**Documento de Arquitectura**

PROYECTO: “Sistema adaptativo de formación educativa (SAFE)”

Integrantes – Año 2016

|  |  |  |
| --- | --- | --- |
| Legajo | Nombre | E-Mail |
| 1173388 | Santiago, Peralta | [speralta83@gmail.com](mailto:speralta83@gmail.com) |
| 1202388 | Obregon, Juan Facundo | [facundo.obregon@safabox.com](mailto:facundo.obregon@safabox.com) |
| 1223161 | Montañez, Cinthia | [cinthiamontaez@gmail.com](mailto:cintiamontaez@gmail.com) |
| [1](mailto:cintiamontaez@gmail.com)237937 | Caro, Jonatan | [joni1087@gmail.com](mailto:joni1087@gmail.com) |
| 1134115 | Matsui, Gerardo | [gerardo.matsui@gmail.com](mailto:gerardo.matsui@gmail.com) |

Profesores:

***Director de Cátedra:*** *Dra. Inés Casanovas*

***Profesor a cargo del curso:*** *Ing.*

***Profesor a cargo del proyecto:*** *Lic. Silvia Balduzzi - Ing. Pablo Abramowicz*

***Controller:*** *Mag.Ing. Gabriela Salem*

Contenido

[Historial de Revisión 3](#_Toc458094719)

[1. Objetivo 4](#_Toc458094720)

[2. Diagrama de Contexto 4](#_Toc458094721)

[3. Diagrama de Arquitectura 5](#_Toc458094722)

[**3.1.** **Arquitectura General** 5](#_Toc458094723)

[**3.2.** **Arquitectura de Infraestructura** 7](#_Toc458094724)

[Tecnologías 9](#_Toc458094725)

[4. Atributos de calidad – Requisitos No Funcionales 10](#_Toc458094726)

[5. Tácticas para garantizar RNF 11](#_Toc458094727)

[6. Aprobaciones 15](#_Toc458094728)

Historial de Revisión

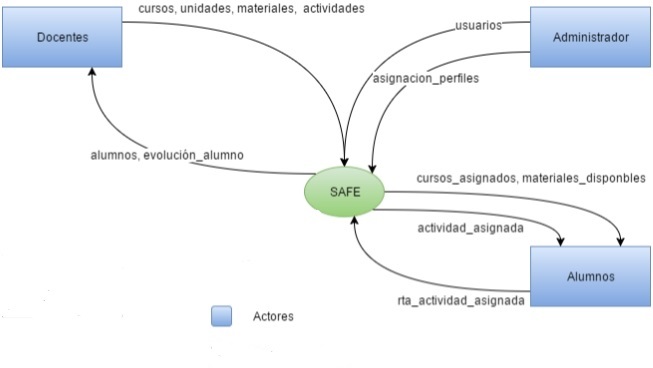
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fecha | Versión | Descripción | Rol | Autor |
| 03/06/2016 | 1.0 | Creación del documento. | Arquitecto de Software | Gerardo Matsui |
| 10/06/2016 | 1.1 | Corrección del diagrama de contexto, diagrama de arquitectura y patrones de diseño.  Definición de la arquitectura de infraestructura.  Agregado de RNF, tácticas de RNF, métodos de integración y patrones. | Arquitecto de Software | Gerardo Matsui |
| 24/06/2016 | 1.2 | Correcciones al documento. | Arquitecto de Software | Gerardo Matsui |
| 26/06/2016 | 1.3 | Correcciones en diagramas y descripciones | Arquitecto de Software | Gerardo Matsui |
| 04/08/2016 | 1.3 | Corrección de diagramas y agregado de descripciones en infraestructura y arquitectura lógica. | Arquitecto de Software | Juan Facundo Obregon |
| 02/09/2016 | 1.3 | Correcciones para los mecanismos de integración, sin servicios web. Corrección de requerimientos no funcionales | Arquitecto de Software | Gerardo Matsui |

# Objetivo

Este documento describe la arquitectura general de la solución implementada en el proyecto, los componentes individuales que componen la solución y la forma en que interactúan.

# Diagrama de Contexto

El siguiente diagrama muestra en un alto nivel de detalle, los entes y entidades que interactúan con el sistema, determinando los límites y el ambiente del mismo.



# Diagrama de Arquitectura

## **Arquitectura General**

El sistema esta comprendido por un grupo de módulo, cada uno con su respectivas funcionalidades y responsabilidades.



**Módulo de Perfiles:** Gestionará a los usuarios del sistema.

* Alta, baja y modificación de los usuarios del sistema.
* Asignación del perfil del usuario: define el tipo de perfil que tendrá un usuario pueden ser del tipo Alumno, Docente o Administrador.
* Autorización: Permite/deniega determinadas operaciones de acuerdo al perfil del usuario.

**Módulo de Alumnos:** Gestionará la información referente a los cursos y actividades que realizan los alumnos

* Cursos en los que se encuentra asignado el alumno.
* Las actividades a realizar por el alumno.
* El avance en las actividades realizado por el alumno.

**Módulo de Docentes:** Administrará la información de los cursos que imparte y el avance de los alumnos en cada curso.

* Cursos impartidos por el docente.
* Alumnos asignados en cada curso.
* El avance de cada alumno del curso.

**Módulo de Cursos:** Gestionará la información referente a los cursos.

* Alta, baja y modificación de los cursos.
* Definición del temario del curso.
* Dependencias entre cada tema.

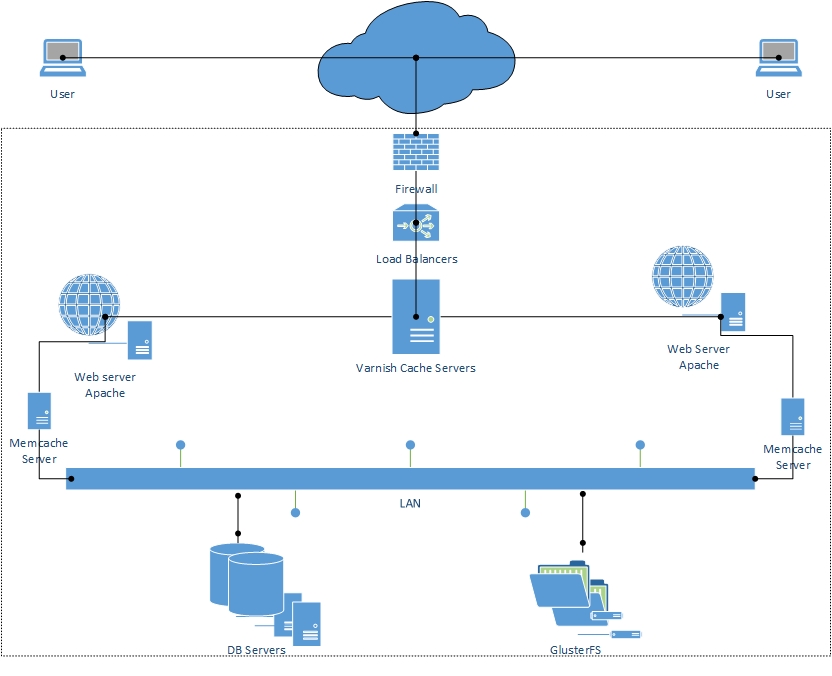
**Módulo de Actividades:** Gestionará la información referente a las actividades, la misma pueden ser de carácter teórico, práctico o ambas.

* Alta, baja y modificación de actividades.

**Módulo de Autoaprendizaje:** Será el encargado de registrar los progresos de cada alumno y determinar en base a la información recolectada la siguiente actividad a realizar por el alumno.

## **Arquitectura de Infraestructura**

El siguiente diagrama ofrece una vista de la infraestructura que utilizará la solución:



La solución se basará en un sistema web, los usuarios podrán acceder al mismo mediante sus exploradores, en la sección 4 se listan los exploradores compatibles.

Se implementará el uso de un Firewall de Aplicación Web (WAF) como punto de acceso a la plataforma para brindar seguridad a la red, filtrando y controlando toda entrada y salida de paquetes, permitiendo analizar el tráfico web, detectar y prevenir cualquier ataque malicioso al sistema y limitar el acceso de sectores donde los usuarios no deben de acceder resguardando la información.

El balanceador de carga físico permitirá distribuir la carga entre diferentes nodos para garantizar la mejor capacidad de procesamiento y confiabilidad como así también asegurar la disponibilidad de las aplicaciones monitorizando el estado de las misma.

Para mejorar las prestaciones y los tiempos de respuesta de la aplicación se utilizarán dos niveles de caché.

El primer nivel de cache será implementando servidores de cache Varnish que guardaran la información estática consultada con más frecuencia como vistas o contenido pesado y otro a nivel acceso a datos utilizando servidores MemCache reduciendo el número de consultas a la base de datos.

La solución se desarrollará en PHP, por consiguiente, se utilizará servidores de aplicación Apache. A los servidores Apache se les configurarán los módulos necesarios para poder correr el framework Symfony en su versión 2.8 LTS.

Como sistemas de archivos para contenidos se utilizará GlusterFS montado sobre la granja de servidores. Esta solución implementará un sistema de archivos virtual que puede configurarse en varios servidores físicos de menor potencia, permitiendo tener una mayor flexibilidad y dar la posibilidad de escalar horizontalmente, distribuyendo el almacenamiento en los diferentes servidores físicos brindando posibilidades de distribución y replica de datos.

Asimismo, se implementará un clúster de base de datos MySQL buscando brindar una mayor flexibilidad y incluyendo la posibilidad de escalar horizontalmente, distribuyendo los en los diferentes servidores físicos brindando posibilidades de distribución y replica de datos.

La siguiente tabla muestra un resumen de las diferentes tecnologías a utilizar por la plataforma:

### Tecnologías

|  |  |
| --- | --- |
| **Tecnología** | **Descripción** |
| ***Firewall*** | *Firewall de aplicaciones (WAF).* |
| ***Load Balancer*** | *Balanceador de carga, Cisco ASA 5508 o Barracuda Load Balancer 240.* |
| ***Varnish Cache Servers*** | *Servidores de Cache Varnish para contenido estático pesado.* |
| ***Web Server*** | *Servidores web apache.* |
| ***MemCache Servers*** | *Servidores de cache MemCache, para reducir consultas y accesos a la base de datos.* |
| ***Data Base Cluster*** | *Cluster de servidores de bases de datos PostgreSQL.* |
| ***File Servers*** | *GlusterFS.* |
| ***Symfony 2.8 LTS*** | *Framework de aplicaciones PHP.* |

# Atributos de calidad – Requisitos No Funcionales

|  |  |
| --- | --- |
| ***Portabilidad*** | |
| ***Estímulo*** | *Origen del Estímulo* | | *Implementador* |
| *Estímulo* | | *Implementación del sistema* |
| ***Ambiente*** | *Ambiente* | | *Al momento de instalar el sistema* |
| *Componente* | | *Sistema* |
| ***Respuesta*** | *Respuesta* | | *El aplicativo debe permitir su instalación en plataforma Windows y GNU/Linux* |
| *Medida de Respuesta* | | *Cada implementación* |

|  |  |
| --- | --- |
| ***Escalabilidad Horizontal*** | |
| ***Estímulo*** | *Origen del Estímulo* | | *Alto volumen de carga* |
| *Estímulo* | | *Mayor capacidad de procesamiento* |
| ***Ambiente*** | *Ambiente* | | *Crecimiento en el volumen de usuarios en el sistema* |
| *Componente* | | *Sistema* |
| ***Respuesta*** | *Respuesta* | | *El aplicativo debe permitir funcionar con otras instancias en paralelo* |
| *Medida de Respuesta* | | *420 request por segundo* |

# Tácticas para garantizar RNF

A continuación, se detallan las tácticas para lograr los atributos de calidad.

* **Táctica para garantizar la compatibilidad**: Identificar el tipo y versión del navegador en el Request al sistema. Se agregará un algoritmo que obtendrá el tipo y versión del navegador que está siendo utilizado por el usuario. En caso de que el navegador no sea soportado por el sistema se le mostrará un cartel con el listado de navegadores soportados.
* **Táctica para portabilidad**: Utilizar recursos disponibles en ambas plataformas.

Para el desarrollo del sistema solo se utilizarán recursos (lenguaje, intérpretes, librerías, etc) disponibles en las plataformas GNU/Linux y Windows.

* **Escalabilidad horizontal**: Soporte para integrar un balanceador de carga.

Se desarrollará el sistema con capacidad de anteponerle un balanceador de carga y ejecutar varias instancias del sistema.

Mecanismos de integración

Las integraciones entre los componentes del sistema se realizarán mediante la técnica API REST sobre el protocolo HTTP y accesos a la base de datos y servidores de archivos mediante TCP/IP.

Patrones de diseño y de Arquitectura

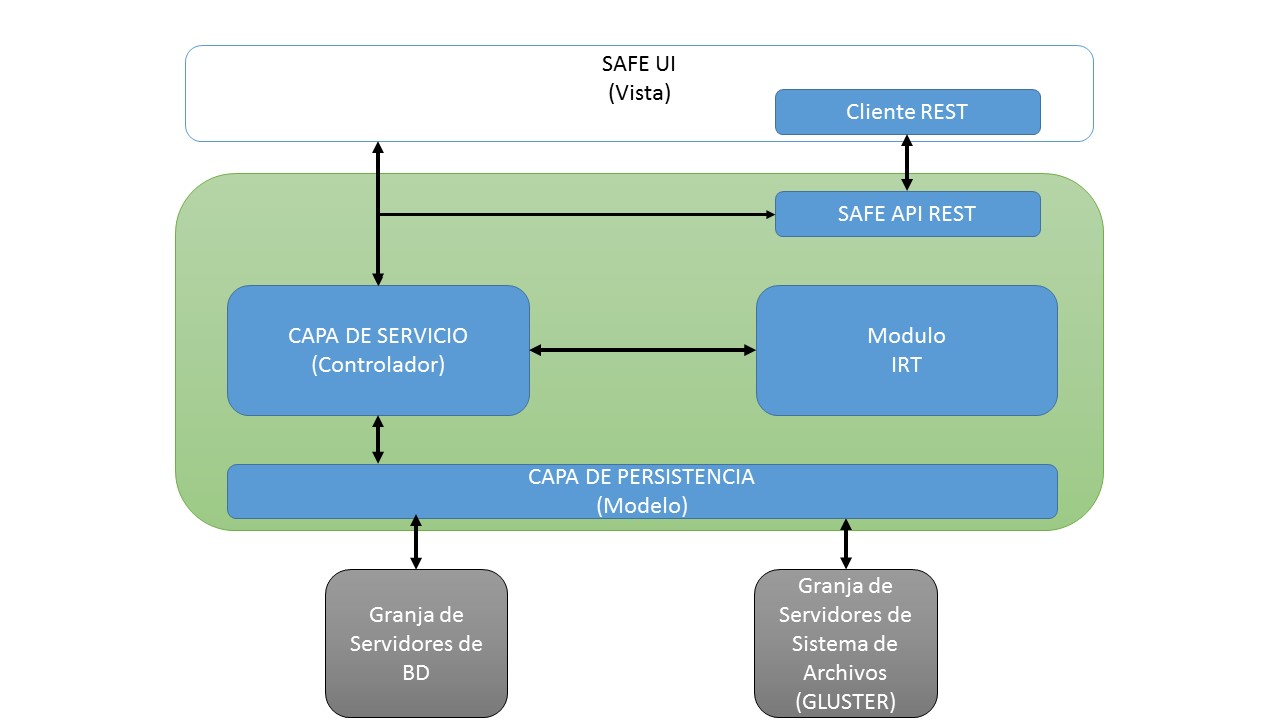
A continuación, se listan los diferentes patrones de diseños que serán utilizados para el desarrollo de la solución y como los componentes interactúan entre sí.

El framework Symfony en su versión 2.8 LTS (siglas en inglés para Soporte a Largo Plazo) será el utilizado como base para construir la plataforma.

Siguiendo la filosofía de Symfony, la solución se estructurará en diferentes Bundles que funcionaran como módulos donde se alocara la lógica del sistema, permitiendo un desarrollo modular y un mejor control de los mismos.

Cada Bundle estará compuesto por diferentes elementos que implementaran las diferentes funcionalidades del Patron MVC (Modelo – Vista - Controlador), con excepción del módulo IRT (Modulo de Autoaprendizaje) que será el encargado de realizar el auto aprendizaje de los usuarios y la evaluación de los mismos.

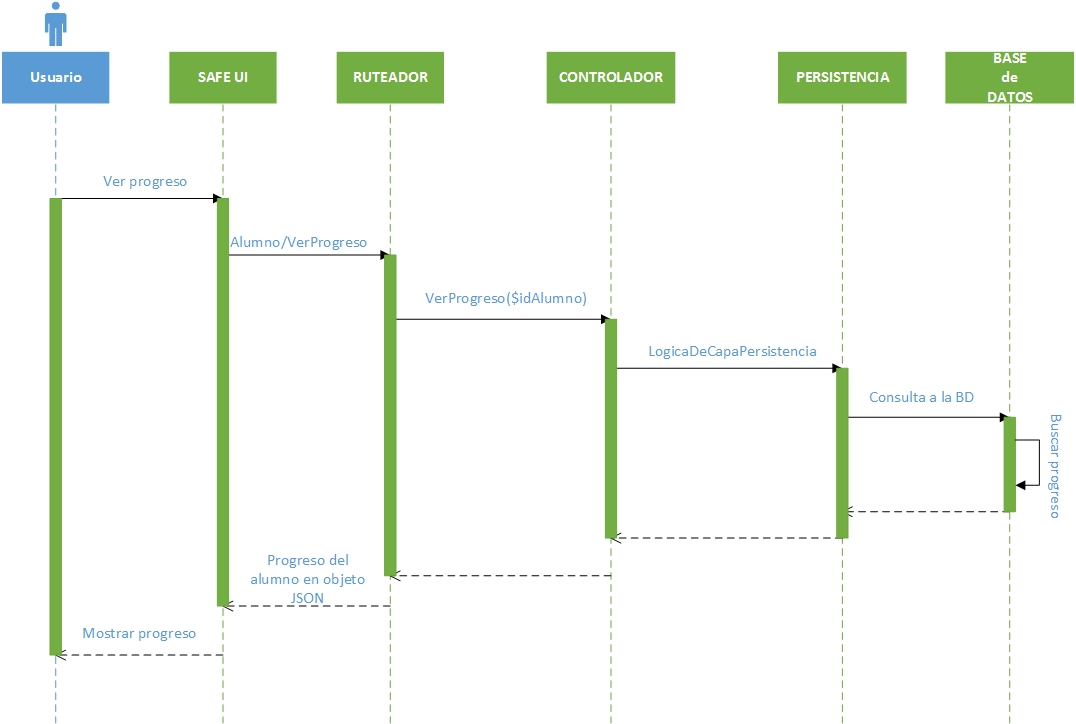
El siguiente grafico muestra una versión simplificada de la arquitectura lógica de la aplicación. Los controladores, vistas y modelos se agrupan en las capas de SAFE UI, SERVICIO y PERSISTENCIA para facilitar la comprensión del diagrama.



Asimismo, para obtener dinamismo y mejorar la experiencia del usuario, la capa SAFE UI (vistas) implementaran interfaces CLIENTE / API REST utilizando HTML5, AngularJS (en su versión 1.5.8 / 1.2.30) y Bootstrap 3.0, permitiendo desarrollar interfaces web agiles y responsiva siguiendo los estándares actuales.

Las interfaces API Rest se comunicarán con los diferentes controladores para realizar las diferentes operaciones CRUD (siglas en inglés para Crear, Consultar, Actualizar y Eliminar) y nunca accederán a la capa de modelo de forma directa.

El siguiente diagrama muestra una interacción de un pedido de usuario y como los diferentes componentes interactúan:



Cada capa tiene una responsabilidad dentro de la aplicación, brindando servicios a las capas superiores y consumiendo servicios de las capas inferiores.

# Aprobaciones

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Fecha:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Aprobación Técnica de la Arquitectura

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Fecha:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Profesor a cargo del Proyecto