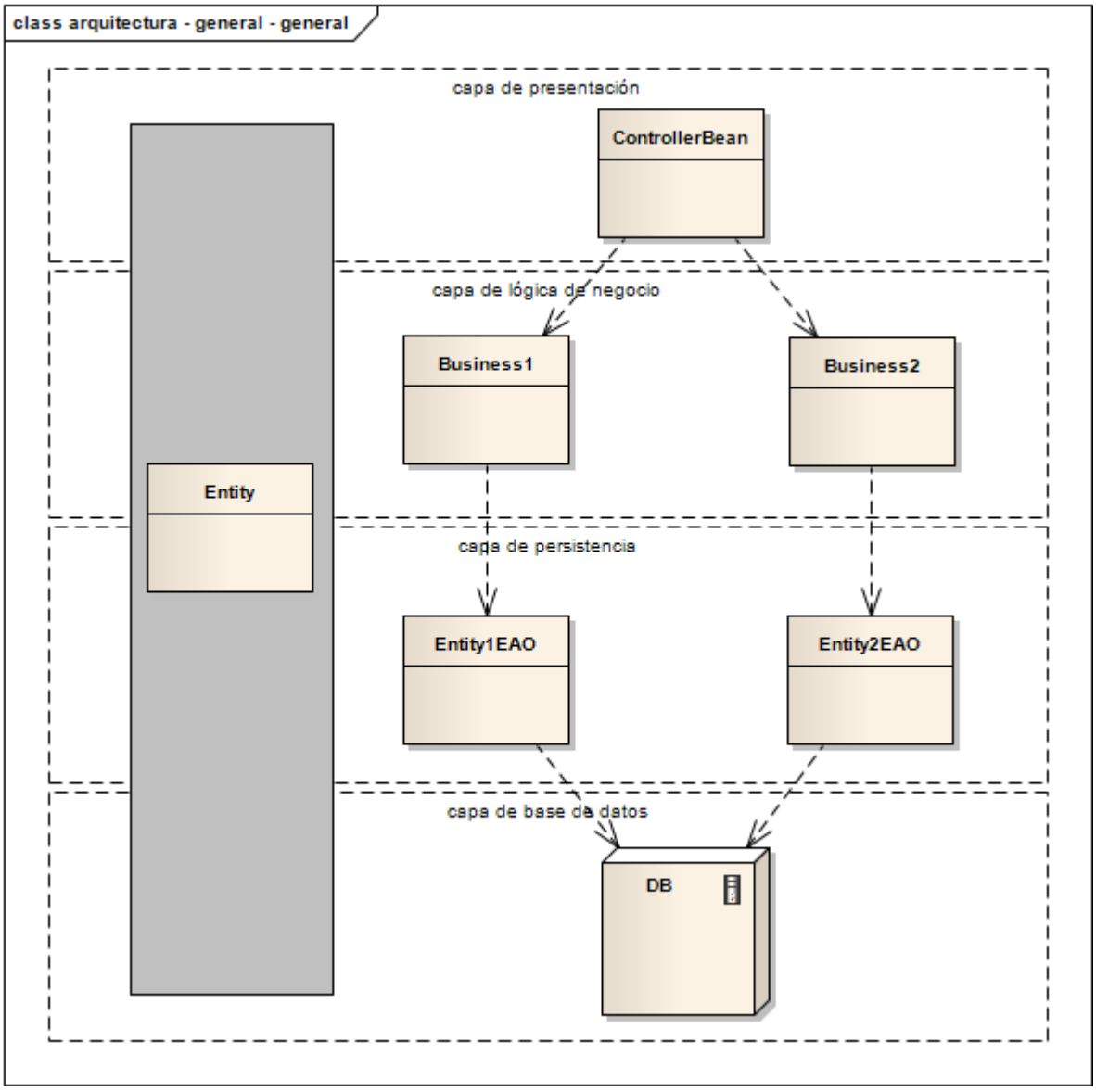
Diagrama de paquetes



Nótese como las entidades se atraviesan todas las capas de la aplicación.

Arquitectura general

Se eligió una arquitectura de capas. Esto nos permite desacoplar las distintas partes de la aplicación, pudiendo realizar cambios fácilmente en una capa sin afectar a otras. Otro beneficio de la separación en capas planteada, es que luego se podrán armar paquetes para ser distribuidos en diferentes servidores, lo que será útil al momento de armar el cluster.



En el gráfico anterior podemos observar la separación en capas, pero también se hace referencia a patrones específicos utilizados en cada capa.

- Capa de Presentación: MVC (Model-View-Controller)
- Capa de lógica de negocio
- Capa de persistencia: EAO (Entity-Access-Object)
-

Dichos patrones y otros que no figuran en el gráfico serán expuestos en las secciones siguientes.

Presentación

La capa de presentación utilizara el patrón MVC, en particular utilizará el Framework SpringMVC. Este framework es el encargado de manejar los pedidos (Request) de los navegadores, y hacer la llamada al Controlador correspondiente, decidiendo luego la vista a renderizar.
(http://static.springsource.org/spring/docs/2.0.x/reference/mvc.html)

El proceso descripto se muestra a continuación:

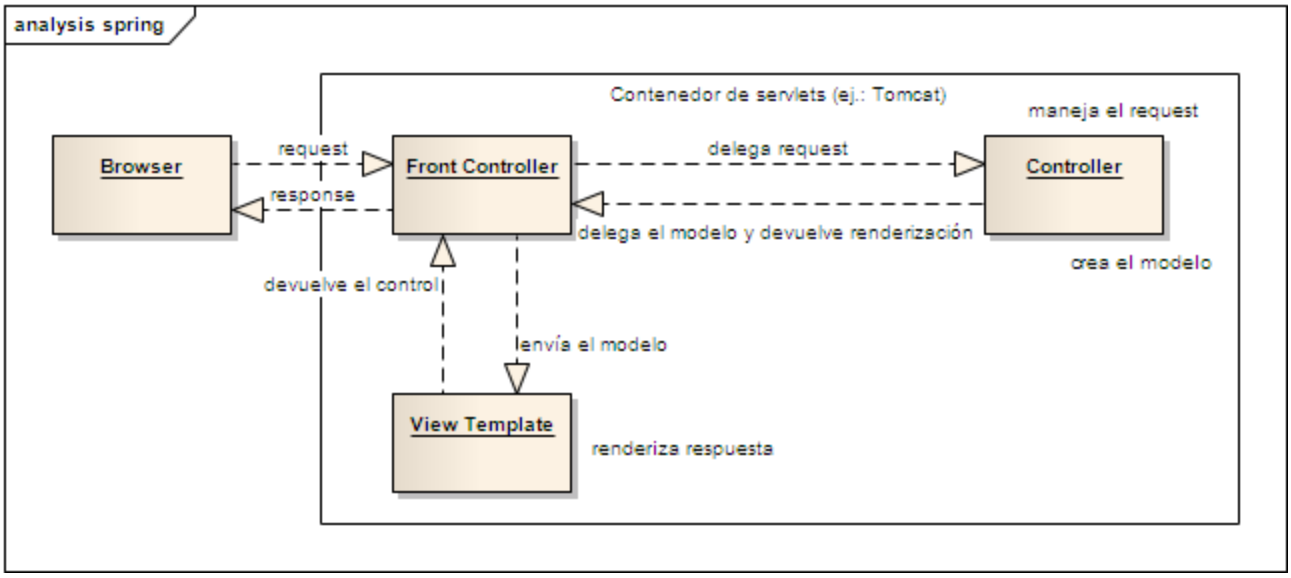


Diagrama de clases del paquete controller

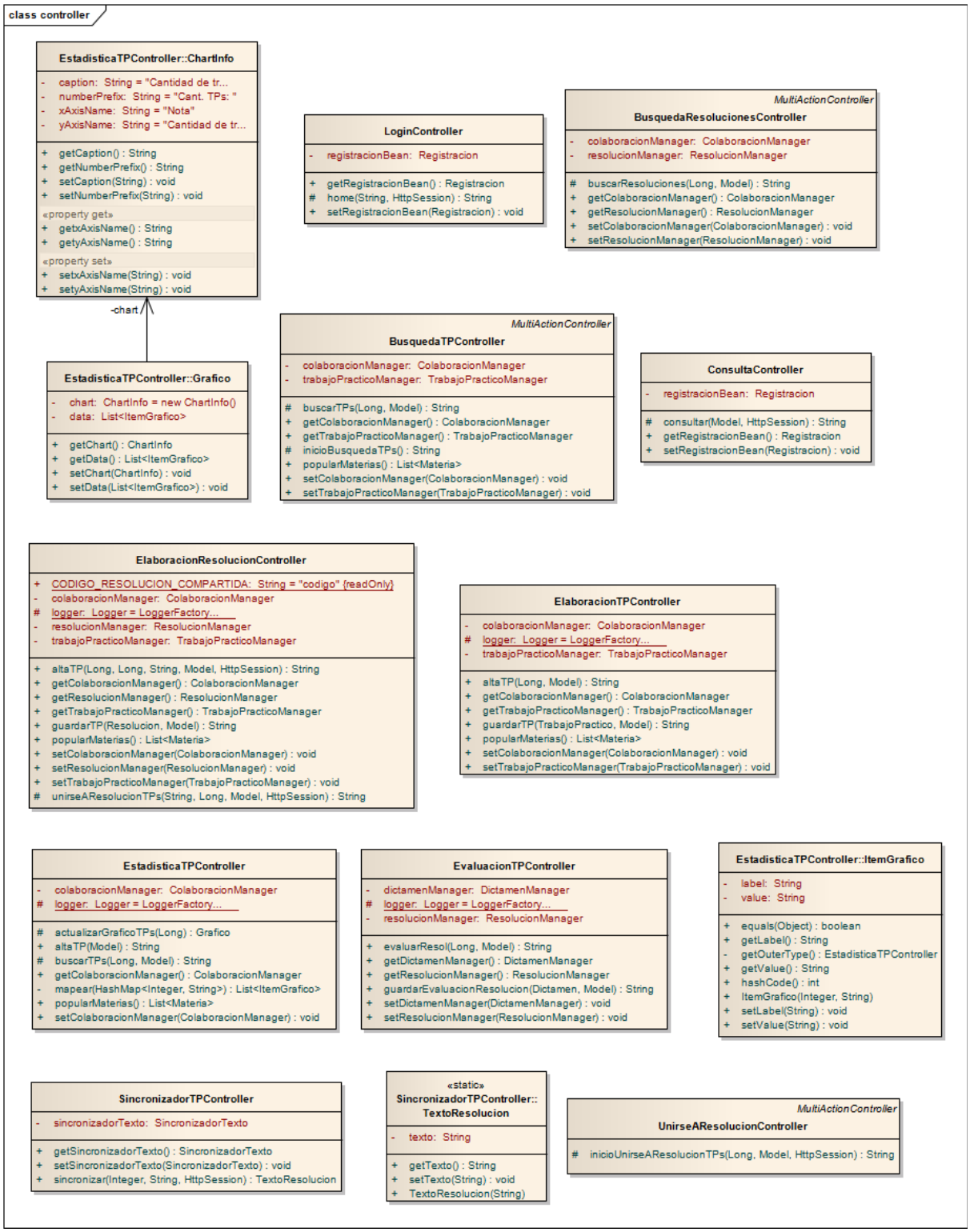
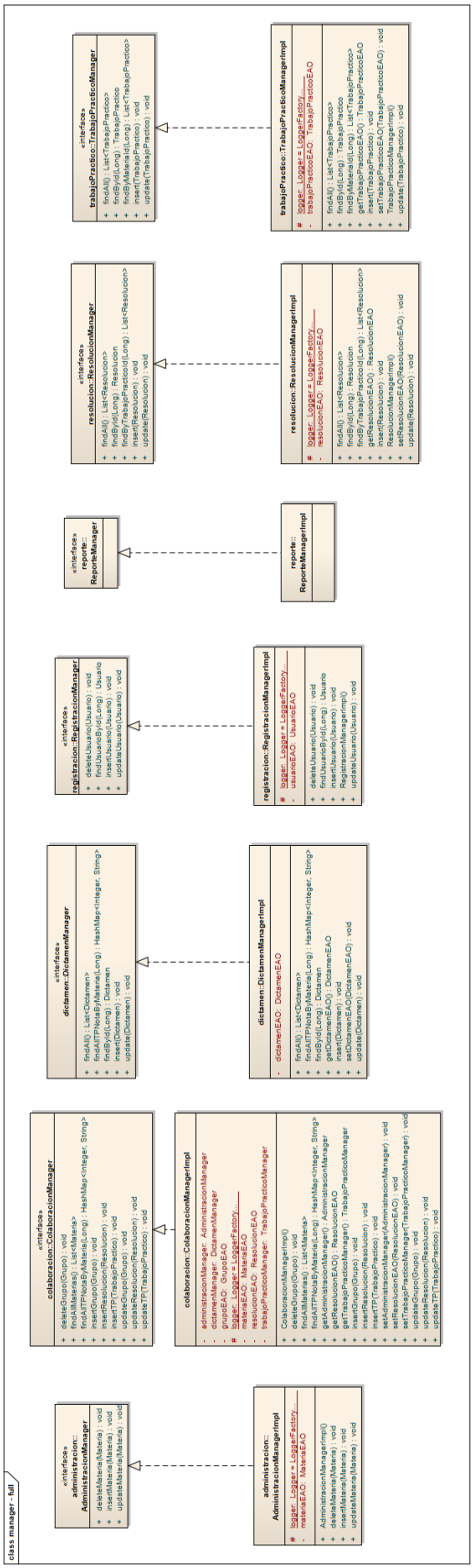
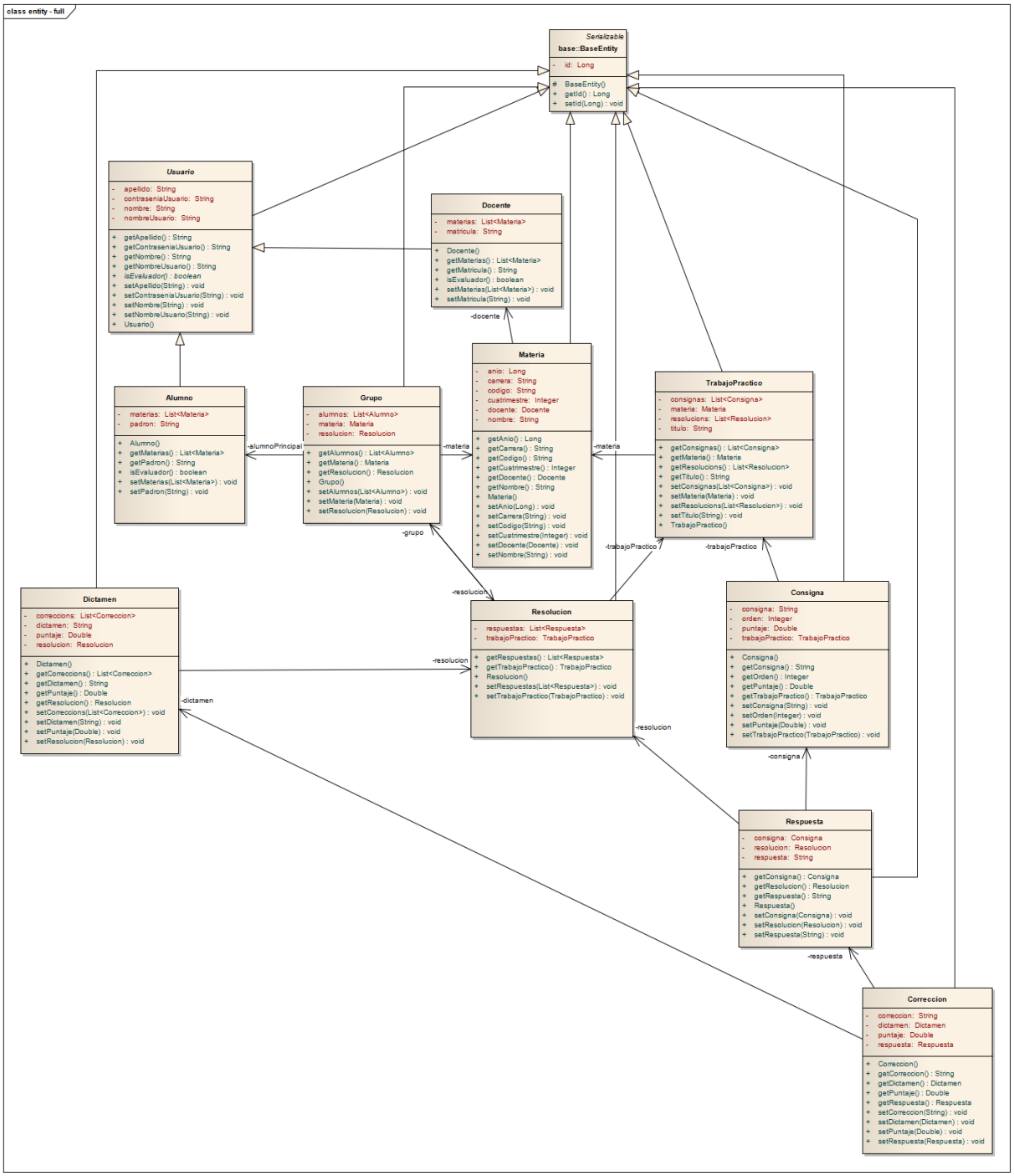


Diagrama de clases del paquete manager



Lógica de negocio

Diagrama de entidades completo



Persistencia

Para separar la capa de persistencia se utiliza el patrón Entity-Access-Object (EAO – propuesto por el libro "EJB3 In Action" que figura en la bibliografía), este patrón provee una interfaz con las operaciones de persistencia, suficientemente versátil como para permitir trabajar correctamente en ambos servidores considerados.

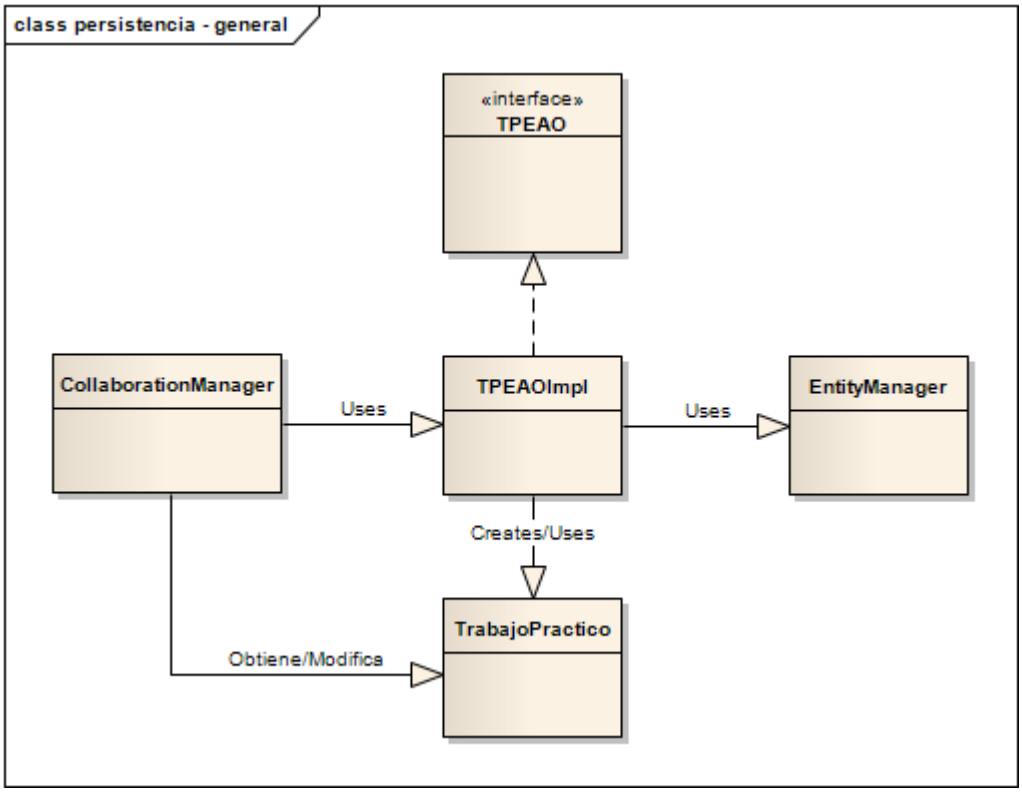
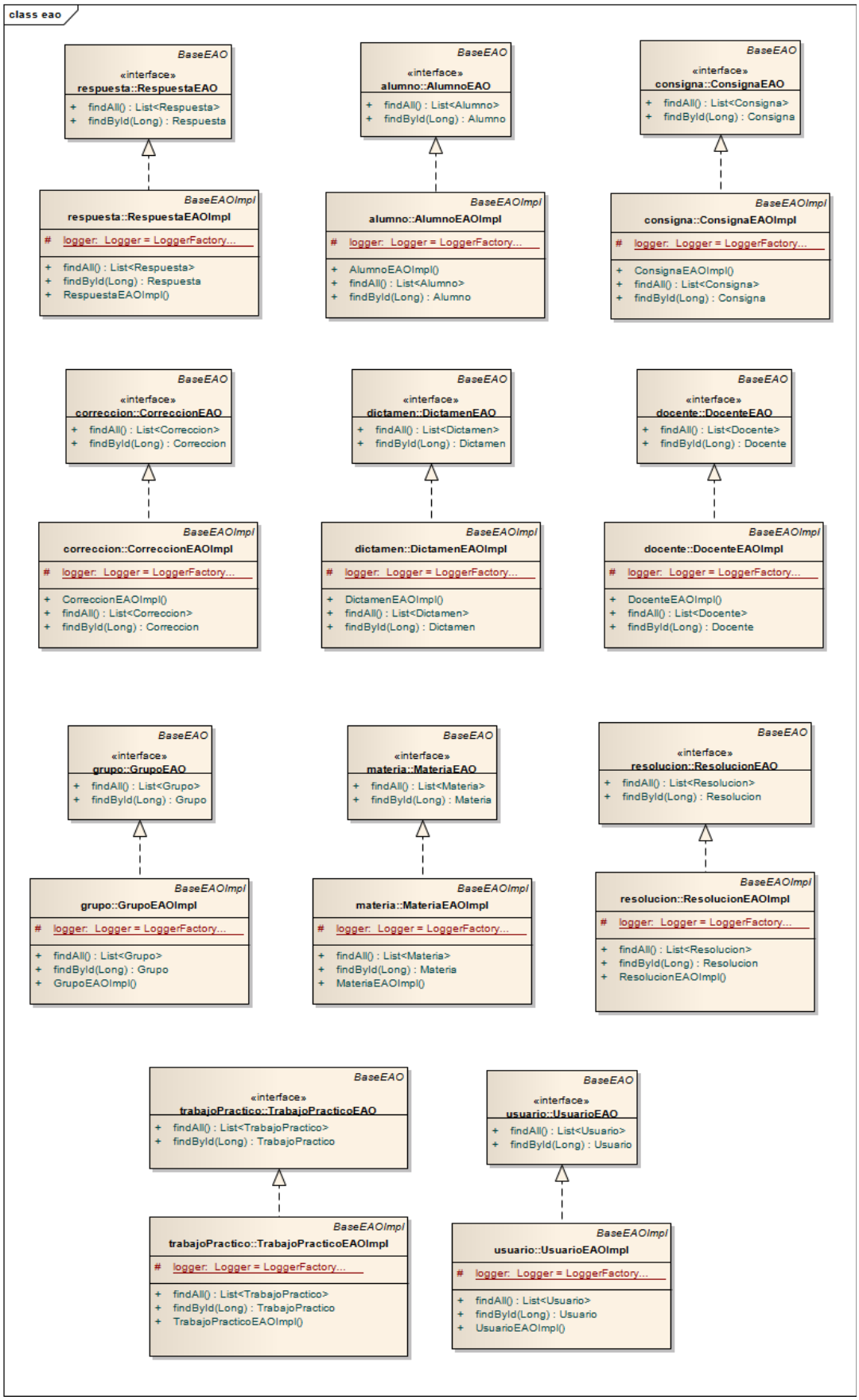


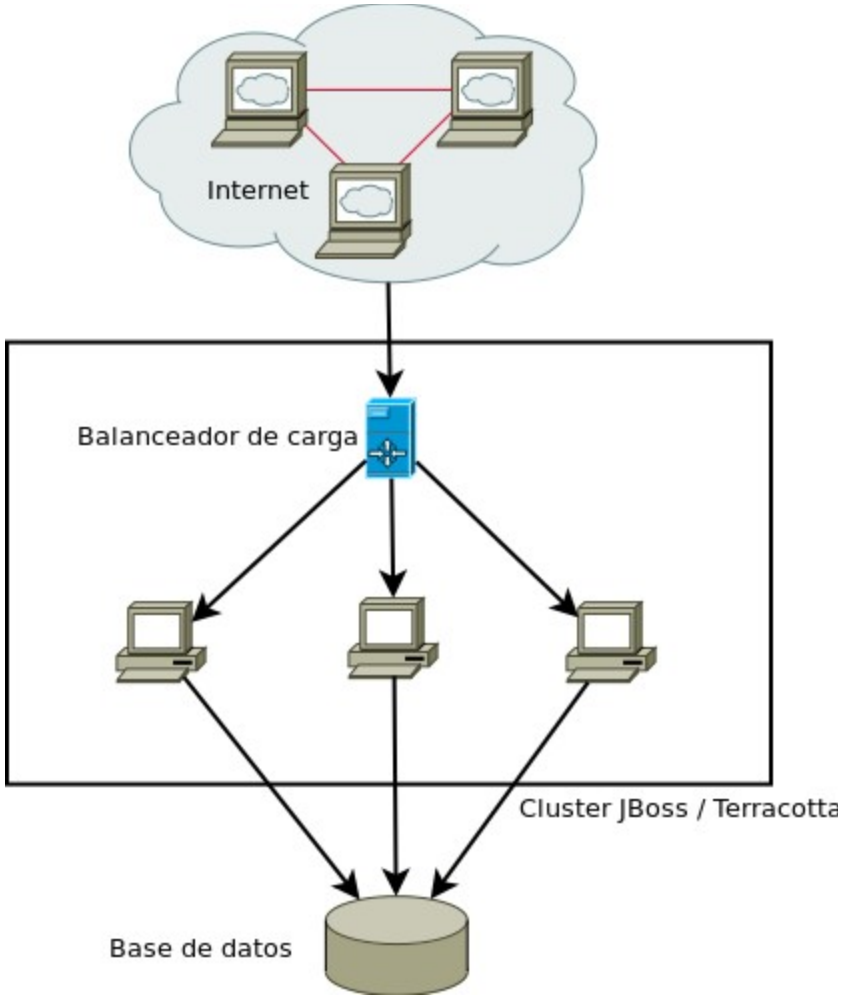
Diagrama de clases del paquete eao



02.- Desarrollo de casos de prueba generales

Para desarrollar los casos de prueba generales, se debe definir el contexto normal de funcionamiento de la aplicación, para poder efectuar pruebas en escenarios comparables con el caso general de funcionamiento, y con más y con menos exigencia que en este último caso.

El contexto normal de funcionamiento es:



[…]

No se contempla que el servidor con la base de datos experimente falencias. Se considera que en un esquema productivo real, se pueden invertir parte de los recursos preservados por administrar un esquema de clustering en infraestructura para el servidor de base de datos.

Tiempos de respuesta esperados:

Descripción de la situación	Limite tiempo
Usuarios manipulando objetos en forma directa en la UI	.1 segundo
Usuarios navegando la aplicación	1 segundo
Mostrar gráficos y reportes	10 segundos
Aceptar y procesar toda la entrada del usuario	10 segundos

Estos tiempos son tiempos aproximados y no necesariamente significa que si la respuesta tarda mas del límite es inaceptable, sino que son tiempos de espera deseados, y bajo los cuales no se pierde la atención del usuario, ya que el mismo siente cierta fluidez en la interacción con la aplicación. Dichos tiempos pueden variar, pero deberían en promedio mantenerse por debajo de los límites citados.
La tabla de tiempos es estándar en cualquier aplicación, en particular en una aplicación web no se espera ninguna variación con respecto a tiempos en una aplicación local. (http://java.sun.com/products/jlf/at/book/Responsiveness5.html)

02.01.- Planilla

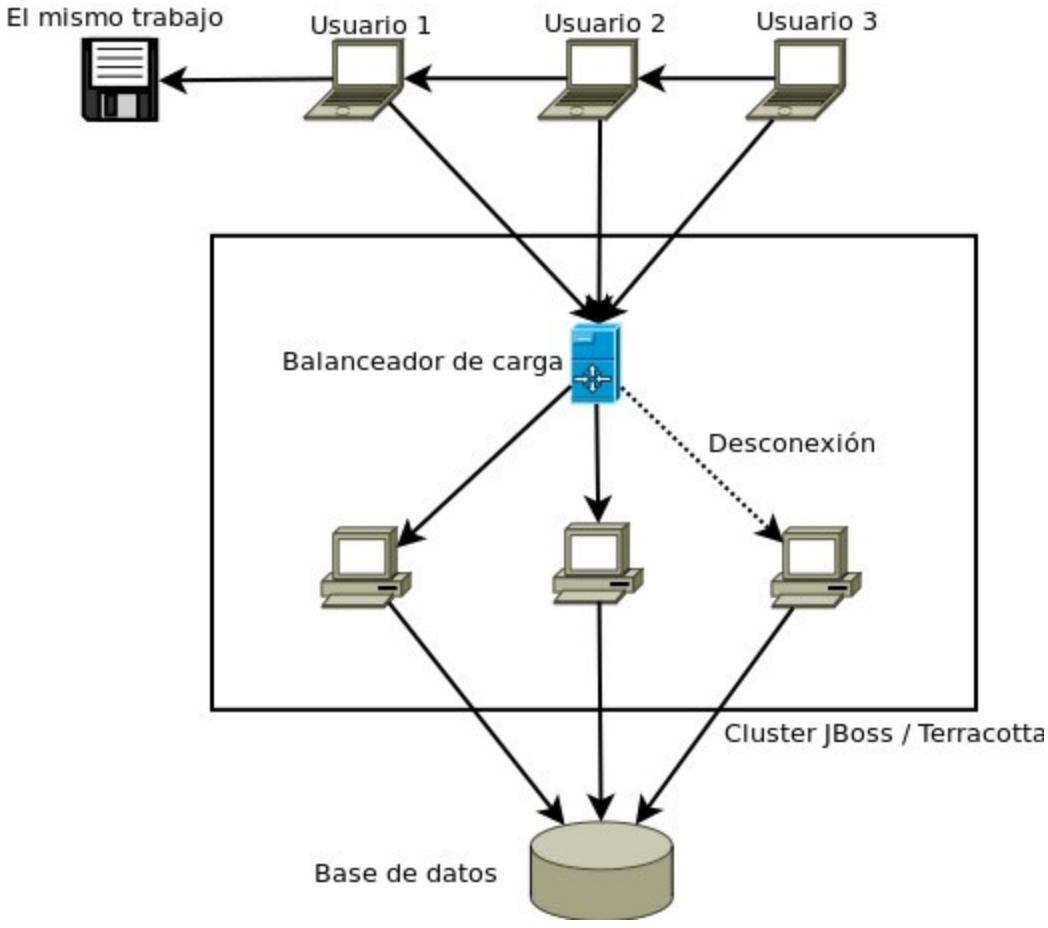
Para armar los casos de prueba generales, se confeccionó la siguiente planilla, que instanciaremos en cada uno de las pruebas generales definidas.

Descripción					
Clasificación		**Aspecto**			
		Disponibilidad	**Carga**	**Performance**	**Distribución de cache**
Capa de ejecución	**Vista – Controlador (VC)**				
	Controlador – Negocio (CN)				
Tipo de ejecución		**[Manual / Automática]**			
Entrada					
Salida					
Criterio de éxito					

02.02.- Pruebas generales definidas

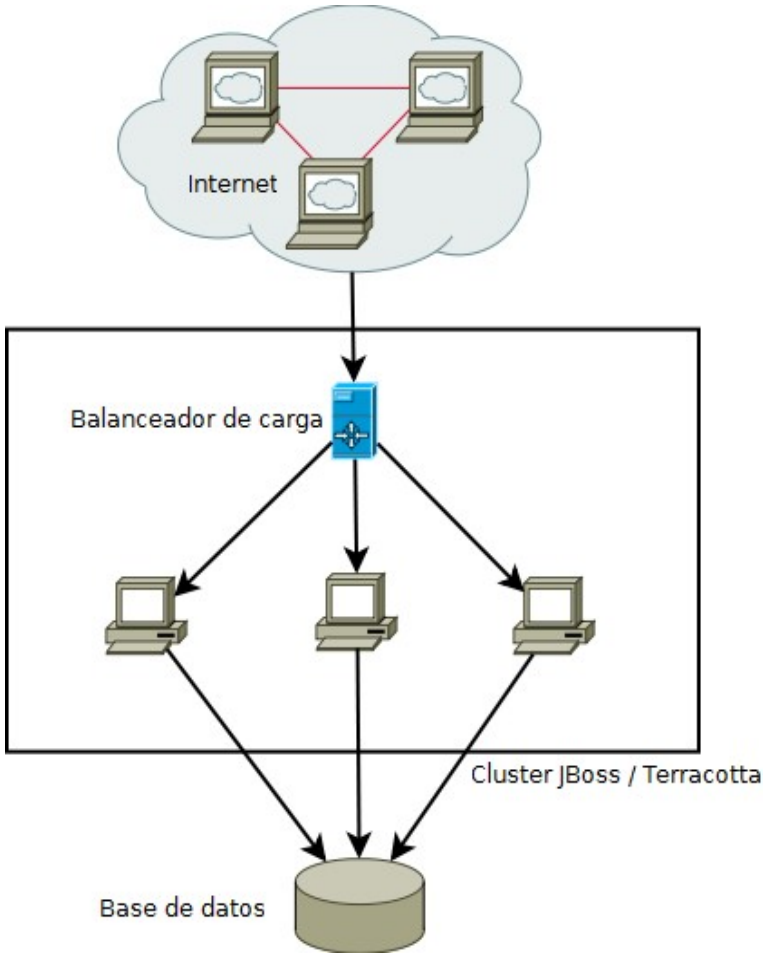
02.02.01.- Prueba general: Desconexión de un nodo en esquema colaborativo

Descripción		Desconexión de un nodo en un esquema colaborativo, ya sea de confección de trabajo práctico o de su correspondiente dictamen
Clasificación		**Aspecto**
		Disponibilidad
Capa de ejecución	**VC**	**X**
Tipo de ejecución		**Manual**
Entrada		Instancia de contexto normal de funcionamiento con texto sincronizado
Salida		Trabajo práctico o dictamen en un determinado estado de sincronización
Criterio de éxito		Correcto funcionamiento e ininterrumpido de la sincronización del texto del trabajo práctico o dictamen



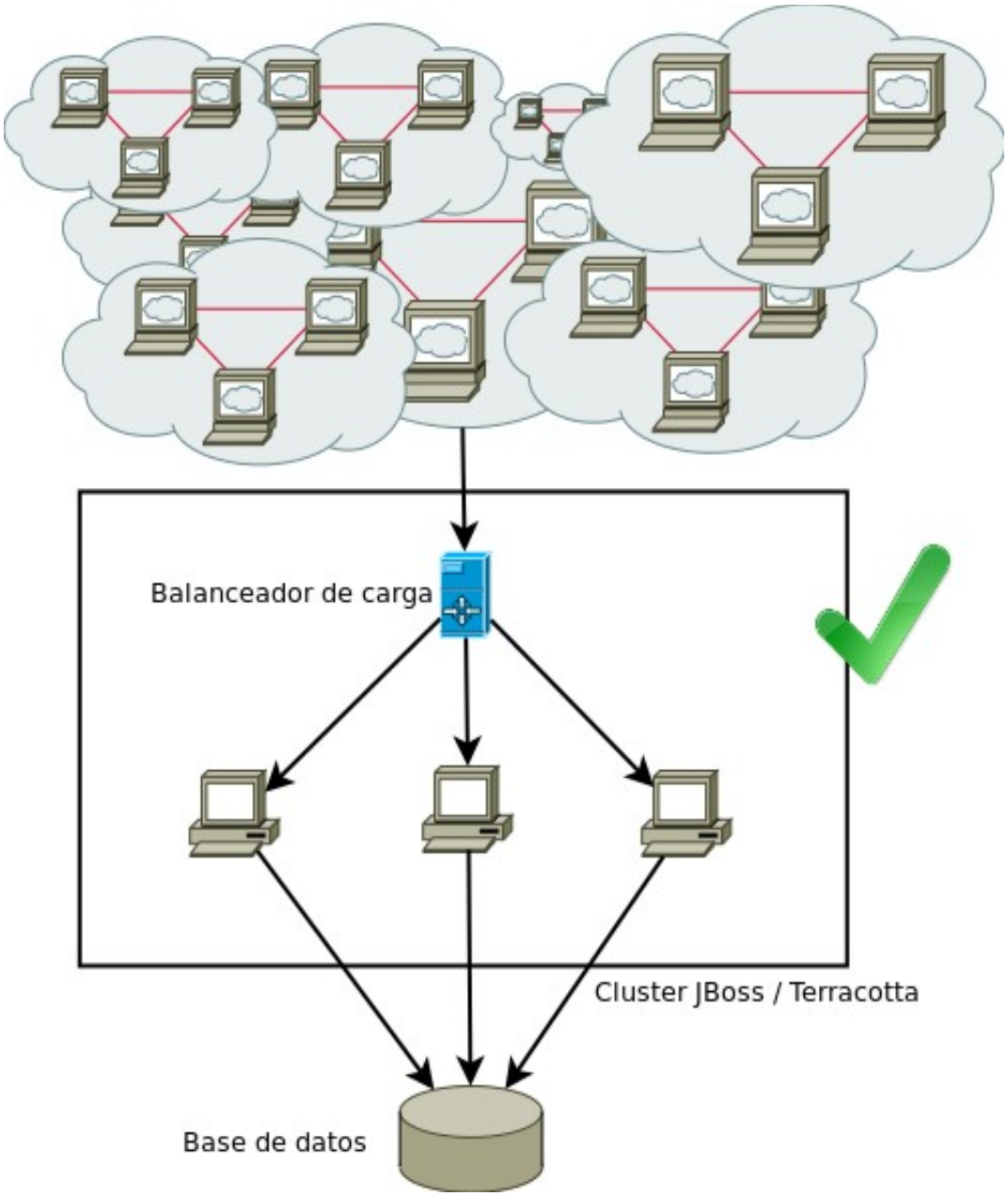
02.02.02.- Prueba general: Performance dentro del contexto normal de funcionamiento

Descripción		Performance dentro del contexto normal de funcionamiento
Clasificación		**Aspecto**
		Performance
Capa de ejecución	**VC**	**X**
Tipo de ejecución		**Automática**
Entrada		Contexto normal de funcionamiento
		Instancia con JMeter
Salida		Mediciones
Criterio de éxito		Mediciones que indican tiempos normales de funcionamiento



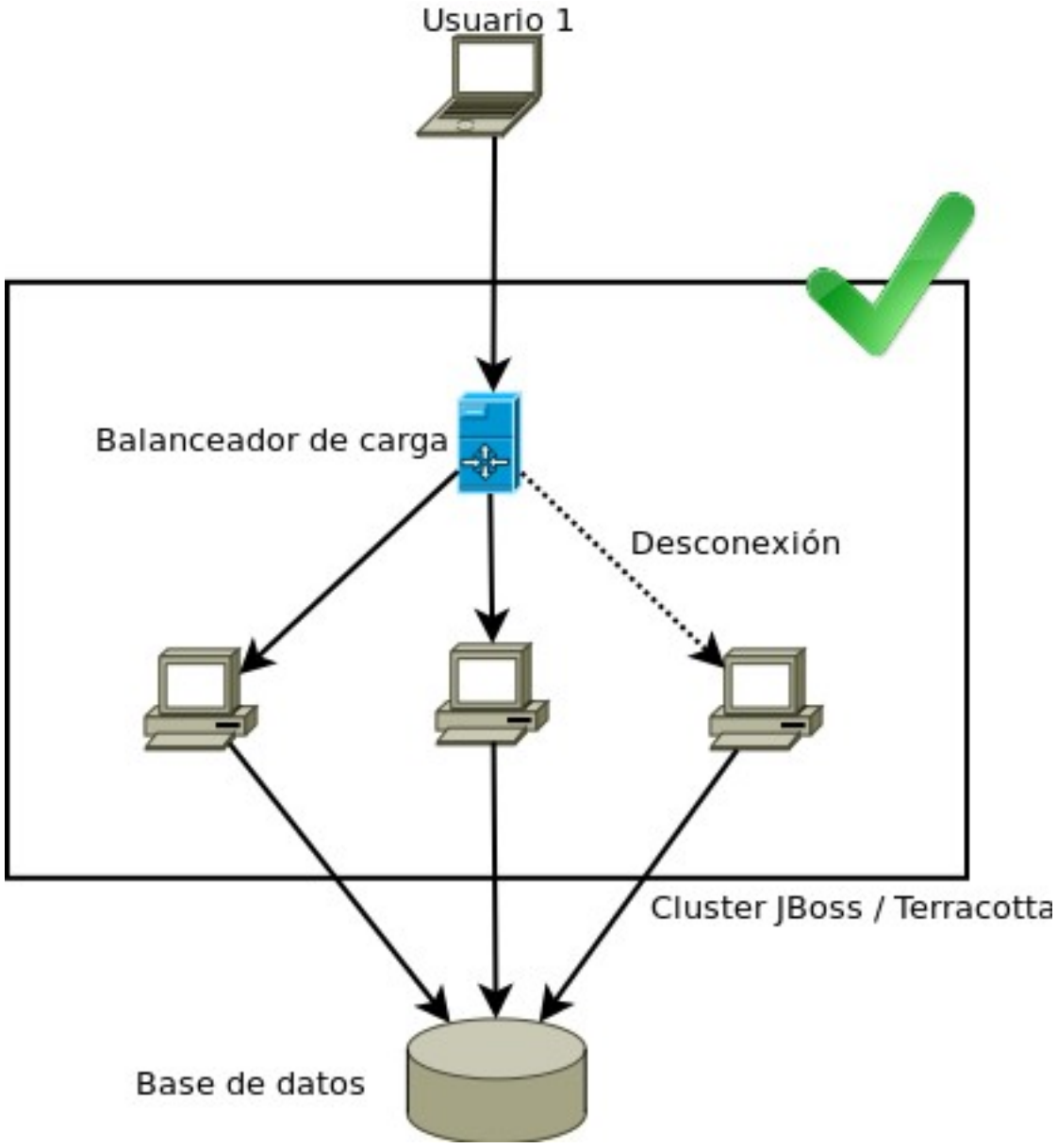
02.02.03.- Prueba general: Prueba normal de carga

Descripción		Escenario de prueba normal de carga
Clasificación		**Aspecto**
		Carga
Capa de ejecución	**CN**	**X**
Tipo de ejecución		**Automática**
Entrada		Escenario que represente un nivel de carga al menos 10 veces superior al nivel de carga normal
		Instancia con JMeter
Salida		Mediciones
Criterio de éxito		Mediciones que indican que el sistema siempre dá respuesta



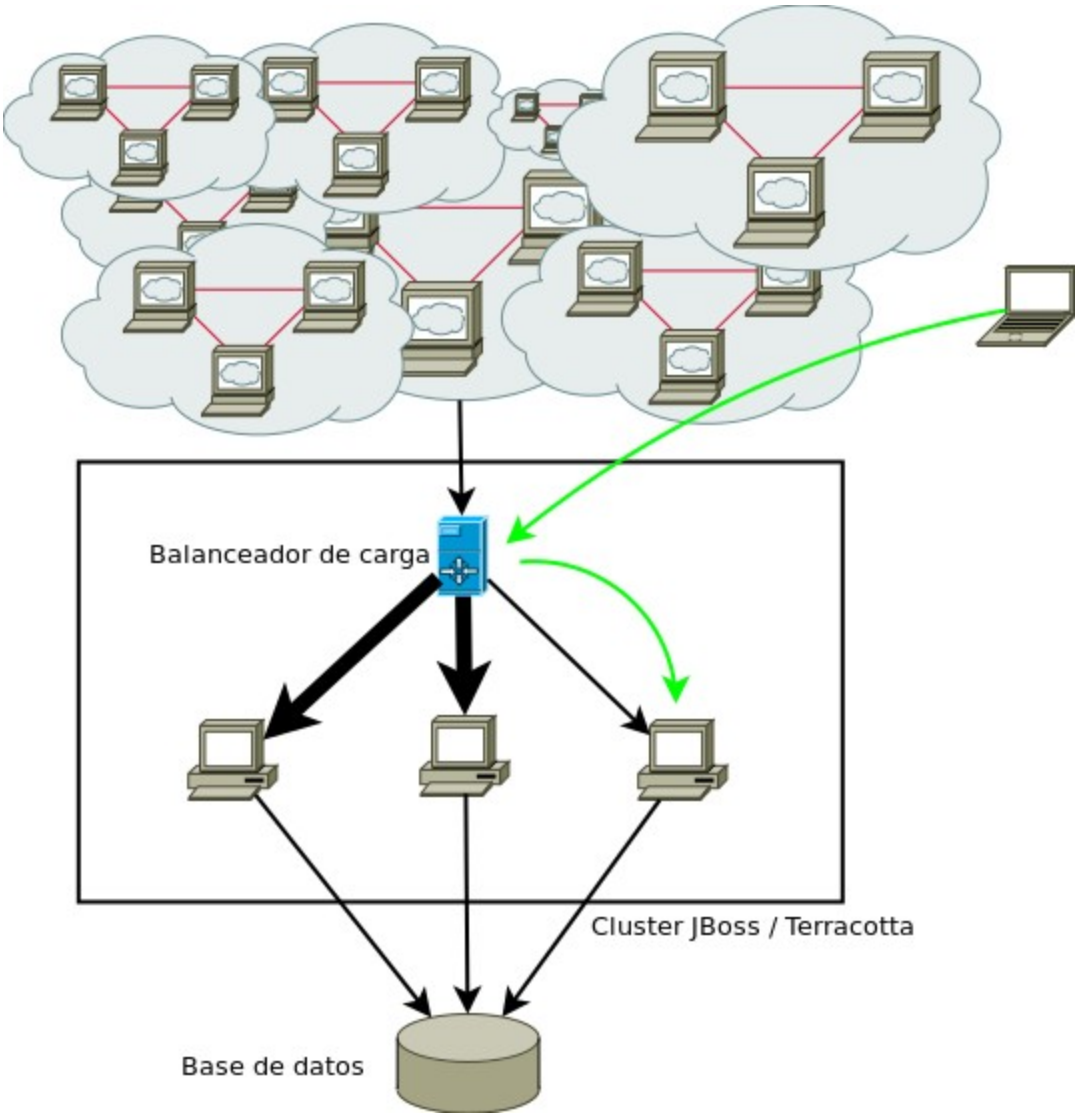
02.02.04.- Prueba general: Disponibilidad automática

Descripción		Disponibilidad incesante y automática
Clasificación		**Aspecto**
		Disponibilidad
Capa de ejecución	**CN**	**X**
Tipo de ejecución		**Manual**
Entrada		Escenario con N nodos al que se conecta un cliente
		Corte intencional de conexión de uno de los nodos
Salida		Conclusión respecto a la disponibilidad
Criterio de éxito		La aplicación siempre está disponible, aunque el cable que se retira sea del servidor preponderante en la configuración



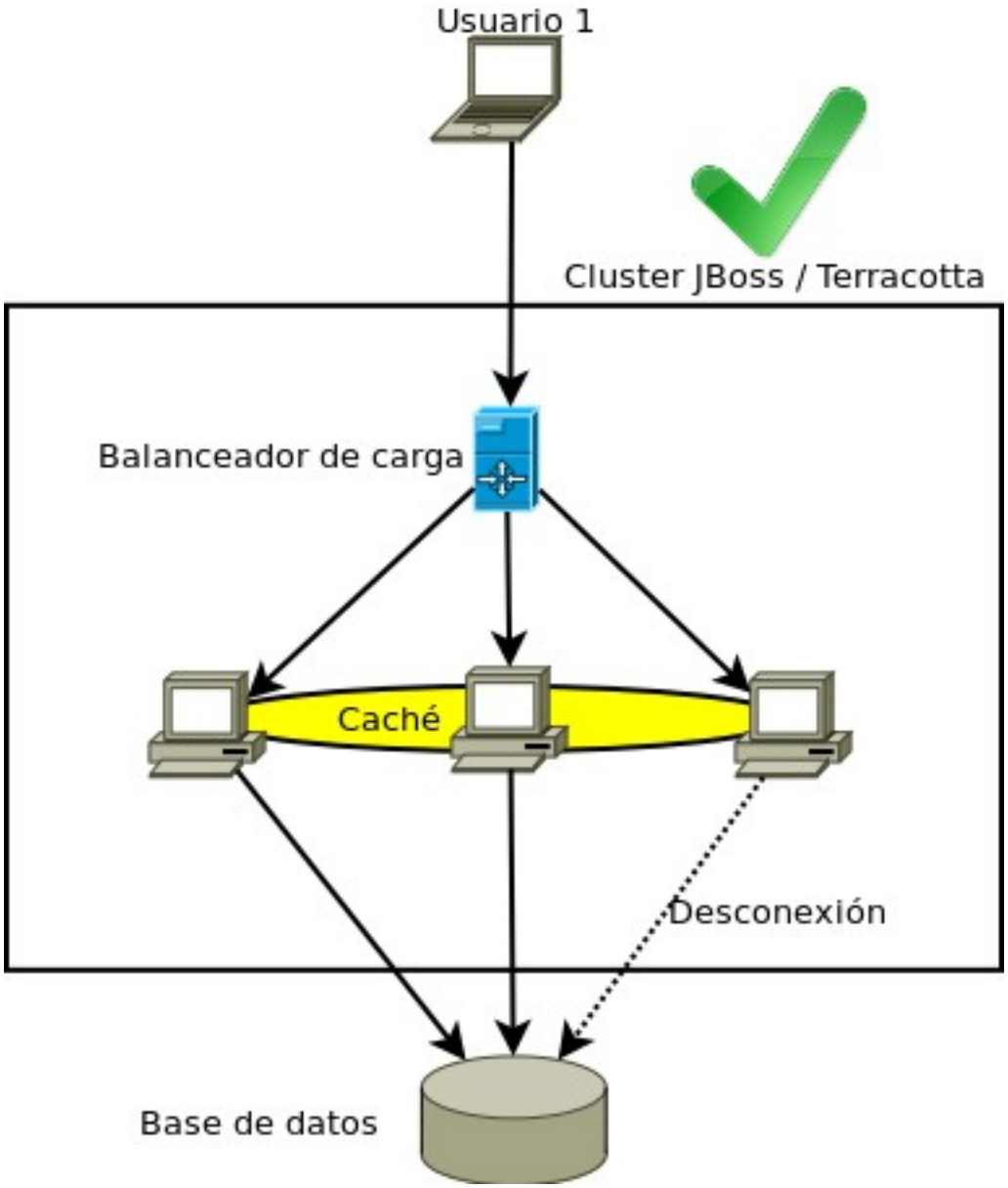
02.02.05.- Prueba general: Carga

Descripción		Evaluar balance de carga del sistema al ponerlo bajo la presión de cerrar la sesión que está usando el cliente
Clasificación		**Aspecto**
		Carga
Capa de ejecución	**CN**	**X**
Tipo de ejecución		**Automática**
Entrada		Escenario que represente un nivel de carga al menos 10 veces superior al nivel de carga normal
		Instancia con JMeter
Salida		Mediciones
Criterio de éxito		Mediciones que indican que responde el nodo con menos carga



02.02.06.- Prueba general: Caché

Descripción		Actualizar datos del usuario y verificar distribución del caché luego de desconectar la base de datos
Clasificación		**Aspecto**
		Distribución de cache
Capa de ejecución	**CN**	**X**
Tipo de ejecución		**Automática**
Entrada		Instancia de escenario normal; modificación sobre datos de un usuario, desconexión de la base de datos, y posterior consulta de los mismos desde otro nodo
Salida		Resultado en el segundo nodo
Criterio de éxito		El segundo nodo muestra los datos actualizados sin conexión a la base de datos



02.02.07.- Prueba general: Sesión

Descripción		Verificar que frente a un error en uno de los nodos, la sesión del usuario es administrada en otro nodo manteniendo la disponibilidad
Clasificación		**Aspecto**
		Disponibilidad
Capa de ejecución	**VC**	**X**
Tipo de ejecución		**Automática**
Entrada		Instancia de escenario normal; ingreso al sistema con un usuario, y posterior desconexión del nodo que lo atendió inicialmente, siendo que hay otros nodos disponibles
Salida		Resultado en otro nodo
Criterio de éxito		El otro nodo continúa administrando la sesión del usuario sin problemas