

**Parcial 2**

**DOCUMENTO DE ARQUITECTURA DE SOFTWARE**

**Grupo de Sistemas Distribuidos**

**DIEGO ALBERTO RINCÓN YÁÑEZ MCSc**

Índice

[Control de Cambios 4](#_Toc512437641)

[Propósito y Alcance 5](#_Toc512437642)

[Definiciones, Acrónimos y Abreviaciones 5](#_Toc512437643)

[Organización del Documento 5](#_Toc512437644)

[Representación de Stakeholders 5](#_Toc512437645)

[Definiciones de Puntos de Vista 6](#_Toc512437646)

[Punto de Vista Gerencial 6](#_Toc512437647)

[Descripción 6](#_Toc512437648)

[Stakeholders y sus Necesidades 6](#_Toc512437649)

[Lenguajes para Representar las Vistas 6](#_Toc512437650)

[Punto de Vista Lógica de Negocio 6](#_Toc512437651)

[Descripción 6](#_Toc512437652)

[Stakeholders y sus Necesidades 6](#_Toc512437653)

[Lenguajes para Representar las Vistas 6](#_Toc512437654)

[Punto de Vista Técnico 6](#_Toc512437655)

[Descripción 6](#_Toc512437656)

[Stakeholders y sus Necesidades 6](#_Toc512437657)

[Lenguajes para Representar las Vistas 6](#_Toc512437658)

[**Antecedentes de la Arquitectura** 7](#_Toc512437659)

[**Antecedentes del Problema** 10](#_Toc512437660)

[**Visión General del Sistema** 11](#_Toc512437661)

[**Objetivos y Contexto** 12](#_Toc512437662)

[**Vistas** 13](#_Toc512437663)

[**Vista de Componentes** 13](#_Toc512437664)

[**Capa de Negocio** 13](#_Toc512437665)

[**Capa API** 13](#_Toc512437666)

[**Vista de Despliegue** 14](#_Toc512437667)

[Referencias 15](#_Toc512437668)

[Glosario 16](#_Toc512437669)

[Bibliografía 17](#_Toc512437670)

[Cibergrafía 17](#_Toc512437671)

# 

# 

# Control de Cambios

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Autor | Versión | Comentarios |
| 22-Abril-2018 | David Cortes | 1.0 | Documento inicial |
| 23-Abril-2018 | Mateo Garzon | 1.1 | Antecedentes de arquitectura  Definiciones, Acrónimos y Abreviaciones  Organización del documento |
| 24-Abril-2018 | David Cortes | 1.2 | Propósito y alcance |
| 25-Abril-2018 | Cryssthian Roa | 1.3 | Referencias y bibliografía |
| 25-Abril-2018 | Felipe Muñoz | 1.4 | Diagramas de componentes y distribución |
| 25-Abril-2018 | Andrés Quintero | 1.5 | Puntos de vista |

**Introducción**

## Propósito y Alcance

Este documento es actualizable de acuerdo a los cambios técnicos de arquitectura que vayan apareciendo. Se detalla el equipo que mantiene la infraestructura y los requerimientos de cambios que el equipo pueda tener.

El propósito del documento es mantener organizada la Arquitectura Técnica y proveer una fuente de referencia para los analistas y diseñadores de la aplicación.

Este documento describe a detalle el diseño de la arquitectura propuesta para dar solución a la problemática planteada, se representarán diagramas de: clases, casos de uso y diagramas de flujo entre otros.

## Definiciones, Acrónimos y Abreviaciones

**MVC:** Modelo, Vista Controlador.

**API:** Application Programing Interface

## Organización del Documento

El documento está basado en el formato que se encuentran en la plantilla “SAD-1709.doc”, propuesta para llevar a cabo el documento de arquitectura del proyecto propuesto.

Representación de Stakeholders

|  |  |  |  |
| --- | --- | --- | --- |
| Interés | Mucho | ONG, comunidades de software libre | Free Software Fundation |
| Poco | Usuarios | Gobierno, Entidades Publicas |
|  |  | Poca | Mucha |
|  |  | Influencia | |

## 

## **Definiciones de Puntos de Vista**

### Punto de Vista Gerencial

Cómo gerente, empresario o profesional de los negocios la tecnología de información, especialmente en aplicaciones empresariales basadas en telecomunicaciones, ayuda a una empresa a superar las barreras geográficas, de tiempo, de costo y estructurales para lograr el éxito empresarial.

Por ejemplo, los vendedores viajeros y los de oficinas de ventas regionales pueden utilizar Internet, extranets y otras redes para transmitir pedidos de clientes desde laptops o computadores personales de escritorio, venciendo así las barreras geográficas.

### Punto de Vista Lógica de Negocio

**Capa de cliente.** La capa de cliente está formada por la lógica de la aplicación a la que el usuario final accede directamente mediante una interfaz de usuario. La lógica de la capa de cliente podría incluir clientes basados en navegadores, componentes de Java que se ejecuten en un equipo de escritorio o clientes móviles de JavaTM Platform, Micro Edition (plataforma J2METM) que se ejecuten en un dispositivo portátil.

**Capa de presentación.** La capa de presentación está formada por la lógica de aplicación, que prepara datos para su envío a la capa de cliente y procesa solicitudes desde la capa de cliente para su envío a la lógica de negocios del servidor. La lógica en la capa de presentación está formada normalmente por componentes de J2EE como, por ejemplo, Java Servlet o los componentes de JSP que preparan los datos para enviarlos en formato HTML o XML, o que reciben solicitudes para procesarlas. Esta capa también puede incluir un servicio de portal que proporcione acceso personalizado y seguro a los servicios de negocios en la capa de servicios de negocio.

**Capa de servicios de negocios.** La capa de servicios de negocio consiste en la lógica que realiza las funciones principales de la aplicación: procesamiento de datos, implementación de funciones de negocios, coordinación de varios usuarios y administración de recursos externos como, por ejemplo, bases de datos o sistemas heredados. Esta capa suele estar formada por componentes firmemente acoplados que se ajustan al modelo de componentes distribuidos de J2EE como, por ejemplo, los objetos Java, los componentes EJB o los beans conducidos mediante mensajes. Pueden montarse componentes de J2EE individuales para ofrecer servicios de negocios complejos, como, por ejemplo, un servicio de inventario o uno de cálculo de impuestos. Los componentes individuales y los ensamblados de servicios se pueden encapsular como servicios web que no estén firmemente acoplados en un modelo de arquitectura orientada a servicios, que se ajuste a los estándares de la interfaz SOAP (Simple Object Access Protocol). Los servicios de negocios también se pueden crear como servidores independientes como, por ejemplo, un servidor de mensajería o un servidor de calendario empresarial.

**Capa de datos.** La capa de datos está formada por los servicios que proporcionan los datos persistentes utilizados por la lógica de negocios. Los datos pueden ser datos de aplicaciones almacenados en un sistema de administración de bases de datos o pueden incluir información de recursos y directorios almacenada en un almacén de datos de protocolo ligero de acceso a directorios (LDAP). Los servicios de datos también pueden incluir alimentación de datos de orígenes externos o datos a los que se puede obtener acceso desde sistemas informáticos heredados.

### Punto de Vista Técnico

Se requiere un software que brinde Funcionalidad, Escalabilidad, Confiabilidad, Compatibilidad y Facilidad de operación. Entre las características mínimas del desarrollo de este software se requiere que sea multiplataporfa y que posea una interfaz web, la funcionalidad del software debe permitir:

• La aplicación debe estar disponible para cualquier dispositivo con conexión a internet

• la funcionalidad principal del aplicativo es difundir la información disponible en Wikipedia de manera abierta y accesible para todos, asi como su edición y creación de nuevas entradas del aplicativo

# **Antecedentes de la Arquitectura**

**MVC:**

Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

Este modelo de arquitectura se puede emplear en sistemas de representación gráfica de datos, como se ha citado, o en sistemas CAD, en donde se presentan partes del diseño con diferente escala de aumento, en ventanas separadas.

Este modelo de arquitectura presenta varias ventajas:

· Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado

· Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.

· La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los Componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas. Este escenario contrasta con la aproximación monolítica típica de muchos programas Java. Todos tienen un Frame que contiene todos los elementos, un controlador de eventos, un montón de cálculos y la presentación del resultado. Ante esta perspectiva, hacer un cambio aquí no es nada trivial.[3]

**Definición de las partes:**

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

La porción del programa que transforma los datos dentro del Modelo en una presentación gráfica es la Vista. La Vista incorpora la visión del Modelo a la escena; es la representación gráfica de la escena desde un punto de vista determinado, bajo condiciones de iluminación determinadas.

El Controlador sabe que puede hacer el Modelo e implementa el interface de usuario que permite iniciar la acción.

**Cloud Computing:**

El Cloud computing que hoy nos ofrecen es la suma de la evolución de varias tecnologías, las cuales ofrecen una mejor experiencia al usuario final.

• Aumento de la capacidad de procesamiento. Desde el origen de la informática, la capacidad de cómputo de los ordenadores personales se ha ido incrementando de forma vertiginosa.

• Conexión a Internet. La Red se ha convertido en una herramienta casi indispensable en la vida cotidiana de las personas. Su evolución implica aumento en la velocidad de conexión y en el número de conexiones en los hogares y en el trabajo.

• Dispositivos móviles. La miniaturización de los componentes informáticos ha permitido la aparición de dispositivos móviles que permiten la conexión permanentemente a Internet. Hoy en día, en un negocio es necesario poderse conectar con los recursos de la empresa, tanto desde ordenadores fijos como desde dispositivos portátiles, convirtiéndose la ubicuidad y movilidad en requisitos de gran importancia.

En conclusión, los avances en los tres campos mencionados anteriormente (capacidad de procesamiento, conexión a Internet y dispositivos móviles) junto a las importantes inversiones realizadas por las grandes empresas que dominan el panorama tecnológico mundial han propiciado la rápida evolución e implantación del cloud computing. Hasta tal punto que muchos usuarios ya disfrutan los servicios en la nube sin darse cuenta. [2]

## **Antecedentes del Problema**

El proyecto consiste en el desarrollo de una plataforma distribuida de consumo híbrido (web, desktop (multi plataforma) y móvil (Multi OS)) y masivo, capaz de gestionar la base de conocimiento actual y futura de Wikipedia. Debido a que la nueva plataforma será en memoria de Aaron Swartz, esta seguirá su filosofía, que dice que el conocimiento es de la humanidad, y que todas las personas del mundo podrán tener acceso a este. Esto en términos de funcionalidad quiere decir que, cualquier persona, el cual tenga acceso a un dispositivo que sea capaz de navegar en internet, por cualquier medio, podrá utilizar los servicios de acceso a conocimiento que se prestan en la organización. El objetivo es que el grupo de Sistemas Distribuidos en la Franja Nocturna Analice y Diseñe y presente la propuesta del sistema utilizando lo aprendido en clase, desde los conceptos iniciales hasta control de transacciones. Lo anterior complementado con su conocimiento de computación y herramientas digitales de nueva generación.

### **Visión General del Sistema**

La aplicación propuesta plantea lograr la mayor unificación de información y permitir el acceso libre con mayor rapidez, efectividad y confiabilidad a esta, para 2030 visualizamos ser la base de conocimiento más grande del mundo logrando que el acceso a la información sea para todos, sin restricciones, permitiendo con una estructura distribuida en Cloud que permite tener acceso eficaz a los datos que contendrán la información que se va a compartir y un modelo de arquitectura MVC, que permite su facilidad en la escalabilidad y mantenimiento, garantizar el funcionamiento continuo con una base de indisponibilidad baja. Con el control de lo que se hace con los documentos en la aplicación se tendrá una moderación y confiabilidad, para poder tener calidad de frente a la comunidad permitiéndoles así tener forma de expresar su opinión mediante foros de discusión en los artículos publicados.

### **Objetivos y Contexto**

· Los usuarios (N) podrán trabajar sincronizadamente en los documentos de forma colaborativa.

· Se conservará el estado (autoguardado) de los documentos a medida que los usuarios trabajan en el mismo.

· En cada archivo se deberá guardar la información de quien lo modificó y en qué fecha, así como el listado total de usuarios involucrados.

· Los usuarios podrán trabajar sincronizadamente y realizar operaciones indiscriminadamente en el sistema usando un cliente local (desktop o móvil), así como el cliente web.

· Para el manejo de la durabilidad de las transacciones se dispondrá de una operación adicional de generar un guardado permanente (commit) que podrá ser activada cuando el usuario lo desee o cada cierto tiempo, definido por el administrador del sistema

· Cada cambio realizado en el sistema deberá ser propiamente guardado, junto con un TIMESTAMP, el usuario, el cambio (similar a un REDOLOG) y la terminal de donde fue accedido el documento. Facultad de Ingeniería Ingeniería de Sistemas y Computación 2

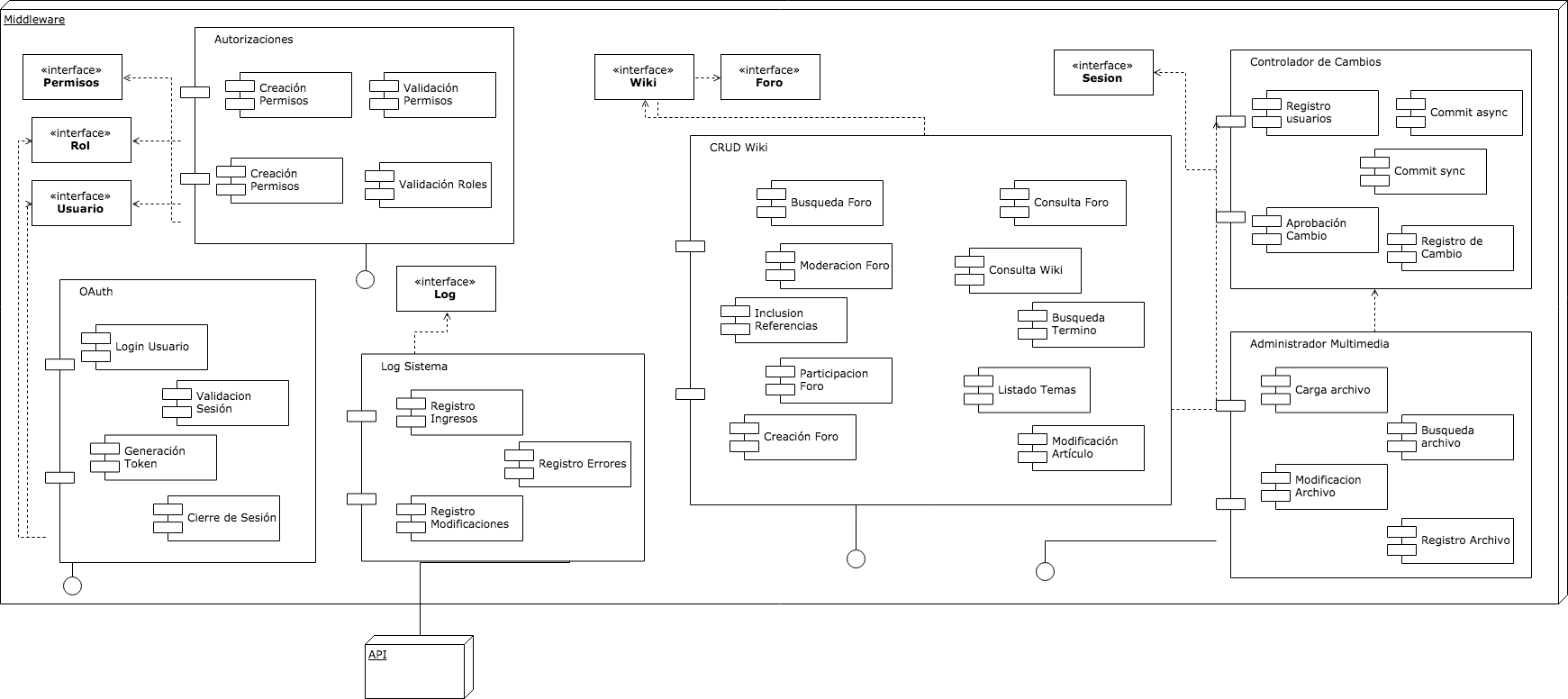
· Adicionalmente, por cada documento se deberá crear un “foro”, de discusión para la edición de artículos de importancia.

· Deberá existir un esquema de permisos de usuario capaz de recrear, rankings de experticia, para moderar y gestionar el contenido de artículos catalogados de interés.

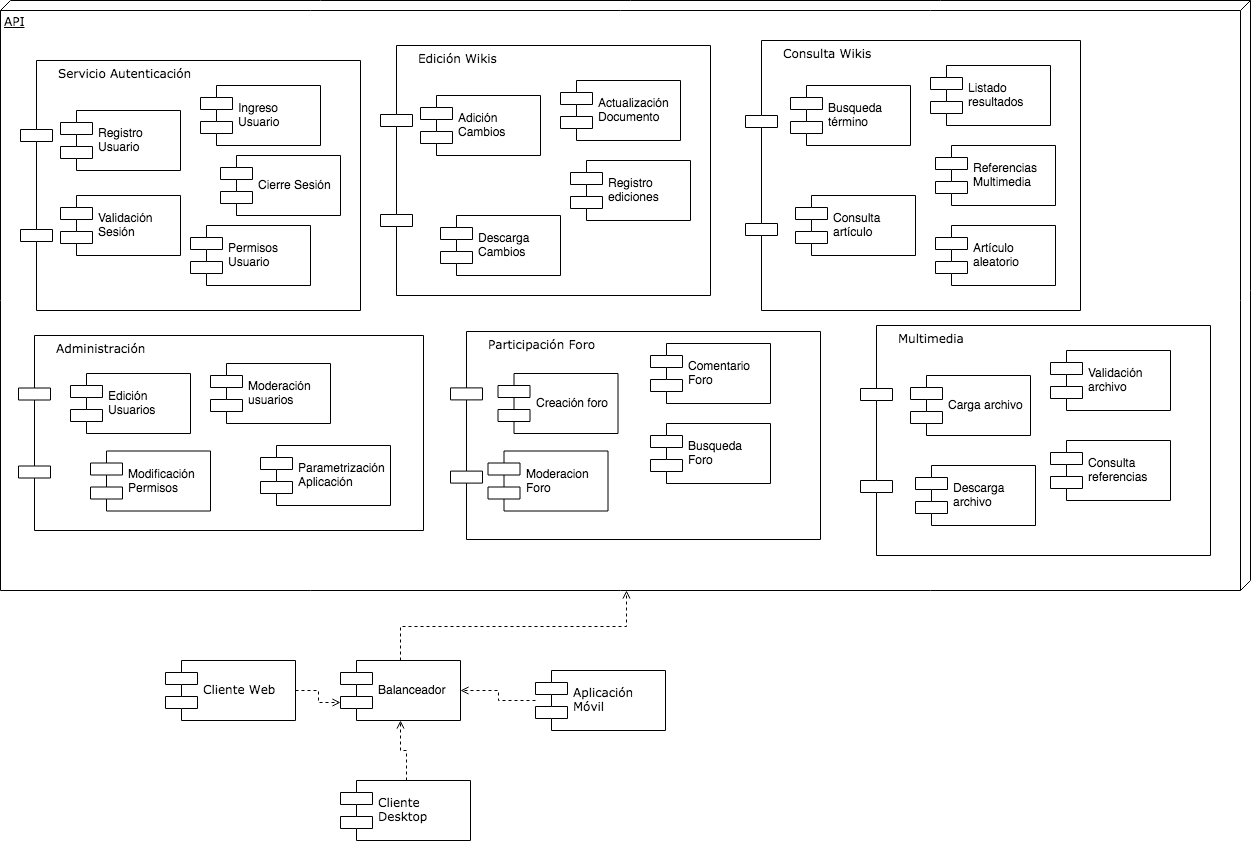
# **Vistas**

## **Vista de Componentes**

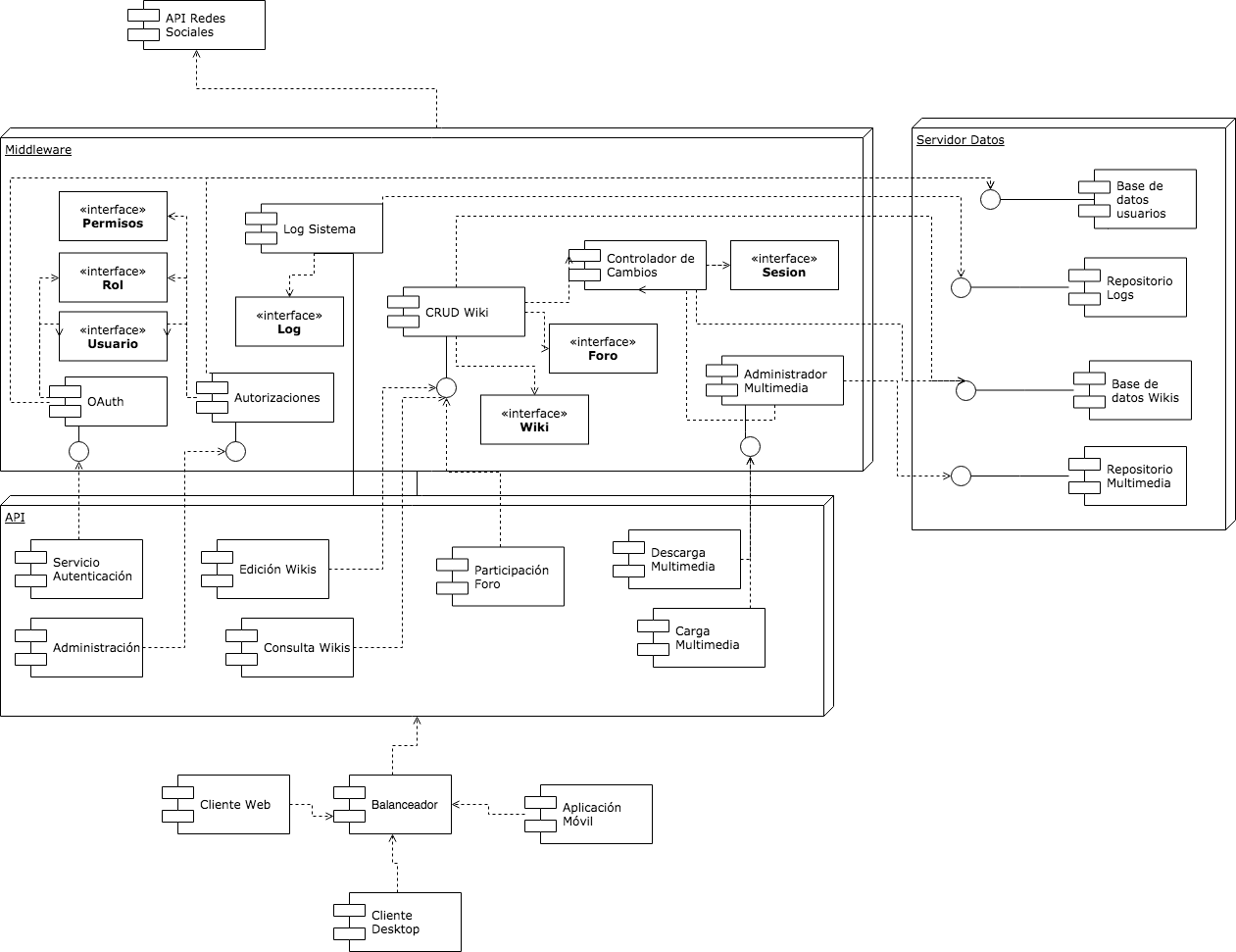
### **Capa de Negocio**



### **Capa API**



### **Vista de Despliegue**



# 

# Referencias

1. Fernández, L. (2008). *Arquitectura de software.* Recuperado el 22 de abril de 2018 de <https://sg.com.mx/content/view/409>
2. Escamilla, J. (2012). *Cloud computing - Antecedentes.* Recuperado el 22 de abril de 2018 de <http://cloud-fi.blogspot.com.co/2012/11/antecedentes-cloudcomputing-no-es-un.html>
3. Catalani, E. (2007). *Arquitectura Modelo/Vista/Controlador.* Recuperado el 22 de abril de 2018 de <https://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/>
4. Leuf Bo, Cunningham Ward (2001). Lew the Wiki Way: Quick Collaboration on the Web. Recuperado el 24 de abril de 2018 de http://www.citeulike.org/group/1136/article/114322

# Glosario

API: Capa de abstracción, conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software. Las siglas API vienen del inglés Application Programming Interface. En español sería Interfaz de Programación de Aplicaciones.

Oauth: Open Authorization (OAuth) es un [estándar abierto](https://es.wikipedia.org/wiki/Est%C3%A1ndar_abierto) que permite flujos simples de autorización para sitios web o aplicaciones informáticas.

# Bibliografía

1. Fernández, L. (2008). *Arquitectura de software.* Lugar de publicación: SG Software Guru.Recuperado el 22 de abril de 2018 de <https://sg.com.mx/content/view/409>
2. Escamilla, J. (2012). *Cloud computing - Antecedentes.* Lugar de publicación: Bolgspot. Recuperado el 22 de abril de 2018 de <http://cloud-fi.blogspot.com.co/2012/11/antecedentes-cloudcomputing-no-es-un.html>
3. Catalani, E. (2007). *Arquitectura Modelo/Vista/Controlador.* Lugar de publicación: WordPress.Recuperado el 22 de abril de 2018 de <https://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/>

# Cibergrafía

1. <https://oauth.net/2/>
2. <https://es.wikipedia.org/wiki/Wiki_estructurada>
3. http://www.citeulike.org/group/1136/article/114322