Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Arquitectura de Software

Obligatorio

Santiago Pérez – 170441

Martín Arzuaga – 172215

Docente: Pablo Arreche – Rodrigo Freire

Contenido

1. Introducción

Este documento presenta la arquitectura del sistema de la empresa AllNews, la cual se dedica a la publicación de artículos informativos.
La arquitectura del software de AllNews, es un conjunto de estructuras, elementos y relaciones, con el objetivo de cumplir con los requerimientos funcionales, los atributos de calidad y los requisitos.
Por último, para explicar la arquitectura nos basamos en vistas arquitectónicas, las cuales son una abstracción del sistema desde una perspectiva o punto de vista.

Este documento está enfocado hacia miembros de equipos de desarrollo de software que trabajan con este sistema así como otros interesados en el mismo.

1.1. Propósito

El propósito del presente documento es proveer una especificación completa de la arquitectura del sistema que desarrollará AllNews.

2. Antecedentes

2.1. Propósito del Sistema

El propósito de nuestro sistema es crear una aplicación Java EE que provea los servicios web para que dicha aplicación consuma. Además, permitirá que los lectores puedan acceder a los artículos, así como también que un usuario administrador pueda crear nuevos.

Cabe aclarar que el trabajo que se pretende especificar con este documento es el del desarrollo de la aplicación que correrá del lado del servidor, la cual proveerá los servicios necesarios para que la aplicación para Smartphone de AllNews funcione correctamente.

2.2. Requerimientos significativos de Arquitectura

Para la seguridad de la aplicación será necesario como mínimo que las contraseñas se guarden encriptados para una mayor seguridad de los usuarios.
Se prevé que AllNews incremente los tipos de publicaciones que realiza para así poder captar un mayor número de lectores.

2.2.1. Requerimientos funcionales

Los requerimientos funcionales se clasifican en tres grupos: Usuarios, Publicadores y Ambos. A continuación se muestran dichos requerimientos:

USUARIOS

Requerimiento	Descripción
Comentar un Post	Permite al usuario realizar un comentario sobre un Post existente
Seguir a un Publicador	El usuario podrá seguir a un publicador y de esta forma recibirá la información que éste publique.
Ver artículos de Publicadores	El usuario verá la información de todos los publicadores que sigue.

PUBLICADORES

Requerimiento	Descripción
Crear un Post	Crear un Post los cuales serán visibles para todos los usuarios
Actualizar un Post	Actualizar la información de un Post creado por el Publicador actual
Borrar un Post	Eliminar un Post creado por el Publicador actual
Bloquear un Usuario	Impide al usuario ingresar a la aplicación
Buscar información de un usuario	Ver la cantidad de Post, Seguidores y Datos personales.

AMBOS

Requerimiento	Descripción
Buscar un Post	Busca un Post del Sistema por el título ingresado
Ver lista de comentarios de un Post.	Permite visualizar la lista de comentarios que tiene un Post
Crear un Usuario	Crear un usuario nuevo para utilizar el sistema
Modificar su propio Usuario	Modificar los datos del usuario propio
Poder ver una lista con información resumida de cada post. Para así poder cargar más rápidamente la lista de artículos del usuario	Devuelve la siguiente información de un Post: idPost, título, positivos y negativos, usuario que lo creó y fecha de última modificación
Comentar un Post	Comentar un Post existente en el Sistema
Calificar un Post como Positivo o Negativo	Calificar un post existente en el Sistema. Un usuario podrá calificar una única vez un Post.
Eliminar un comentario	Elimina un comentario de una publicación
Obtener Posts con mejores calificaciones	Obtiene una lista de los Post mejor calificados

2.2.2. Requerimientos no funcionales

El sistema a implementar contará con distintos atributos de calidad los cuáles son:

1. **Modificabilidad**: El sistema a implementar busca ampliar los canales de difusión de los artículos informáticos de AllNews, pero para ello se decidió empezar implementando una mínima cantidad de las funcionalidades (tipos de publicaciones, usuarios, roles, etc). En caso de que la aplicación sea exitosa se prevé un gran aumento de funcionalidades y usuarios, por lo que es deseable que el sistema inicialmente se diseñe teniendo en consideración que podrá ser modificado en un futuro.

2. **Disponibilidad**: Este atributo de calidad se considera básico para proveer un servicio de calidad como el que brinda AllNews a sus clientes actualmente mediante su página web. Se desea que el sistema tenga la mayor disponibilidad posible.

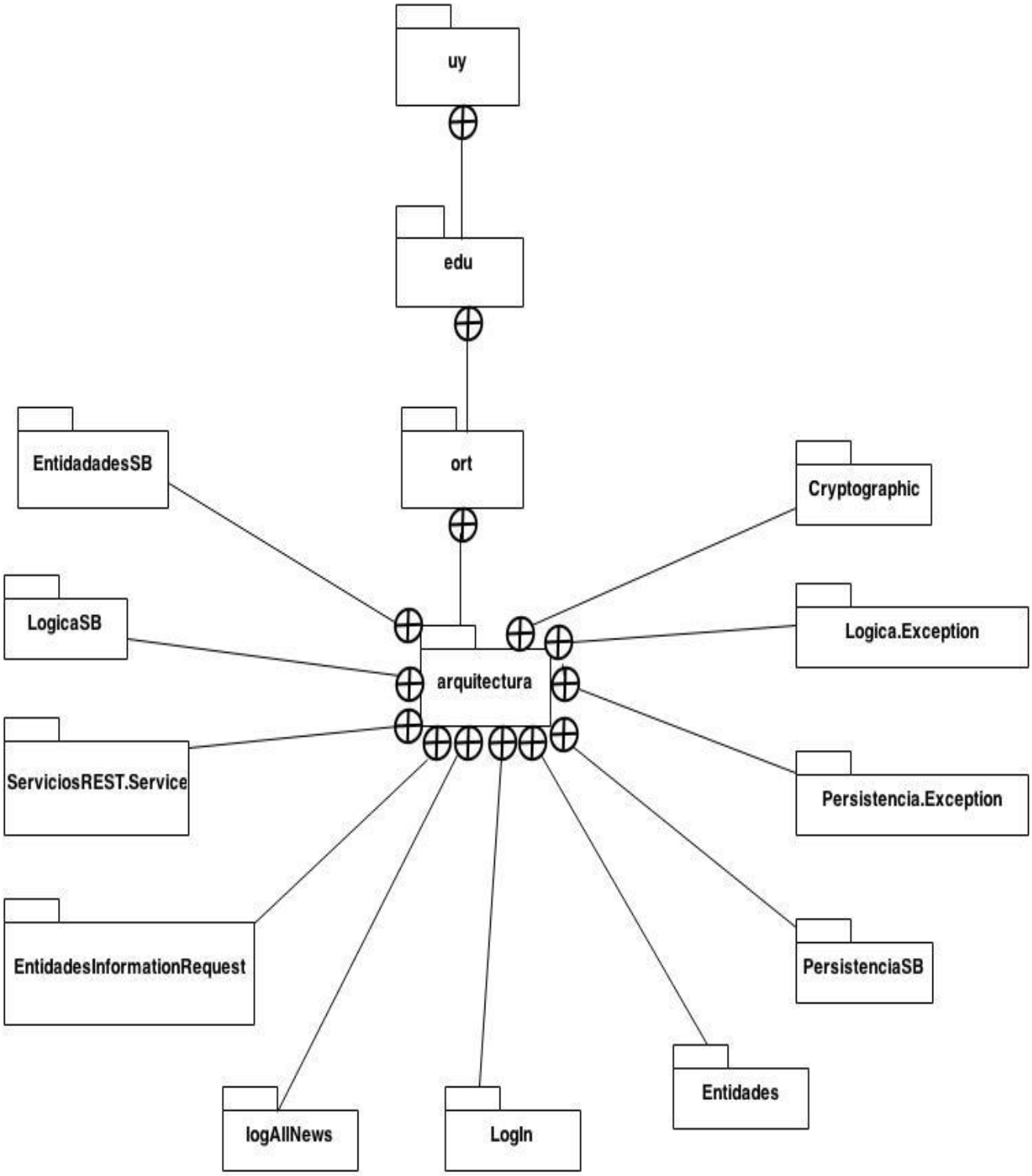
3. **Seguridad**: AllNews es consciente de la facilidad con la que hoy en día se puede corromper un sistema informático que no tenga las medidas básicas de seguridad. Se pretende que este proyecto cubra los principales riesgos informáticos desde el punto de vista de la seguridad. Estos aspectos son: confidencialidad, integridad, disponibilidad, etc.
4. **Usabilidad**: En caso de existir algún error por parte del sistema, se buscará que el impacto en los usuarios sea el mínimo posible.

3. Documentación de la Arquitectura

3.1. Vistas de Módulos

3.1.1. Vista de Descomposición

3.1.1.1. Presentación



3.1.1.2. Catálogo de Elementos

Elementos y Propiedades

Elemento	Descripción/Responsabilidades
EntidadesSB	Es un paquete que contiene un SessionBean local por cada entidad que se pueda persistir en la base de datos. Es el encargado de la entidad y es el que llama a los métodos de session bean de PersistenciaSB correspondientes para realizar las consultas que desee y así obtener la información que necesite.
LogicaSB	Es un paquete que contiene únicamente un SessionBean local, el cual se encarga de proveer las funciones más complejas en este componente. Funciona como fachada para un conjunto de funciones. Alguna de las funcionalidades que provee son: - agregar seguidor a un usuario - obtener los post recientes de los seguidores de un usuario
ServiciosRest.Service	Es el servicio REST, el cual se encarga de proveer las operaciones que se desean consumir desde otra aplicación. Permitiendo interoperabilidad con otras aplicaciones. Este servicio se encarga de consumir el elemento recibido (json,texto,etc.), realizar la llamada al componente necesario y devolver la respuesta correspondiente para cada caso.
Entidades	Es el paquete que contiene todas las entidades de la lógica de negocio necesarias que se pueden persistir. Estas entidades fueron creadas a partir de la base de datos, porque consideramos una mejor practica realizar primero el modelo entidad relación y a partir de éste hacer las entidades.
EntidadesInformationRequest	Contiene clases que son utilizadas como molde para cargar cierta información en un formato diferente al de las entidades y así poder retornar información más precisa.
LogicaSB.Exception	Este paquete, al igual que el paquete Persistencia.Excepcion se encargan de crear las excepciones correspondientes dependiendo del caso, pudiendo así diferenciar donde ocurrió el error.
PersistenciaSB.Exception	Idem que en Logica.Exception
PersistenciaSB	Es un paquete que contiene un SessionBean local por cada entidad que se pueda persistir en la base de datos. Cada uno de estos session bean extiende de Persistencia Abstracta que es quien implementa todos los métodos de consulta necesarios para almacenar su correspondiente en la base de datos así como también recupéralos.

Interfaces

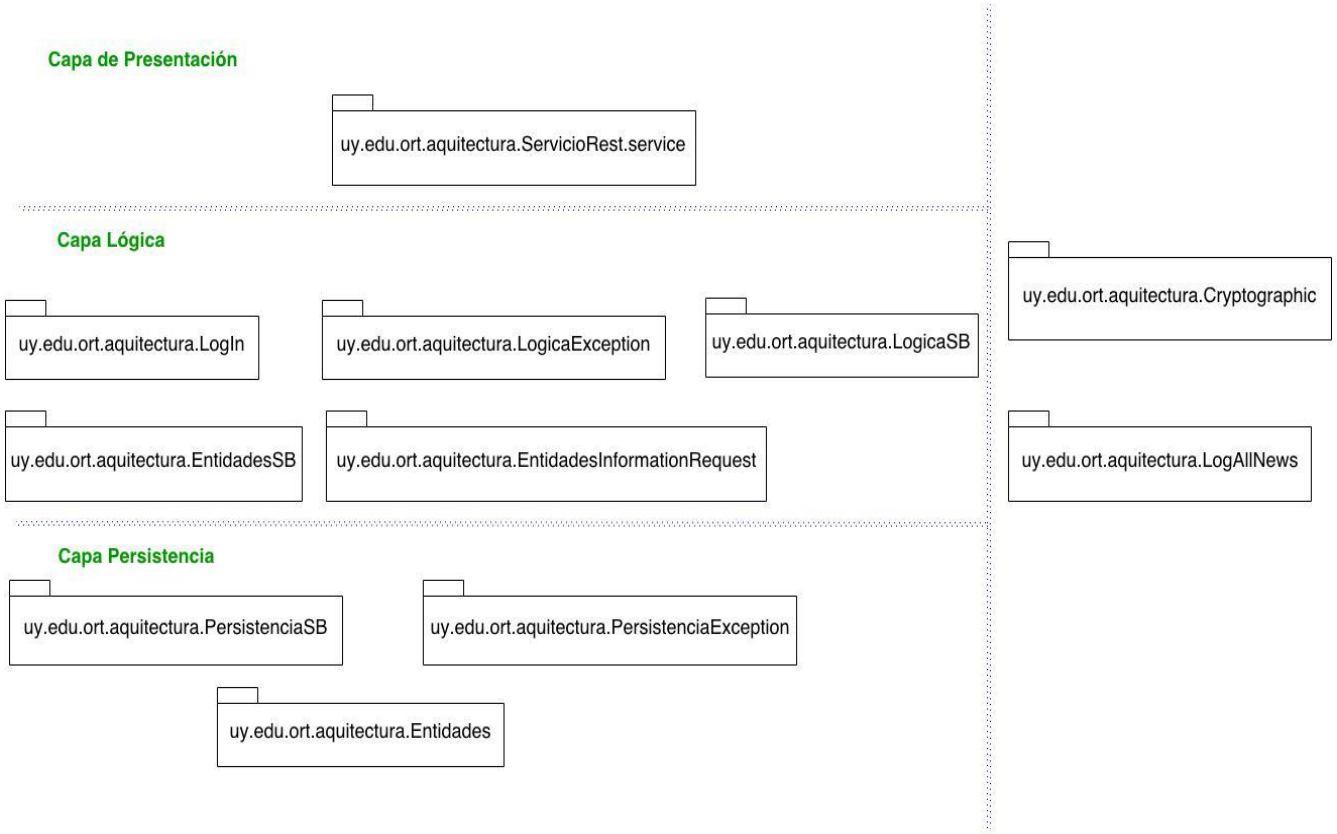
En esta vista no se describe ningún detalle referente a Interfaces.

3.1.2.3. Justificaciones de diseño

Como se ve en la representación primaria del diagrama de descomposición, nuestro software se encuentra dividido en varios paquetes. Esto permite que cada paquete se encargue de realizar únicamente una responsabilidad, permitiendo así que sea fácil de mantener y extender en un futuro. Justificación más detallada se extiende en vista de uso.

3.1.2. Vistas de Layers

3.1.2.1. Presentación



3.1.2.2. Catálogo de Elementos

Elementos y Propiedades

El catálogo de elementos para este caso es el mismo que para el diagrama de vista de descomposición.

Relaciones y sus propiedades

La tabla de vista de uso de paquetes explica las relaciones entre los paquetes presentados en esta sección.

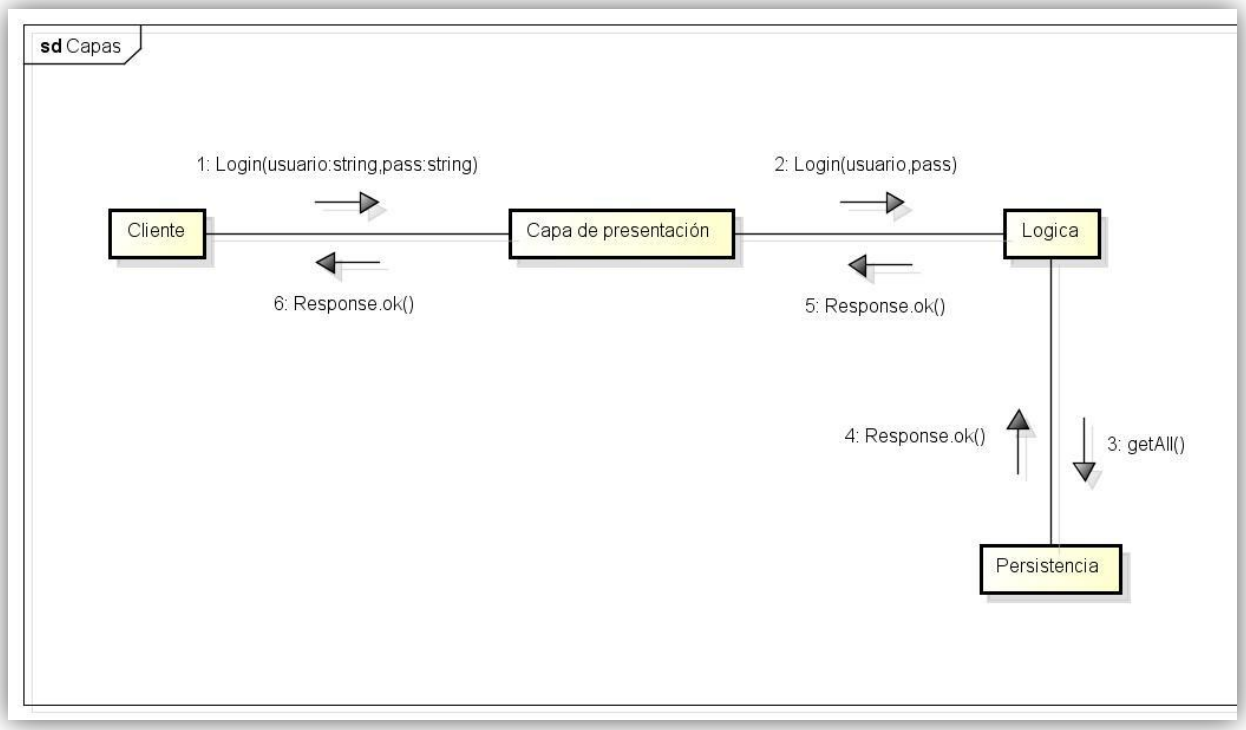
Interfaces

Los detalles correspondientes a las interfaces se describen en el apartado 3.2 de Vistas de Componentes y conectores.

Comportamiento

Login

A continuación se muestra el diagrama de colaboración del método Login mostrando cómo interactúan las distintas capas del Sistema.



3.1.2.3. Guías de Variabilidad

El modelo por capas nos permite, pensando a un posible crecimiento a futuro, poder distribuir las diferentes capas en distintos ordenadores y/o servidores, facilitando enormemente la escalabilidad del sistema.

Por ejemplo en un supuesto caso de que la capa de presentación se vea saturada, se podría tener más de un ordenador con dicha capa, pudiendo así dividir la carga entrante y/o saliente.

3.1.2.4. Justificaciones de diseño

Se decidió utilizar el patrón Layer debido a que dividir el sistema en capas bien marcadas permite una mayor modificabilidad, debido a que es más fácil de comprender su funcionamiento, y una mayor extensibilidad, debido a que la dependencia que tienen las capas entre si nos permiten reemplazar una de ellas sin impactar en capas que no sean adyacentes, así como poder modificar una de las capas sin que haya un impacto mayor o impredecible.

Cada capa tiene su propia responsabilidad, diferenciándola claramente de las demás. Como ya se comentó antes, las capas solo pueden comunicarse con sus adyacentes, Por ejemplo, la capa Lógica solo puede comunicarse con la capa de Persistencia y la capa de Presentación. Esto nos permite, sin mayores esfuerzos y entre otras cosas, poder implementar un sistema de excepciones en cada capa y que sean lanzadas hacia la capa de presentación, y que esta se encargue de realizar las acciones correspondientes y registrar los logs del sistema.

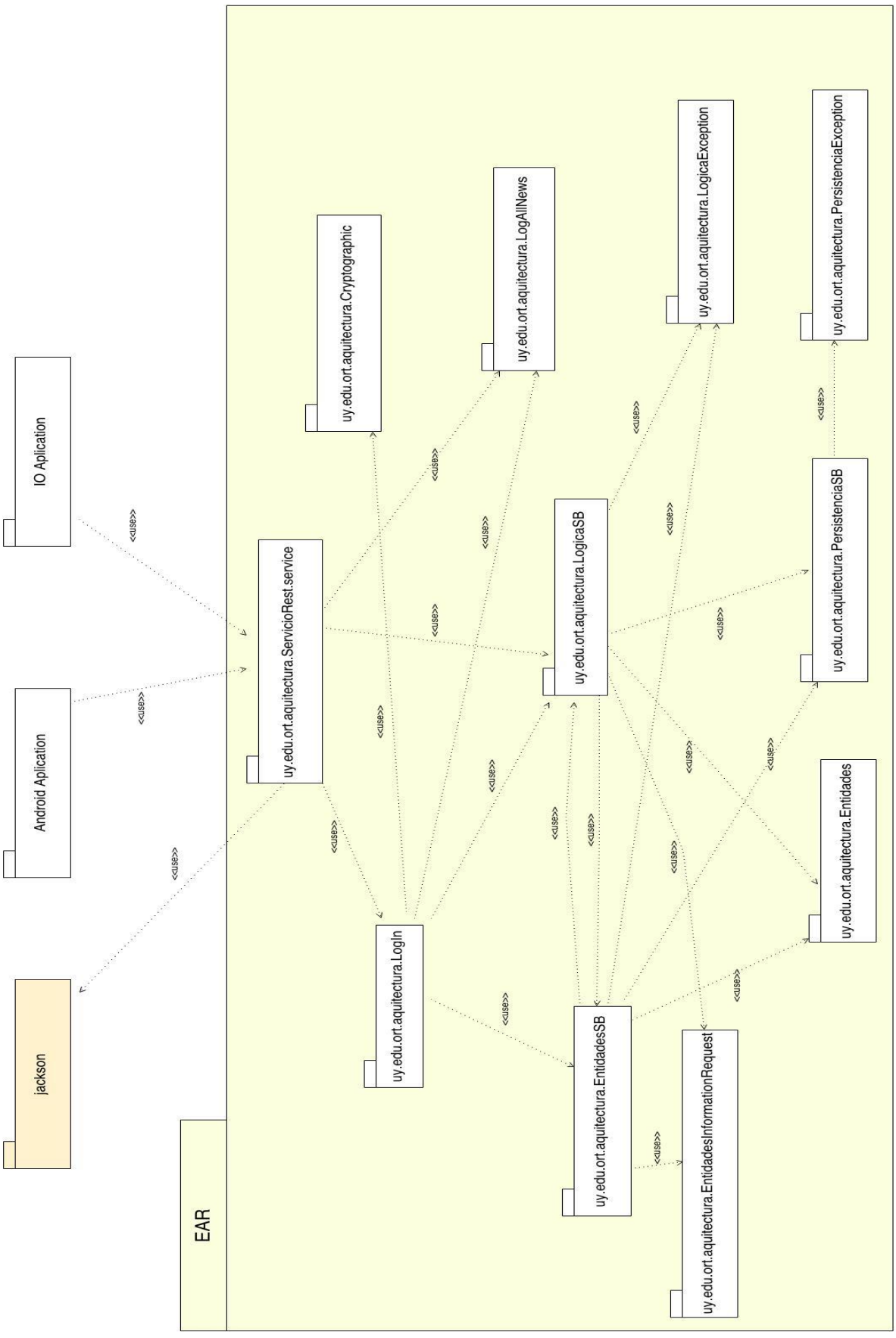
Otra de las ventajas por las que se eligió el patrón Layer es que el desarrollo puede paralelizarse, un desarrollo por cada capa, de forma de poder acelerar el proceso total de desarrollo o tercierizar alguna de las mismas a una empresa externa a AllNews. Solo basta con definir interfaces o API's que consumirán las capas entre sí para poder llevar a cabo lo antes dicho.

Siempre teniendo en consideración que se está hablando de una arquitectura para una aplicación que correrá en un servidor de aplicaciones y que pretende proveer servicios a dispositivos móviles es que el sistema fue dividido en 3 capas, una capa de Presentación, otra Lógica y una última de Persistencia.
La capa de Presentación es la que expone los servicios REST para que los clientes (aplicativos Mobile) puedan consumir. Cabe destacar que se decidió implementar los servicios siguiendo la arquitectura REST, esto es debido a que REST es, en comparación con otras arquitecturas como SOAP, mucho más sencillo, soporta muchos más formatos de datos, su documentación es más fácil de comprender, tiene mayor escalabilidad y rendimiento (teniendo en cuenta que nuestros clientes serán aplicaciones móviles el rendimiento juega un papel clave), entre otras ventajas frente a SOAP.

Se puede aclarar que hay paquetes que son y/o podrían ser consumidos desde varias capas. Los mismos proveen servicios independientemente de la solución del sistema por lo que se encuentran aislados de posibles cambios en el mismo. En el caso de esta arquitectura los paquetes son "Cryptographic" y "logAllNews" que se encargan de proveer servicios de criptografía (seguridad) y realización de logs respectivamente, para que el resto de las capas que así lo requieran los consuman.

3.1.3. Vistas de Uso

3.1.3.1. Presentación



3.1.3.2. Catálogo de Elementos

Relaciones y sus propiedades

Relación	Descripción/Relación
EntidadesSB - EntidadesInformationRequest	Utiliza este paquete para manejar información que no es posible manejar solamente con las entidades.
EntidadesSB - Entidades	Obtiene las entidades del Sistema para poder trabajar con ellas.
EntidadesSB - PersistenciaSB	Utiliza el paquete PersistenciaSB para persistir las entidades en la base de datos.
EntidadesSB - LogicaSB	Accede a las distintas operaciones de la lógica de negocio que involucran más de una entidad y las relaciones entre ellas.
EntidadesSB - LogicaException	En caso de error, utiliza el paquete LogicaException para el manejo de excepciones de la capa lógica.
PersistenciaSB - PersistenciaException	Si existe un error al persistir los datos, el paquete PersistenciaSB utiliza el paquete PersistenciaException para manejar la excepción en cuestión.
LogicaSB - PersistenciaSB	El paquete LogicaSB utiliza el paquete de Persistencia para almacenar los datos cuando alguno de sus métodos lo requiera.
LogicaSB - Entidades	Obtiene las entidades del Sistema para poder trabajar con ellas.
LogicaSB - EntidadesInformationRequest	Utiliza este paquete para manejar información que no es posible manejar solamente con las entidades. Específicamente para hacer él envió en formato Json hacia el cliente.
LogicaSB - EntidadesSB	Accede a las distintas operaciones que se pueden realizar sobre las entidades del Sistema para poder realizar operaciones que impliquen mayor lógica.
LogicaSB - LogicaException	En caso de error, utiliza el paquete correspondiente para el manejo de excepciones de las entidades del Sistema.
LogIn - LogicaSB	Accede a las distintas operaciones de la lógica de negocio que involucran más de una entidad y las relaciones entre ellas.
LogIn - EntidadesSB	Utiliza la entidad Usuario del paquete EntidadesSB.
LogIn - LogAllNews	Utiliza el paquete LogAllNews para crear un log cuando el usuario ingresa al sistema.
LogIn - Cryptographic	Para manejar de forma segura los datos de los usuarios, se utiliza el paquete Cryptographic para la criptación.
ServicioRest.service - LogIn	Utiliza el paquete LogIn para el manejo de Logueo de los usuarios que utilizan el Sistema.

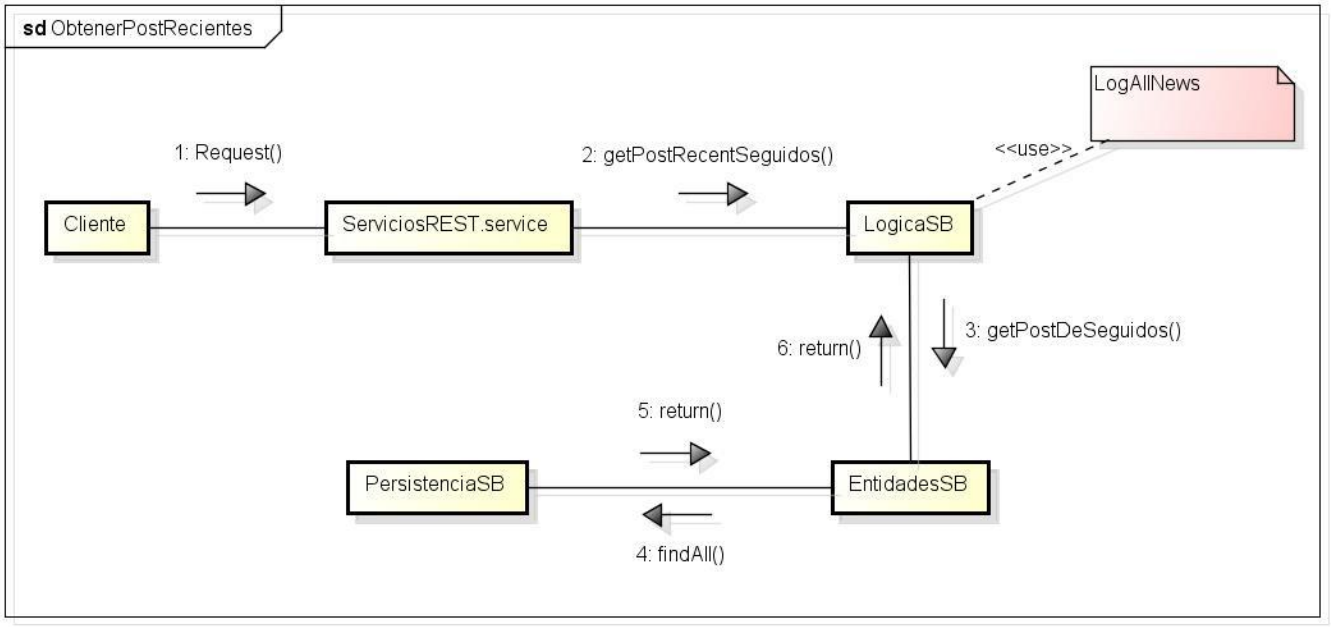
ServicioRest.service - LogAllNews	Utiliza el paquete LogAllNews para crear un log cuando el usuario ingresa al sistema.
ServicioRest.service - LogicaSB	Accede a las distintas operaciones de la lógica de negocio que involucran más de una entidad y las relaciones entre ellas.
Android Aplication - ServicioRest.service	Accede a los servicios que ofrece la aplicación.
IO Aplication - ServicioRest.service	Accede a los servicios que ofrece la aplicación.

Interfaces

Los detalles correspondientes a las interfaces se describen en el apartado 3.2 de Vistas de Componentes y conectores.

Comportamiento (diagramas de secuencia, colaboración, estado)

A continuación se presenta el diagrama de colaboración que muestra cómo obtener los Post reciente de los seguidores.



3.1.3.3. Guías de Variabilidad

La arquitectura fue pensada teniendo en consideración que la propia AllNews sería la que llevaría a cabo el proyecto, sabiendo que el equipo de IT de la misma no supera las 10 personas, lo que limita algunas posibles características de la arquitectura.

A pesar de lo dicho antes su modelo en capas permite la fácil modificabilidad de cada una de las mismas, pudiendo extenderlas, cambiarlas o incluso realizarles refactoring para optimizar o mejorar algún aspecto en particular.

3.1.3.3. Justificaciones de diseño

Paquete EntidadesInformationRequest fue creado debido a que el aplicativo en muchas ocasiones tenía la necesidad de devolver información en formato JSON hacia

el cliente que no se mapeaba con ninguna de las clases existentes en el sistema, y se consideró que la forma más sencilla de poder almacenar la información que se desea retornar, para luego convertirla a Json, era creando un paquete que contuviera las clases que representarían dicha información.

La librería Jackson fue la elegida para la conversión/desconversion de entidades a formato Json para poder ser enviadas hacia los clientes. Existe una gran variedad de librerías que proveen dicho servicio, la decisión de utilizar Jackson fue debido a que posee características que la diferencias de otras, como por ejemplo su performance, lo que os ayuda a poder cumplir con el atributo de calidad deseado, que no depende de otras librerías, lo que vuelve a ayudar con la performance y bajo acoplamiento. Jackson también es open source, lo que nos permitiría solucionar problemas que surjan y/o adaptarlo a nuestras necesidades.

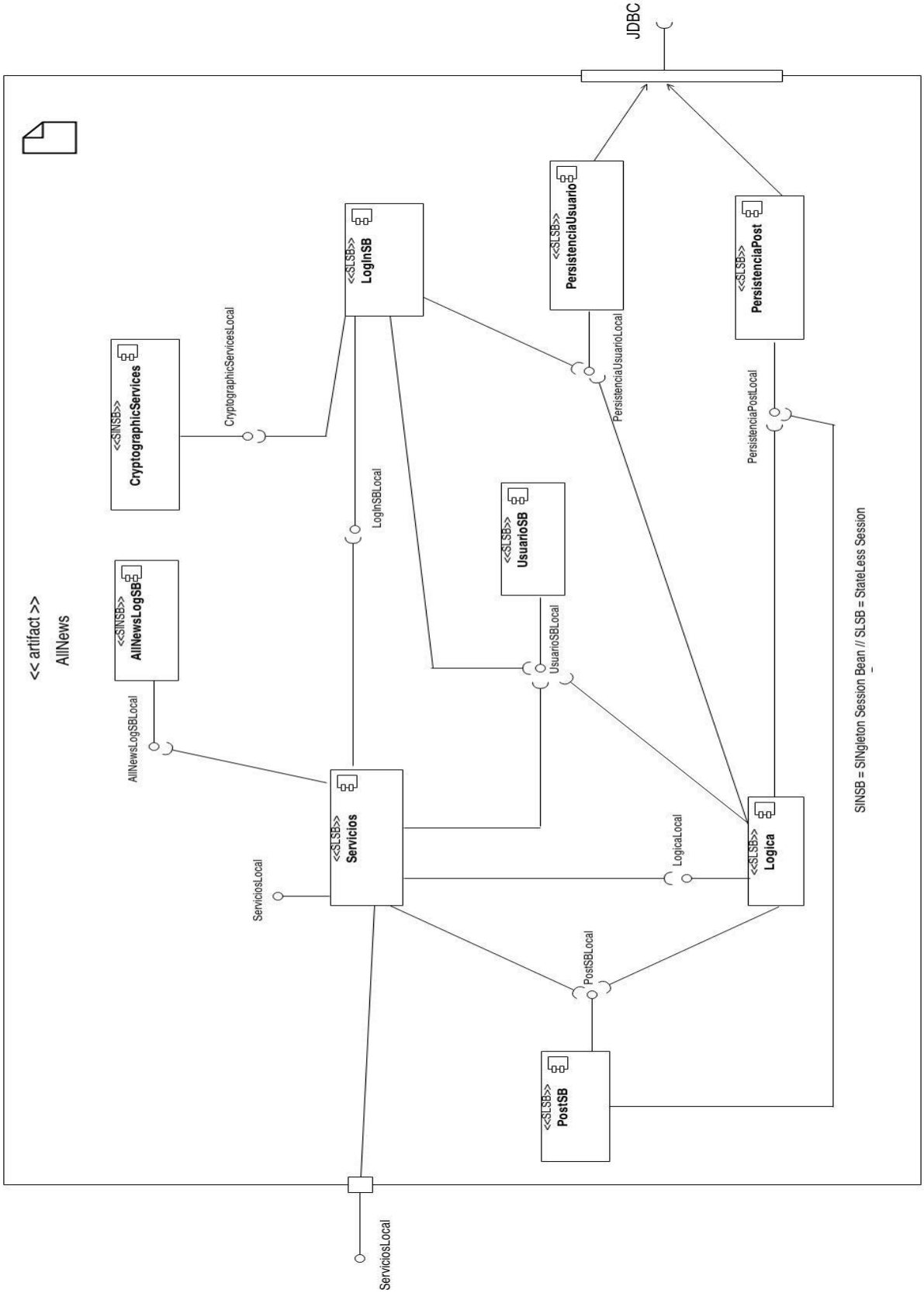
Gracias a la utilización del patrón Layers es que podemos implementar, sin mayores dificultades, un sistema de excepciones. En el cual cada capa posee una excepción diferente, lo que nos permite identificar de dónde provino la excepción y tener lo más claro posible donde ocurrió el error. Es una funcionalidad, que en conjunto con el registro de logs, nos permite poder tener un buen control sobre las acciones y errores que suceden en nuestro aplicativo. Cada vez que sucede un error la excepción es mandada hacia la capa superior y así consecutivamente hasta que la capa lógica o de presentación, dependerá del caso, se encargue de registrar el log correspondiente.

Existe un paquete destinado a la seguridad del sistema y arquitectura, el mismo en un principio provee únicamente servicios de criptografía, pero se lo independizo y se trata de que no dependa de otras librerías para que el mismo en un futuro pueda agregar a sus funcionalidades la encriptación y/o algoritmos más complejos de seguridad para el envió de información por internet (arquitectura cliente-servidor).

3.2. Vistas de Componentes y conectores

3.2.1. Vista de Diagrama de componentes y conectores

3.2.1.1. Presentación



ACLARACIÓN: Falta agregar algunos SLSB y SINSB, pero consideramos que los mostrados son una buena representación del funcionamiento que define la Arquitectura.

3.2.1.2. Catálogo de Elementos

Elementos y Propiedades

Elemento	Descripción/Responsabilidades
PostSB	Provee todas las operaciones relacionadas a los Post, como puede ser: crear, actualizar, borrar, agregar una nueva calificación, obtener los seguidores de un post, buscar un Post, entre otros.
Servicios	Tiene todos los servicios que contiene la aplicación
Lógica	Contiene toda la lógica involucrada con los Seguidores, Publicadores y Comentarios. Permite agregar o quitar un nuevo seguidor, obtener la cantidad de publicaciones o comentarios y obtener los Post que recién fueron seguidos.
AllNewsLogSB	Permite la inserción de un nuevo Log en un archivo txt.
CryptographicServices	Ofrece los servicios para el manejo de la criptación utilizando MD5.
LogInSB	Tiene los métodos relacionados con el login del Sistema.
UsuarioSB	Provee todas las operaciones relacionadas a los Usuarios, las cuales permiten crear un usuario nuevo, actualizar su información, obtener su rol, eliminar un usuario, obtener los seguidores o la lista de usuarios que lo siguen y obtener la información de un usuario, entre otros.
PersistenciaUsuario	Expone los métodos relacionado con la persistencia de los usuarios, los cuáles pueden ser: create, edit, borrar u obtener su rol, entre otros.
PersistenciaPost	Los mismos métodos que el caso anterior pero para Post.

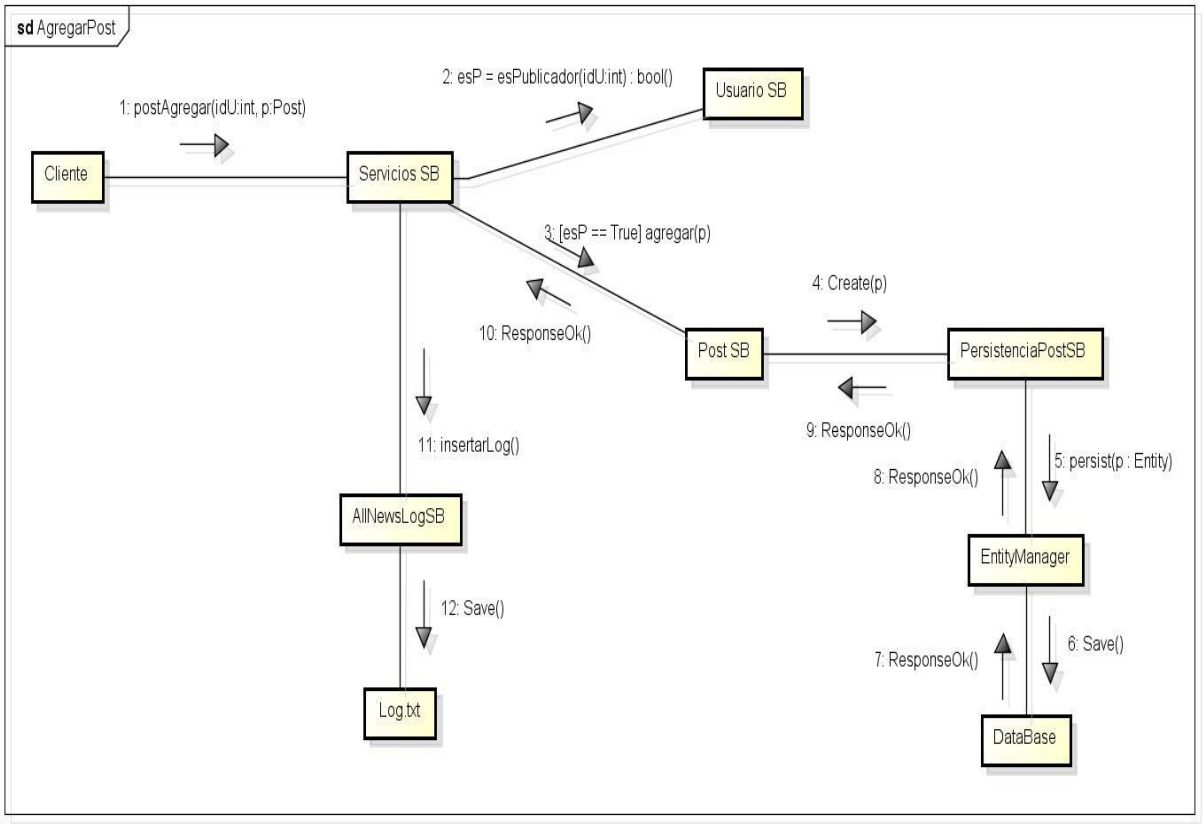
Interfaces

Interfaz	Componente que la provee	Componente que la utiliza	Descripción
ServicioLocal	Servicios		Expone los distintos servicios web.
PostSBLocal	PostSB	Logica - Servicios	Expone todos los métodos para manejar los Post del Sistema
LogicaLocal	Logica	Servicios	Expone servicios que permiten obtener información del negocio, normalmente involucran varias entidades y búsquedas
UsurioSBLocal	UsuarioSB	Logica - Servicios - LogInSB	Expone todos los métodos para manejar los Usuarios del

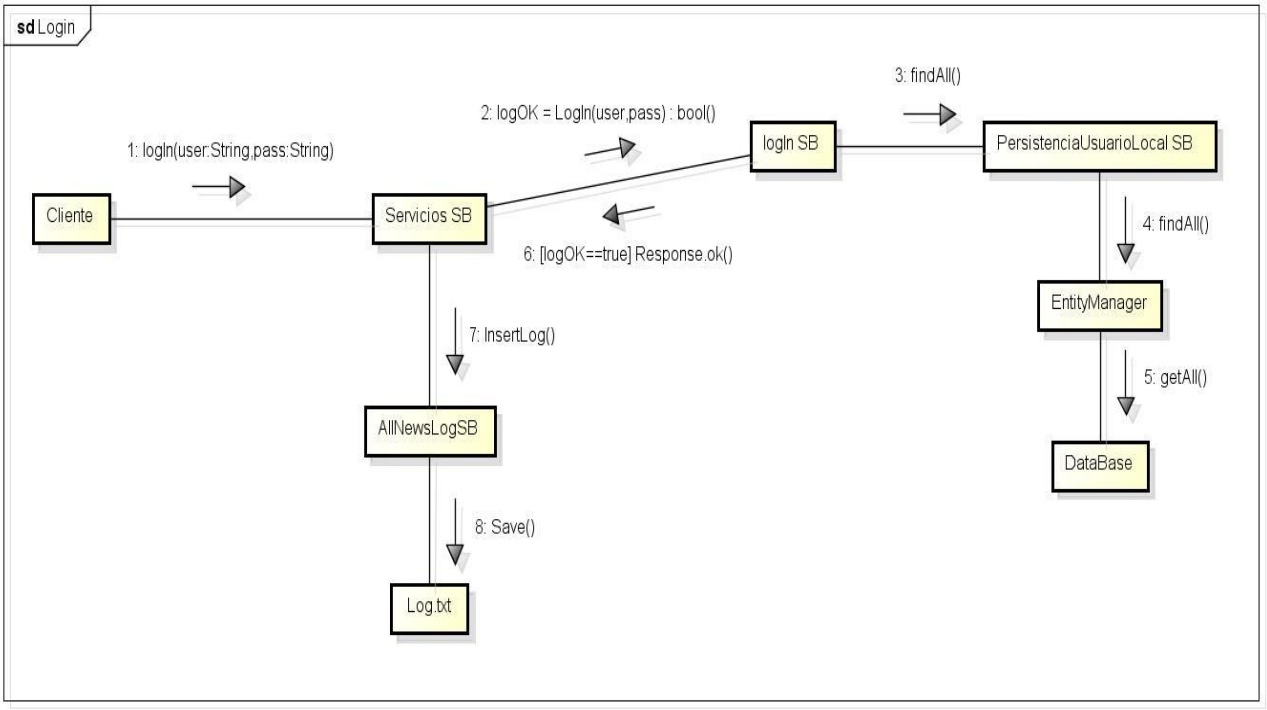
			Sistema
PersistenciaPostLocal	PersistenciaPost	Logica - PostSB	Expone las operaciones referentes a la persistencia de los Post.
PersistenciaUsuarioLocal	PersistenciaUsuario	Logica - LogInSB	Expone las operaciones referentes a la persistencia de los Usuarios.
LogInSBLocal	LogInSB	Servicios	Expone servicios necesarios para realizar el log in de usuarios al sistema.
CryptographicServicesLocal	CryptographicServices	LogInSB	Expone las operaciones referentes a la criptacion utilizando MD5
AllNewsLogSBLocal	AllNewsLogSB	Servicios	Expone los servicios necesarios para insertar logs al sistema.

Comportamiento

Agregar Post



Login



3.2.1.3. Guías de Variabilidad

Una de las ventajas de los Session Beans de Java es que son fácilmente acoplables a otros sistemas, por lo que el día de mañana el aplicativo podría dejar de utilizar las implementaciones propias de registro de logs y excepciones y/o acoplarse a nuevos EJB que provean servicios acordes a las necesidades que podría tener en un futuro AllNews.

Por ejemplo en un supuesto caso de escalabilidad se podría dejar de usar la solución de excepciones y/o registro de logs, provista en esta arquitectura y desarrollada posteriormente por AllNews, para pasar a usar soluciones más robustas y complejas de terceros, soluciones que son inviables que AllNews implemente por su cuenta.

Otro posible caso de expansión podría ser que la lógica del aplicativo se vuelva lo suficientemente compleja que el componente actual LogicaSB tenga que acoplarse a otros componentes para lograr eficientemente y ordenadamente llevar a cabo los pedidos de los clientes.

3.2.1.4. Justificaciones de diseño

AllNews está empezando a expandirse en el mundo Mobile y tecnológico, muchas de las decisiones tomadas prevén que AllNews en un futuro lleve a cabo otros proyectos que puedan potencialmente ser construidos utilizando la tecnología de Java EE.

Para el requerimiento de seguridad la arquitectura prevé un componente independiente que sea el encargado de proveer servicios de criptografía. Para así no almacenar contraseñas de usuarios en texto plano, una de las primeras decisiones que se pueden tomar si queremos considerar la seguridad de la arquitectura.

El componente AllNewsLogSB se define con el objetivo de proveer una solución propia y a medida de registro de logs del sistema creada y diseñada por AllNews. Esto se debe a que el aplicativo no tiene mayores complejidades y una solución a medida a solución de logs le permitiría a AllNews ahorrar económicamente en una solución de terceros e igualmente cumplir con sus necesidades actuales y a mediano plazo.
Se busca que el componente sea independiente y no dependa de otras librerías que no estén incluidas en las por defecto en Java.

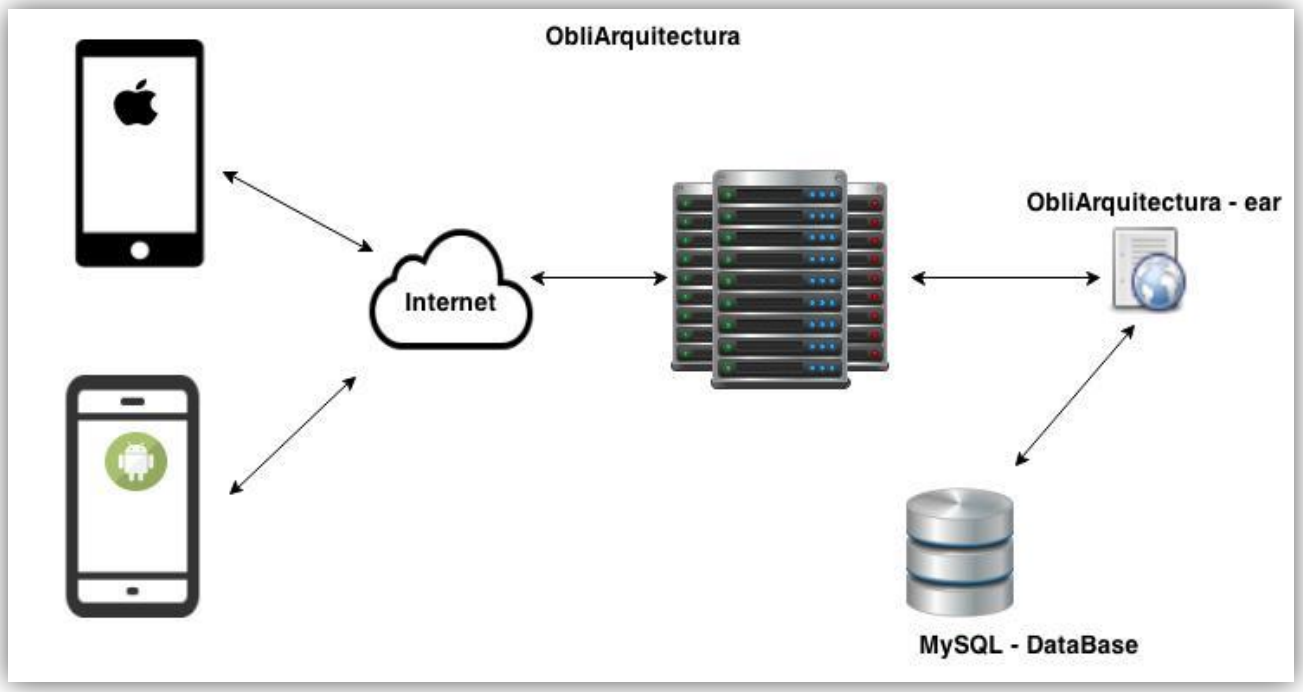
La arquitectura define la creación de una gran cantidad de Session Beans para separar las responsabilidades de la mejor manera posible y dividir de esta forma el sistema. Esto favorece la modificabilidad pero empeora la eficiencia (overhead). De igual manera se han definido todos los Session Beans de forma local para maximizar la performance en su comunicación.

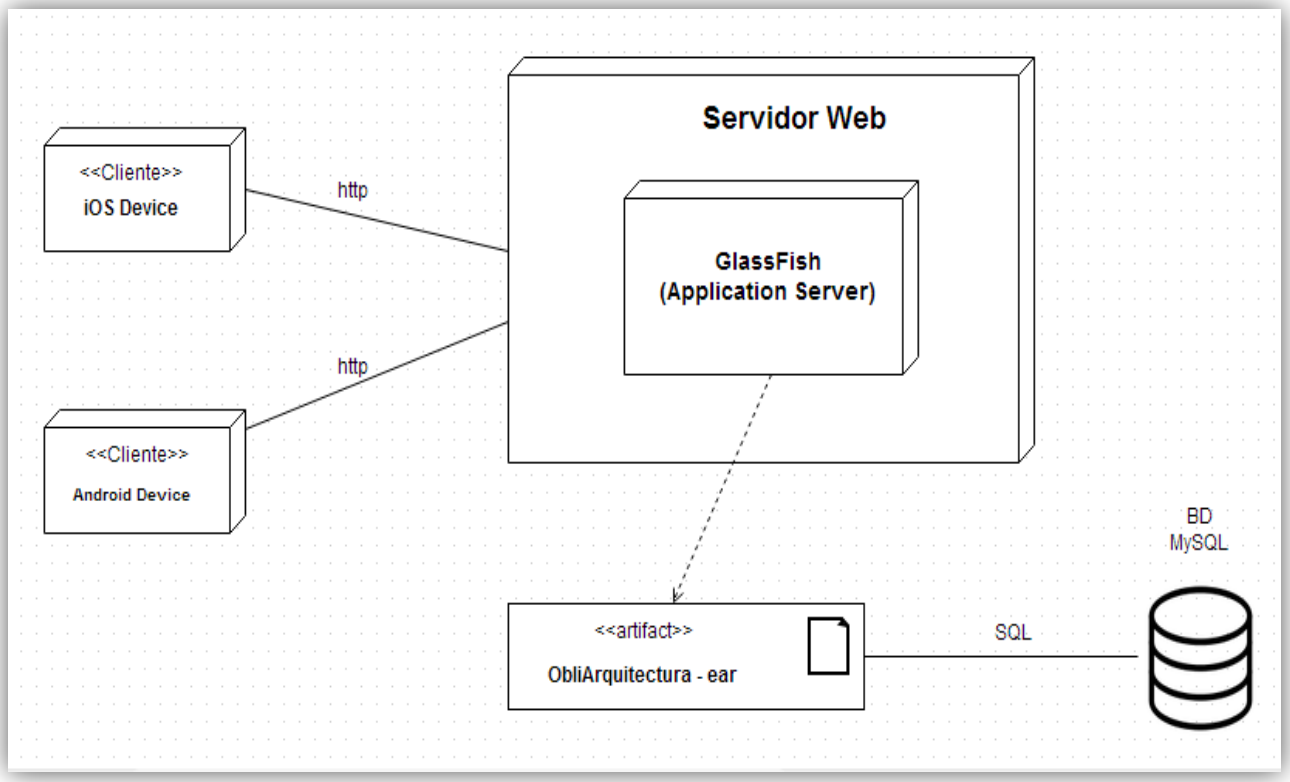
La persistencia del sistema está definida mediante alguna implementación de la API de java JPA, la cual se acopla perfectamente a la base de datos seleccionada (MySQL) y nos brinda modificabilidad debido a que puede acoplarse a casi cualquier motor de base de datos popular.

3.3. Vistas de Asignación

3.3.1. Vista de Despliegue

3.3.1.1. Presentación





3.3.1.2. Catálogo de Elementos

Elementos y Propiedades

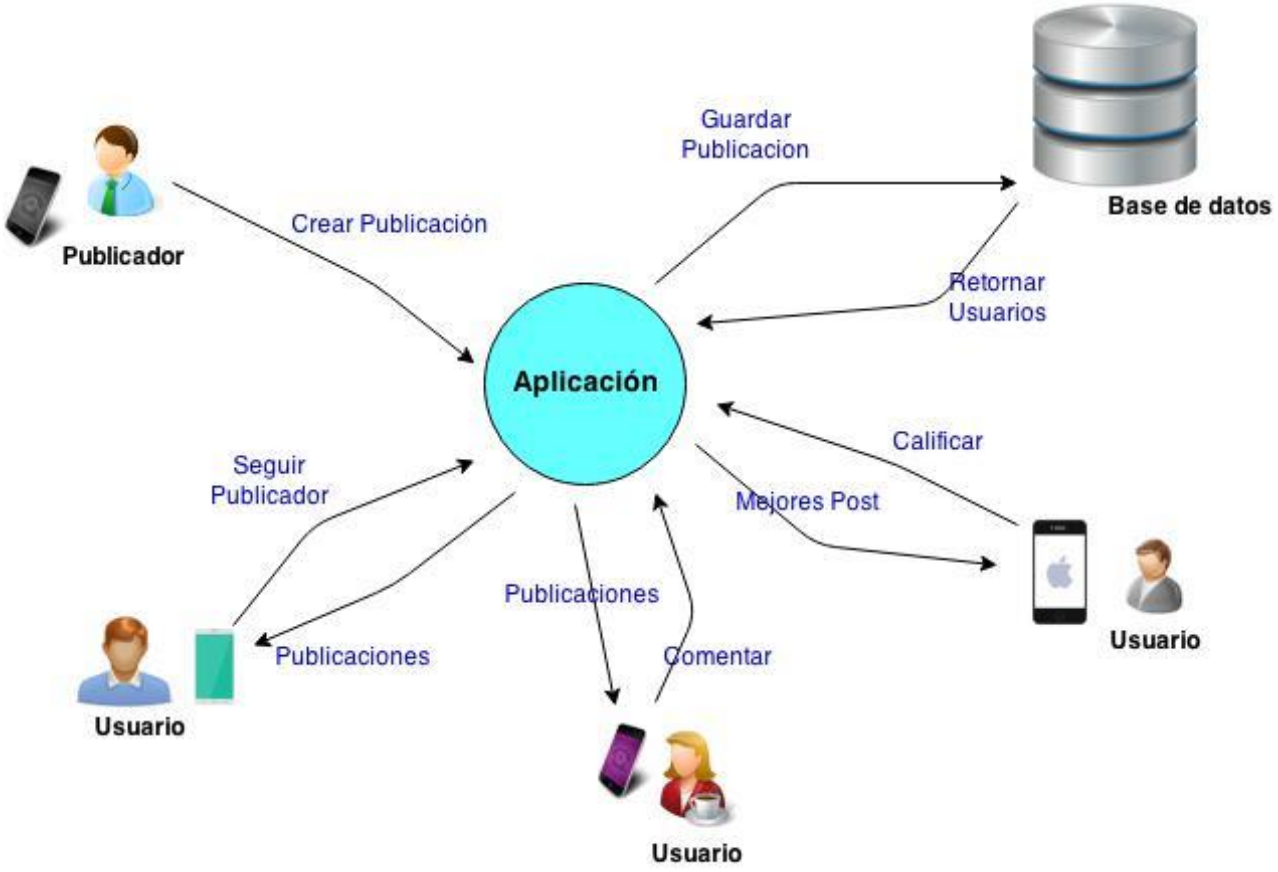
Elemento	Descripción/Responsabilidades
iOS Device	Cliente que interactúa con el Sistema.
Android Device	Cliente que interactúa con el Sistema.
Servidor Web	El servidor web es el servidor en donde está corriendo el GlassFish.
GlassFish	Es el servidor de aplicaciones
ObligArquitectura -ear	Es el ear es donde se empaqueta los distintos componentes de la aplicación.
http	Es el protocolo de comunicación entre el cliente y el servidor web.
BD Mysql	Protocolo de comunicación entre el servidor web y el servidor de base de datos

Relaciones y sus propiedades

Relación	Descripción
Cliente – Servidor	Los clientes se comunican con el servidor utilizando el protocolo http.
Servidor – Aplicativo ear	El aplicativo se ejecuta en el Servidor

Aplicativo ear – Base de datos	La aplicación persiste su información en la base de datos.

3.3.1.3. Diagrama de contexto



3.3.1.4. Guías de Variabilidad

El despliegue del aplicativo será realizado, en un principio, mayoritariamente en un mismo ordenador (servidor web, servidor de aplicaciones, base de datos, etc), esto escapa al diseño de esta arquitectura y está dado por las restricciones que la empresa posee.

Independientemente de lo anterior, y gracias a las tecnologías utilizadas (REST, Java EE, entre otras), el despliegue en un futuro podría escalar, permitiendo un mayor número de conexiones entras y salientes, así como una mayor cantidad de procesamiento en simultaneo. Un posible ejemplo sería que existieran 2 servidores corriendo el aplicativo, que se conecten a una misma base de datos e implementar un sistema de distribución de carga para que las transacciones sean más rápidas y a su vez en caso de haber un problema con uno de los servidores el otro podría suplirlo.

Esta aplicación fue pensada para dispositivos móviles, por lo que su despliegue está basado en este hecho, independientemente no limita que la arquitectura posea una gran variedad de clientes diferentes en un futuro (en parte gracias a la flexibilidad de los servicios REST).

3.3.1.5. Justificaciones de diseño

El motivo de la arquitectura de despliegue esta dado, en gran parte, por requerimientos no funcionales y restricciones económicas o de tiempos que son mayores al proyecto en sí.

A pesar de eso, se decidió maximizar la escalabilidad horizontal, utilizando Java EE y sus componentes EJB que permiten una gran escalabilidad en este sentido, debido a que permiten la ejecución de varias tareas de procesamiento al mismo tiempo, a diferencia por ejemplo de PHP. Por lo que podemos crear aplicaciones que se ejecutan en un servidor y pueden manejar múltiples conexiones entrantes y salientes al mismo tiempo.