Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2173 Arquitectura de Sistemas de Software

Entrega 3: NewsRoom

Profesora: Rosa Alarcón
Chief Architect: Hans Findel
Team: J. Francisco Caiceo
Diego Carey
Pablo Caviedes
Santiago Larraín
Andrea Vásquez

REQUISITOS FUNCIONALES

Prioridad Alta:
- El sistema debe permitir CRUD+L noticias.
- El sistema debe poder suscribirse a RSS .
- El sistema debe poder suscribirse a Atom.
- El sistema debe realizar Polling a bases de datos externas cada una hora.
- El sistema debe permitir clasificar noticias en un conjunto predefinido de categorías.
- El sistema debe permitir al editor *taggear* noticias con un texto libre.
- El sistema debe permitir a los editores calificar noticias con una nota de 1 a 5.
- El sistema debe publicar noticias cuando esta tiene una calificación sobre un valor.
- El sistema debe soportar la creación de roles con sus restricciones. Estos roles son: periodista, editor, editor jefe de área y editor jefe por país.
- El sistema debe proveer una API.
- El sistema debe proveer una plataforma de acceso para escritorio y dispositivos móviles.

Prioridad Baja:
- El sistema debe permitir administradores con capacidad de configuración de permisos y creación de nuevos usuarios.
- El sistema debe permitir configurar y administrar del esquema de la base de datos.
- El sistema debe proveer una auditoría sobre CRUD de usuarios, noticias y noticieros.

REQUISITOS NO FUNCIONALES

- Alta escalabilidad: (riesgo alto, prioridad alta)
140 periodistas/paises x 15 paises = 2.100 periodistas
5 agencias/paises x 15 países x 15 periodístas/agencia = 1.125 periodistas
11 editores/paises x 15 países = 165 editores
El sistema debe soportar en el peor caso 3.390 conexiones simultáneas (para empezar). En el caso más probable, debe soportar 1.695 conexiones simultáneas (la mitad).

8 noticias/(hora x editor) x 8 horas/día x 165 editores = 10.560 noticias/día
10.560 noticias/día x 365 días/año = 3.854.400 noticias/año
El sistema tiene una restricción de 10MB por noticia, así que en el peor de los casos en 1 año se generará 38.7 TB de información aproximadamente.

- Interoperabilidad de infraestructuras tecnológicas: (riesgo medio, prioridad alta)
El sistema debe operar a través de plataformas propietarias o a través de una interfaz Web simple de newsRoom. El sistema debe facilitar el acceso a sus servicios de manera que a futuro pueda surgir un ecosistema de aplicaciones para NewsRoom. Este RNF está relacionado al RF de proveer una API.

- Alta disponibilidad: (riesgo alto, prioridad alta)
El sistema debe estar en línea y los datos deben ser accesibles para los editores 24x7x365.

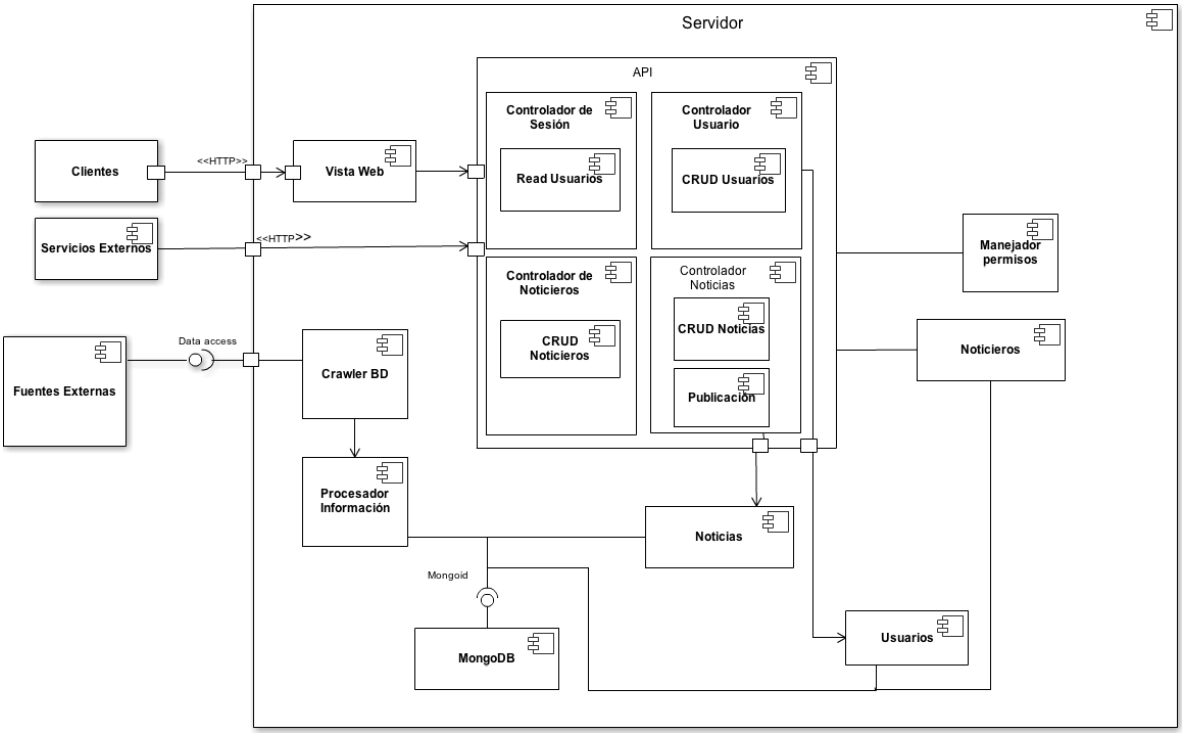
- Usabilidad: (riesgo bajo, prioridad baja)
El sistema debe proveer una interfaz Web simple a los periodistas, editores y agencias de noticias. Además, el sistema incluirá una interfaz simple e intuitiva para el lector. Este es un requisito de baja prioridad, porque el equipo no cuenta con diseñadores que puedan medir lo que debiera ser “simple”.

- Configurabilidad (riesgo bajo, prioridad baja)
El sistema debe proveer, para los administradores, la capacidad de configurar permisos y la creación/eliminación de nuevos usuarios.

ÁRBOL DE DECISIÓN

Característica	Sub-característica	Escenario
Alta Escalabilidad **(riesgo alto, prioridad alta)**	Cantidad de conexiones concurrentes	El sistema debe soportar hasta 3390 conexiones simultáneas
	Cantidad de solicitudes (requests)	El sistema debe soportar al menos la revisión de 10560 noticias al día por parte de los editores
	Cantidad de visitas diarias	El sistema debe soportar al menos 250000 visitas diarias (*ap.org tiene entre 200000-300000*)
	Tamaño de los datos	El sistema debe soportar noticias y archivos con un peso de hasta 10 MB
Interoperabilidad **(riesgo medio, prioridad alta)**	A nivel funcional	El sistema debe proveer una API que permita consumir sus noticias por parte de aplicaciones de terceros
	Consumo servicios externos	El sistema debe obtener noticias de terceros vía subscripción RSS. Por ejemplo de Reuters, Newsknowledge y New York Times
		El sistema debe revisar bases de datos de APIs externas y extraer sus noticias cada una hora. Por ejemplo , de sitios como este: http://blog.programmableweb.com/2012/02/01/81-news-apis-digg-fanfeedr-and-clearforest/
Alta Disponibilidad **(riesgo alto, prioridad alta)**	Porcentaje de tiempo en que la aplicación puede ser usada	La aplicación debe estar operativa 24x7x365
	Recuperación ante fallas	La aplicación debe recuperar automáticamente sus servicios en menos de 10 minutos en caso de falla

DIAGRAMA DE COMPONENTES



- API: Es el componente que se encarga de controlar y gestionar la aplicación. También provee un punto único de acceso para los servicios externos (Web API) y para las aplicaciones web (Vista HTTP).

- Controlador de usuario: Componente encargado controlar las acciones del usuario.

- CRUD Usuarios: Componente encargado de implementar las acciones de crear, leer, actualizar y borrar usuarios.

- Controlador de sesión: Componente encargado de mantener un estado de persistencia durante la sesión de un usuario.

- Read Usuarios: Componente encargado de verificar al usuario y su contraseña.

- Controlador de noticias: Componente encargado de controlar las noticias de la aplicación. Tiene las tareas de taggear y clasificar las noticias.

- CRUD Noticias: Componente encargado de implementar las acciones de crear, leer, actualizar y borrar noticias.

- Publicación: Componente que se encarga de ver la puntuación de las noticias y en caso de superar el puntaje de publicación son publicadas a todos los usuarios-lectores.

- Controlador de noticieros: Componente encargado de la administración de los noticieros.

- CRUD Noticieros: Componente encargado de implementar las acciones de crear, leer, actualizar y borrar noticias.

- Vista Web: Componente encargado de desplegar la información visualmente al lector.

- Crawler BD: Componente encargado de obtener información desde fuentes externas.

- Glutton, utiliza el Crawler para recoger los feeds desde fuentes externas y entregarselas al procesador de información.

- Procesador Información: Componente encargado de "traducir" la información obtenida a partir de las fuentes externas.

- Noticias: Representa la cada noticia, la cual puede ser taggeadas, clasificadas, calificadas y editadas. Facilita la comunicación con la Base de Datos. Además, proveen la lógica y las restricciones asociadas a las noticias. Tiene un componente para comunicarse con la base de datos.

- Noticieros: Representa las fuentes a la cual la aplicación se suscribe para leer noticias. Facilita la comunicación con la Base de Datos. Además, proveen la lógica y las restricciones asociadas a los noticieros. Tiene un componente para comunicarse con la base de datos.

- Usuarios: Componente que facilita la comunicación con la Base de Datos. Además, proveen la lógica y las restricciones asociadas a los usuarios. Tiene un componente para comunicarse con la base de datos.

- Manejador de la base de datos es un componente que se puede agregar a cualquier clase o modelo. Esto le da la facultad al modelo de poder acceder a todos los documentos de ese tipo, guardar nuevos, editarlos o eliminarlos. Es un conector que implementa una variedad de métodos para conectarse a la base de datos.

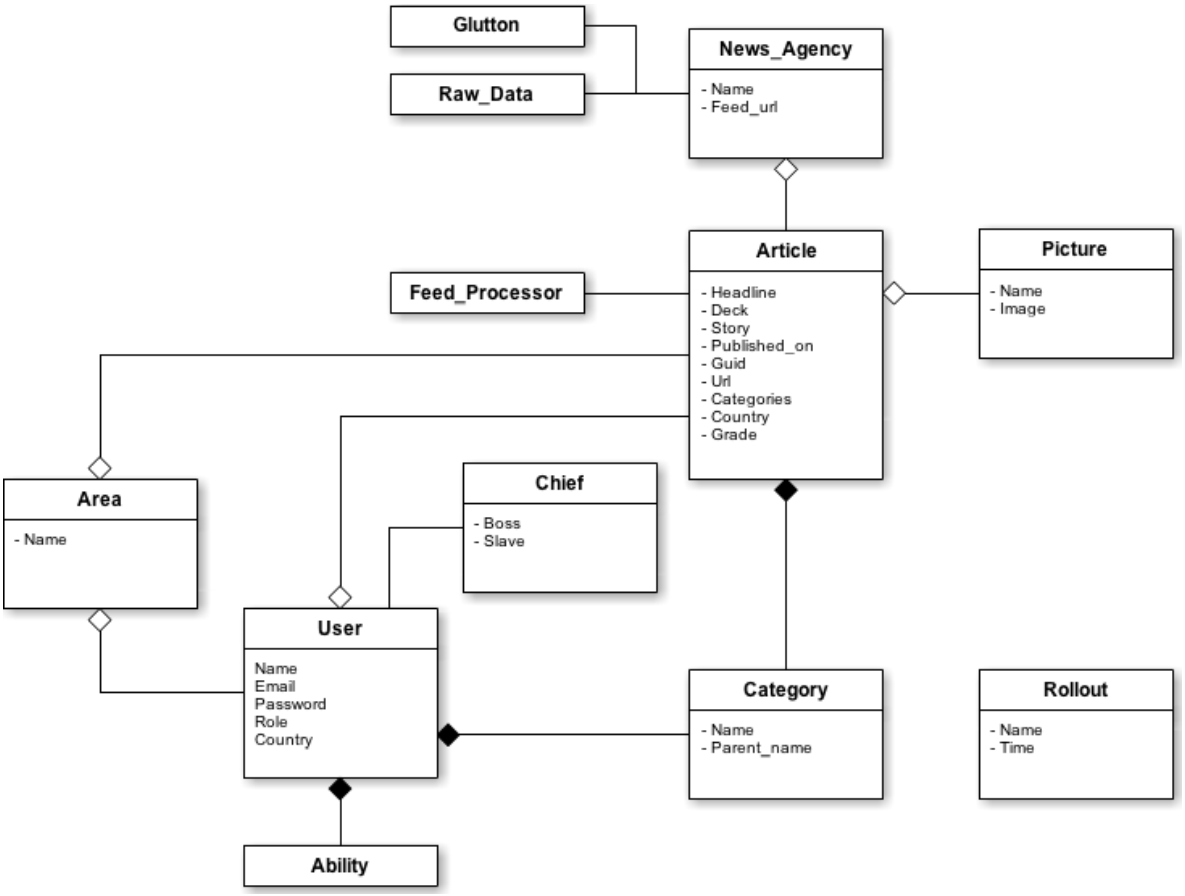
- Manejador de permisos: Componente encargado de la configuración de permisos y el manejo de éstos.

- Fuentes externas: Componente que representa agencias de noticias externas de las cuales se obtiene información.

- MongoDB: Base de datos NoSQL del sistema. Mantiene el estado persistente de los usuarios y noticias del sistema.

No se implementa un Pub/Sub exactamente, ya que no se produce una notificación sino que se va a buscar los feeds a las Fuentes Externas cada cierto intervalos de tiempo.

DIAGRAMA DE CLASES



En el diagrama anterior se puede ver la relación de las clases en el sistema. Si bien uno solo accede a vistas relacionadas a usuarios, agencias de noticias (news_agencies) y artículos, por debajo hay más información que está siendo almacenada.

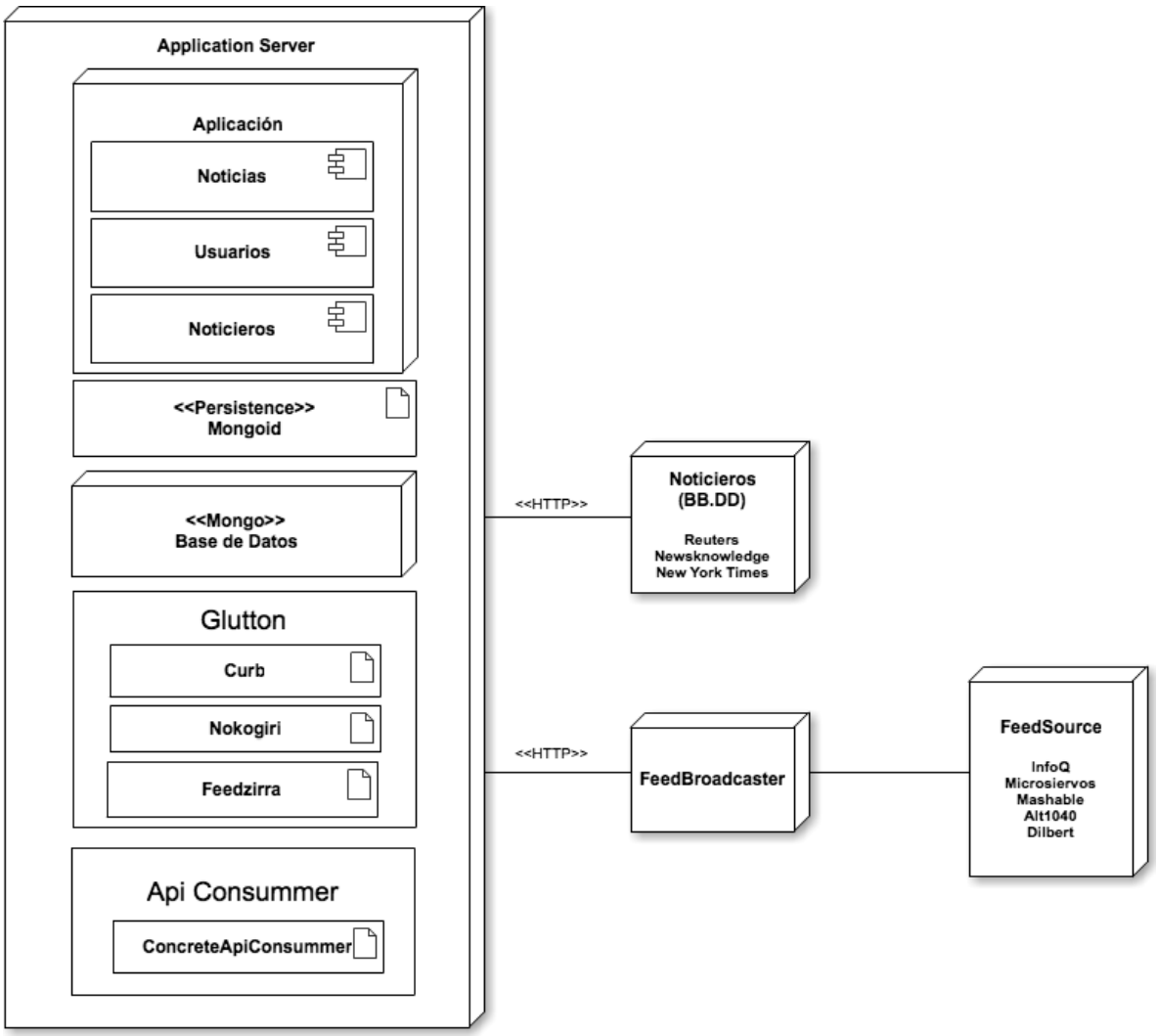
Dentro del diagrama anterior es importante destacar la importancia de algunos modelos que no están respaldados en base de datos como lo son Ability, Glutton, Raw_Data y FeedProcessor. En el diagrama se puede ver que no tienen atributos, pero proveen la lógica de negocio para dar los permisos necesarios al usuario, revisar los noticieros, almacenar la información de las fuentes externas y procesarlas respectivamente.

Vale mencionar también la importancia de Rollout. Este es un registro aparte del sistema, este componente registra los errores de las acciones requeridas por los clientes. Es responsable de decidir cuales *"features"* no están disponibles temporalmente.

DIAGRAMA DE DEPLOYMENT

A un nivel externo de la aplicación, vale destacar que debe conectarse tanto con noticieros externos así como con fuentes que publican sus contenidos a medida que los publican. Dada la existencia de una entidad de noticieros en la aplicación, es posible agregar dinámicamente a los feeders. Sin embargo, no es posible por ahora implementar un equivalente a lo anterior para los noticieros externos. Esto se debe a que las solicitudes deben ser realizadas a través de una api implementada para cada uno de esos servicios.

En particular, la aplicación estará suscrita a Reuters, Newsknowledge y el New York Times. Adicionalmente podrían ser agregadas otras fuentes que pueden ser encontradas en la url: http://blog.programmableweb.com/2012/02/01/81-news-apis-igg-fanfeedr-and-clearforest/.

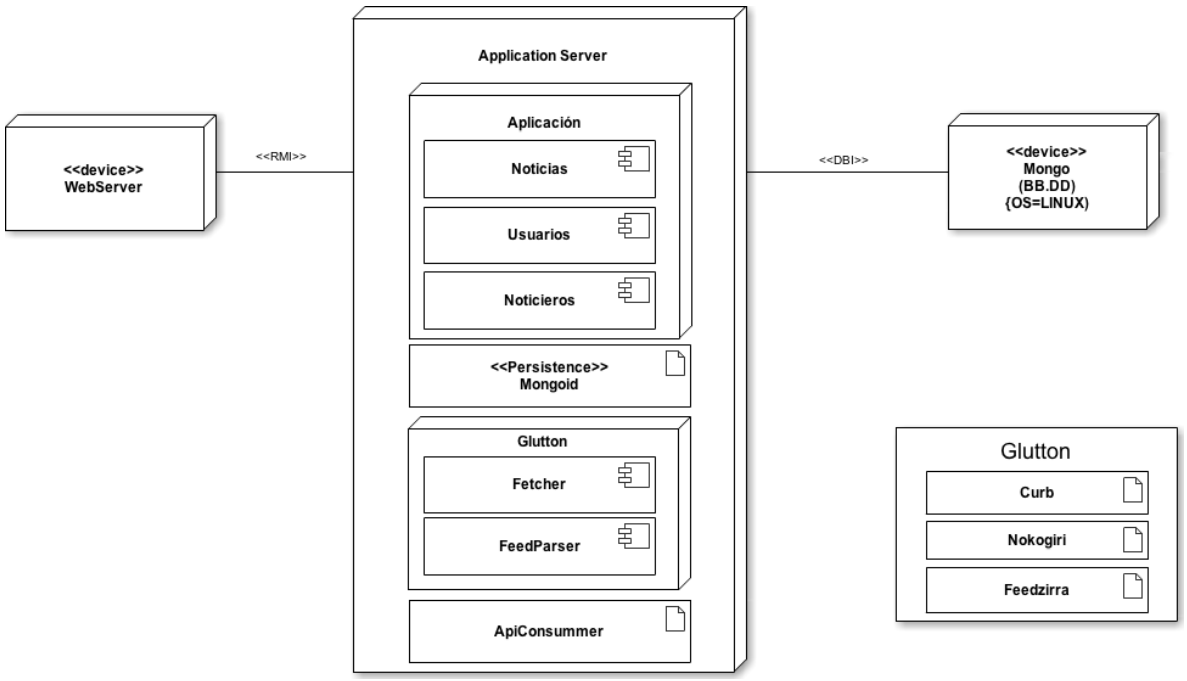


Descripción de componentes de deployment

- Servidor web: donde estará montada la aplicación.
- Base de datos NoSQL (Mongo): pieza de software separada de la aplicación.
- Servidor de aplicación: es donde se encontrará gran parte de nuestro desarrollo a lo largo del semestre. Cuenta con un sistema que permite a usuarios crear, leer y editar noticias para luego proveerlas al público en general.

La aplicación se podrá suscribir a otros noticieros a través de feeds o a través de las API's provistas por estos.

- Glutton: está constituido por tres librerías. La primera, Curb, para facilitar las conexiones con servidores, Nokogiri para parsear eficientemente los xml recibidos y Feedzirra para terminar de manejar correctamente los elementos. Con esto el elemento es suficientemente flexible como para alojar los cúmulos de feed en alguna carpeta temporal para su posterior procesamiento.



Análisis de Riesgos

Condición	Atributos de Calidad	Consecuencia	Probabilidad (baja, media, alta)	Impacto (bajo, medio, alto)	Mitigación	Contingencia
Capacidad de almacenamiento insuficiente	Eficiencia	Error al intentar subir nuevas noticias	Alta	Alto	Aumento de capacidad cuando se alcance cierto uso	Añadir capacidad de almacenamiento
Clausura de Fuentes Externas.	Confiabilidad	Incapacidad de descargar nuevas noticias	Baja	Alta	Revisar periódicamente las fuentes externas.	Buscar otras fuentes externas.
Cambio de interfaces de Fuentes Externas.	Confiabilidad	Incapacidad de descargar nuevas noticias	Baja	Alta	Revisar periódicamente las fuentes externas.	Adecuar lo que se recibe con lo que se necesita.
Sobrecarga de consultas por parte de los servicios externos.	Disponibilidad	Incapacidad de responder a todas las peticiones de los servicios externos.	Media	Alto	Hacer test de carga y compararlo con las fuentes externas suscritas.	Replicar la API en varios servidores para soportar la carga.
Sobrecarga de consultas por parte de los clientes.	Disponibilidad	Incapacidad de responder a todas las peticiones de los clientes.	Media	Alta	Hacer test de carga y compararlo con los usuarios inscritos en el sistema.	Replicar la API en varios servidores para soportar la carga.
Falta de conocimientos técnicos por parte de algunos integrantes		Demora en el desarrollo del trabajo	Media	Alto	Programación en pares con quienes más conocen	Dejar de lado funcionalidades más complejas o intentar lograrlas de manera más simple.
Indisponibilidad de uno de los miembros del equipo		Retraso en el desarrollo.	Baja.	Media	Tener reuniones semanales para programar cada semana y analizar la disponibilidad de cada miembro del equipo.	Distribuir la carga de trabajo del integrante ausente en el resto del equipo.

Indisponibilidad del servidor de deployment	Disponibilidad	Retraso en la instalación y release.	Media	Alta	Levantar el servidor con anticipación a la fecha de término del desarrollo.	Aumentar las horas hombre en el levantamiento del servidor.

Puntos sensibles:

1. Alta escalabilidad: El uso de una base de datos no-relacional de documentos permite el almacenaje de grandes cantidades de datos.
2. Interoperabilidad: Crear una API como acceso de servicios externos a la aplicación permite que varias aplicaciones operen con la aplicación.
3. Alta disponibilidad: El componente servidor tiene un sistema de auto recuperación de máximo 10 minutos.
4. Usabilidad: La creación de una interfaz gráfica amigable y que sea muy intuitiva nos permitirá lograr una buena usabilidad.
5. Configurabilidad: Mediante el componente Manejador de Permisos se logrará asignarle a cada uno de los usuarios los distintos niveles de acceso que tendrán y las opciones que podrán configurar.

Trade - offs:

1. Para aumentar la escalabilidad y el desempeño tenemos que replicar las bases de datos y el componente API lo cual disminuye la mantenibilidad.
2. Usar bases de datos no-relacionales aumenta la escalabilidad y desempeño pero disminuye la confiabilidad.
3. Al hacer ping cada una hora al momento de obtener las noticias aumenta la eficiencia pero disminuye la confiabilidad de la información (las noticias no están en tiempo real).
4. Al hacer fat ping la aplicación es más eficiente para el usuario pero usa más espacio en nuestras bases de datos, perdiendo gestión de recursos.
5. Para aumentar la seguridad se usan permisos y login de usuarios, pero se sacrifica usabilidad para el usuario.

DISTRIBUCIÓN DE TAREAS PARA ESTA ENTREGA.

Pablo Caviedes, Hans Findel: Testing

Hans Findel, Diego Carey, Pablo Caviedes: Corrección del informe

Santiago Larraín: Instalar la aplicación en un servidor de la universidad

José Francisco Caiceo, Andrea Vasquez, Diego Carey, Santiago Larrain, Hans Findel: Desarrollo de la aplicación. Mayor detalle se puede ver en el repositorio.

Todos los archivos se encuentran en el repositorio de Github: https://github.com/afvasque/IIC2173NewsRoom