Plataforma Web para la gestión de los productos e indicadores de responsabilidad en la Talabartería Montenegro

Documento Arquitectura de Software

Historial del Documento

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 14 – Oct - 2016 | 1.0 | Versión inicial del documento. | Valentina Correa Romero.  Andrés Felipe Gómez B.  Cristian Camilo Ramírez M. |
| 15 – Oct - 2016 | 1.0 | Definición requerimientos que afectan la arquitectura. | Valentina Correa Romero.  Andrés Felipe Gómez B.  Cristian Camilo Ramírez M. |
| 16 – Oct - 2016 | 1.0 | Identificación de las vistas del proyecto. | Valentina Correa Romero.  Andrés Felipe Gómez B.  Cristian Camilo Ramírez M. |
| 17 – Oct - 2016 | 1.0 | Identificación de casos de uso relevantes en la arquitectura. | Valentina Correa Romero.  Andrés Felipe Gómez B.  Cristian Camilo Ramírez M. |

Tabla de contenido

1. Introducción 5

1.1 Propósito 5

1.2 Alcance 5

1.3 Definiciones, acrónimos y abreviaciones 5

2. Requisitos arquitectónicos 6

2.1 Requerimientos no funcionales 6

2.2 Requerimientos funcionales 6

3. Vista lógica 6

3.1 Niveles 6

3.2 Subsistemas 8

3.2.1 Dominio 8

3.2.2 Datos 8

3.3 Comunicación Niveles 9

4. Vista de implementación 10

4.1 Estructura de los paquetes 10

4.2 Niveles (MVC en RAILS) 12

4.3 Componentes 13

5. Vista de despliegue 13

6. Vista de casos de uso 15

6.1 Identificación de los Casos de Uso relevantes para la arquitectura 15

Para el diseño de la plataforma Web de la Talabartería Montenegro, se identifican como casos de uso relevantes desde el punto de vista de la arquitectura, los siguientes: 15

6.1.1 Administración de productos: 15

6.1.2 Autenticación de usuarios: 15

6.1.3 Administración de indicadores 15

6.2 Descripción de los casos de uso relevantes para la arquitectura 15

6.2.1 Administración de productos: 15

6.2.2 Autenticación de usuarios: 16

6.2.3 Administración de indicadores 16

6.3 Especificación de Casos de Uso 17

6.3.1 Administración de productos: 17

6.3.2 Autenticación de usuarios: 17

# Introducción

## Propósito

El documento de arquitectura de software evidencia la arquitectura presentada para el desarrollo del proyecto Talabartería Montenegro a través de diferentes vistas, donde cada una de estas muestra un aspecto particular del software en desarrollo, así se pretende mostrar una visión global y fácil de comprender del diseño en cuanto arquitectura del software que se desarrolla.

## Alcance

El presente documento pretende explicar la arquitectura bajo la cual se llevará a cabo el desarrollo del Plataforma Web para la gestión de los productos e indicadores de responsabilidad en la Talabartería Montenegro, se encuentran descripciones y modelos correspondientes a cada una de las 4 + 1 vistas arquitectónicas.

## Definiciones, acrónimos y abreviaciones

* **Requerimientos:** algo que se le pide o solicita a alguien, características que se desea que posea un sistema o un software.
* **ORM:** (Object-Relational mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.
* **MVC:** El modelo – vista – controlador (MVC) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.
* **RAILS:** es un entorno de desarrollo web de código abierto que está optimizado para la satisfacción de los programadores y para la productividad sostenible. Te permite escribir un buen código evitando que te repitas y favoreciendo la convención antes que la configuración. (http://www.rubyonrails.org.es/)

# Requisitos arquitectónicos

## Requerimientos no funcionales

| **Fuente** | **Nombre** | **Relevancia Arquitectónica** | **Addres­sed in:** |
| --- | --- | --- | --- |
| Aceptación | Disponibilidad | La plataforma debe tener una disponibilidad del 95 % de las veces en que un usuario intente acceder, teniendo gran relevancia en la arquitectura de la plataforma con el fin de mantener dicha disponibilidad. | Vista Despliegue |
| Aceptación | Portabilidad | La aplicación web debe poseer un diseño “Responsive” a fin de garantizar la adecuada visualización en múltiples computadores personales, dispositivos tableta y teléfonos inteligentes. | Vista Casos de uso |
| Visión | Seguridad | Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos. | Vista lógica |
| Aceptación | Usabilidad | El sistema debe contar con manuales de usuario estructurados adecuadamente. | Vista Despliegue |

## Requerimientos funcionales

| **Fuente** | **Nombre** | **Relevancia Arquitectónica** | **Addressed in:** |
| --- | --- | --- | --- |
| Cliente | Administrar productos | El usuario administrador de la plataforma (Dueño de la talabartería) administrará dinámicamente los productos que comercializan, es decir crear, editar y eliminar los productos de la plataforma web. | Vista casos de uso |
| Cliente | Autenticación de usuarios | El usuario podrá iniciar una sesión con su cuenta previamente registrada con el fin de obtener más privilegios en la administración de la plataforma web. | Vista de casos de uso |
| Cliente | Autenticación de usuarios | El usuario administrador de la plataforma (Dueño de la talabartería) administrará dinámicamente los indicadores de responsabilidad que se identifiquen en la talabartería, este podrá crear, editar y eliminar estos indicadores de la plataforma web. | Vista de casos de uso |

# Vista lógica

## Niveles

Para el desarrollo del proyecto, la siguiente ilustración muestra la arquitectura impuesta por el framework de desarrollo Rails, en esta se identifican los niveles de presentación, servicio, dominio y datos, para la presentación se cuanta con la sección View, esta conserva las vistas que vera el usuario y serán cargadas con los datos o información correspondiente. Para el nivel de servicio, se puede ver como una petición recorre esta arquitectura con el fin de dar solicitud a esta. Para la capa de dominio, se cuanta con una sección model, esta conserva todas las clases del dominio necesarias para el proyecto. Y para los datos, está la capa de base de datos, en este proyecto se utilizara una base de datos relacional para la persistencia de los datos.

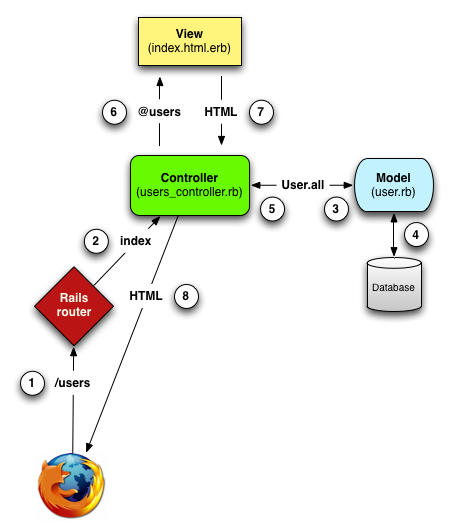


Ilustración 1: Arquitectura MVC – RAILS

A continuación se explica una como petición interactúa en la arquitectura propuesta para el proyecto:

1. El explorador emite una solicitud a la URL /users.
2. Rails enruta /users a la acción index del controlador usuarios.
3. La acción index pide el modelo de usuario para recuperar todos los usuarios (User.all).
4. El modelo de usuario obtiene todos los usuarios de la base de datos.
5. El modelo de usuario devuelve la lista de usuarios con el controlador.
6. El controlador captura los usuarios en la variable @ usuarios, que se pasa a la vista index.
7. La vista utiliza código Ruby embebido para representar la página como HTML.
8. El controlador pasa el código HTML al explorador.

## Subsistemas

En esta sección del documento, se hace referencia a los niveles anteriormente mencionados, haciendo una profundización considerable en estas capas.

### Dominio

A continuación se muestra el diagrama de dominio de la plataforma para la Talabartería Montenegro, en este puede ver las clases para la representación de los productos que allí se comercializan, los identificadores de responsabilidad identificados, además de los usuarios que modificaran el contenido de la plataforma web.

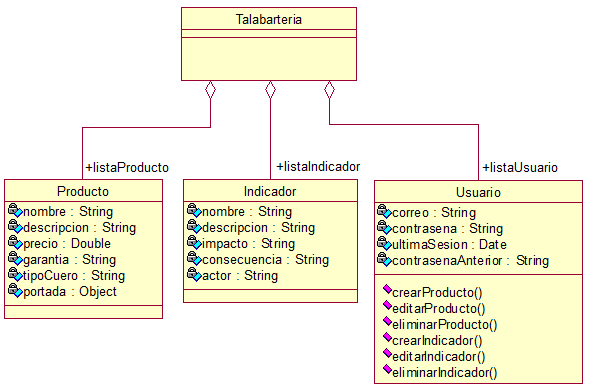


Ilustración 2: Dominio

### Datos

Para la persistencia de los datos de la plataforma web para la Talabartería Montenegro, se implementará una base de datos relacional, basados en el framework de desarrollo Rails de Ruby, cada clase del dominio, hace referencia a una entidad en la base de datos, por esto, el modelo de la base de datos corresponde al siguiente:

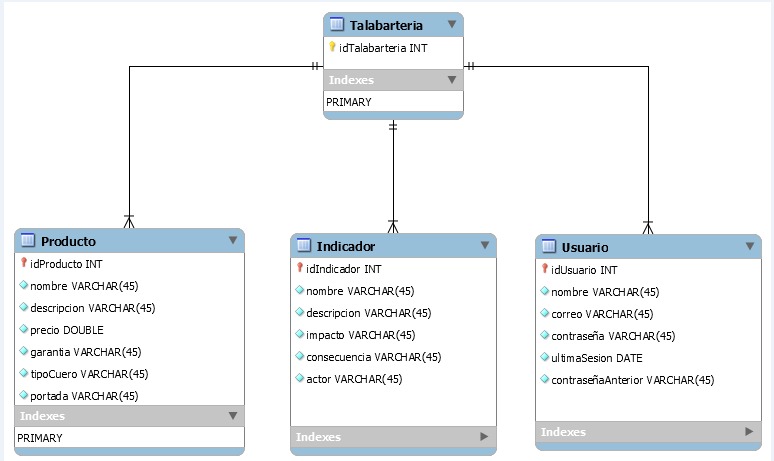


Ilustración 3: Datos

## Comunicación Niveles

En la siguiente ilustración se puede evidenciar como se comunican los niveles o capas de la vista lógica, en este se muestra una petición hecha desde el explorador, esta es atendida por un router rails que se encargar de ubicar el controlador necesario en la petición, a continuación el controlador solicita al modelo los usuarios, luego el modelo se comunica con la base de datos para obtener los datos necesarios, posteriormente el modelo envía estos datos al controlador, este último envía los datos a la vista que se encarga de cargarlos en la vista HTML por medio de código Ruby Embebido, luego la vista retorna al controlador, y por último envía el HTML al explorador.

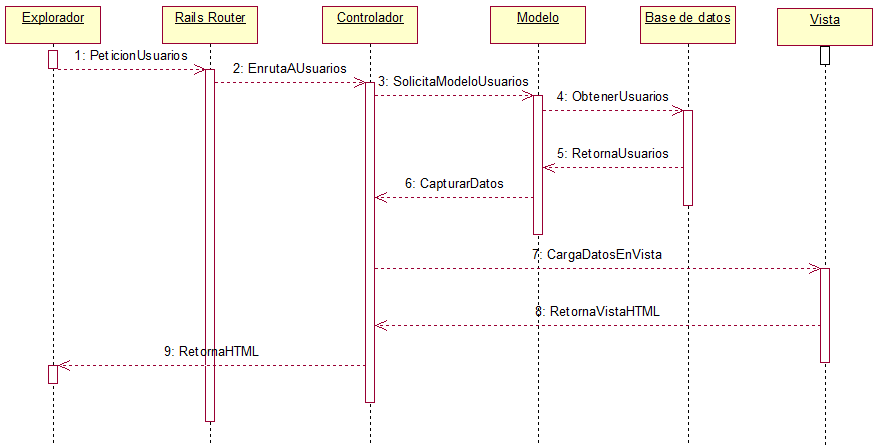


Ilustración 4: Secuencia Niveles

# Vista de implementación

En esta sección se presenta la estructura de paquetes, modelo y componentes para la implementación de la Plataforma Web para la gestión de los productos e indicadores de responsabilidad en la Talabartería Montenegro. Para esta implementación se utilizará el Framework de desarrollo Rails 4 del lenguaje de programación Ruby, lenguaje de programación orientado a objetos.

## Estructura de los paquetes

La estructura de los proyectos utilizando Rails tiene un directorio de aplicaciones llamado app / con tres subdirectorios: modelos, vistas y controladores. Esto se debe a que Rails sigue el modelo-vista-controlador (MVC) de arquitectura, que impone una separación entre "la lógica de dominio" (también llamado "la lógica de negocio") a partir de la entrada y la lógica de presentación asociada con una interfaz gráfica de usuario (GUI). En el caso de las aplicaciones web, la "lógica de dominio" por lo general se compone de los modelos de datos para cosas como los usuarios, artículos y productos, y la interfaz gráfica de usuario es una página web en un navegador web. Esta estructura se puede evidenciar mejor en la **Ilustración 1: Estructura del directorio de la aplicación** e **Ilustración 2: Estructura directorio app.**



Ilustración 5: Estructura del directorio de la aplicación



Ilustración 6: Estructura directorio app

## Niveles (MVC en RAILS)

Generalmente para interactuar con una aplicación de Rails, un navegador envía una solicitud, la cual es recibida por un servidor web y se transmite a un controlador de Rails, que es el encargado de definir qué hacer a continuación. El controlador interactúa con un modelo, que es un objeto de Ruby que representa un elemento de la página (por ejemplo, un usuario, un producto, etc.) y se encarga de comunicarse con la base de datos. Después de invocar el modelo, el controlador, genera la vista y devuelve la página web completa en el navegador como HTML.

**Modelo:**

* Acceso a los datos de la aplicación y reglas para manipularlos
* Modelos almacenados en BBDD relacionales ->ORM -> módulo ActiveRecord
* Una clase representa una tabla.
* Módulo genérico ActiveModel.
* Se descubren automáticamente los campos y la tabla (Producto ! productos)
* Se pueden declarar relaciones con otros modelos/tablas
* Se puede personalizar y añadir métodos

**Vista:**

* Para decidir el aspecto
* Módulo: ActionView
* Una por cada acción de cada controlador
* Corolario: dependen de los controladores
* HTML repetitivo, a funciones externas (helpers)

**Controlador:**

* «Bisagra» del MVC y lógica de cálculos.
* Cada controlador, una clase de Ruby.
* Módulo: ActionPack
* Cada método, una acción.
* «Andamios» (scaffold) para avanzar más rápido.

## Componentes

A continuación se enumeran los componentes y marcos de fabricación que se utilizan en la implementación de la plataforma para la Talabartería Montenegro.

* **Action Pack** Gema que contiene tres componentes
* **Action Controller** parámetros HTTP, sesiones, redirecciones.
* **Action View** genera HTML o XML a partir de plantillas
* **Action Dispatch** enruta las peticiones dentro de la aplicación o a otra aplicación Rack
* **Active Model** interfaz entre Action Pack y la capa de persistencia
* **Active Record** ORM, base de los modelos
* **Active Resource** permite utilizar recursos REST como objetos locales
* **Active Support** clases útiles y extensiones a las de Ruby
* **Railties** para crear aplicaciones y unir los demás componentes

# Vista de despliegue

En esta sección del presente documento, se puede apreciar un diagrama de despliegue, con el fin de ilustrar la arquitectura general en la que estaría la aplicación a nivel de despliegue, esta arquitectura giraría en un entorno cliente servidor, para esto, la plataforma Web para la gestión de los productos e indicadores de responsabilidad en la Talabartería Montenegro expone sus funcionalidades a través de un servidor por un puerto para su comunicación, las cuales son consumidas por n cantidad de clientes en diferentes espacios de tiempo a través de la conexión de los exploradores web.

|  |
| --- |
| <<Use>>  <<Use>>  <<Use>>  <<Use>>  <<Componente>>  PlataformaWebTalabarteriaMontenegro  Servidor de aplicaciones  Cliente 3  <<Artefacto>>  Explorador  Cliente 2  <<Artefacto>>  Explorador  Cliente 1  <<Artefacto>>  Explorador  <<Componente>>  GemSet  <<Componente>>  Rails  <<Componente>>  ActiveRecord  <<Componente>>  Sqlite |

Ilustración 7: Despliegue

| **Nombre** | **Tipo** | **Descripción** |
| --- | --- | --- |
| Sqlite | Nodo (Persistencia de los datos) | Nodo que representa la persistencia de los datos que se manejaran en la plataforma web. |
| ActiveRecord | Nodo | Modelo para gestionar los objetos en la base de datos |
| Rails | Framework | Framework de desarrollo del lenguaje Ruby para aplicaciones web. |
| GemSet | Librerías | Conjunto de implementaciones funcionales que permiten el desarrollo del proyecto. |
| Cliente | Nodo | Usuario de la plataforma que interactúa a través de un explorador. |

Tabla 1: Despliegue

# Vista de casos de uso

La vista de casos de uso presenta una descripción de los casos de uso o escenarios que presentan funcionalidades relevantes en el sistema, que generan gran impacto en la cobertura arquitectónica, este conjunto permite descubrir y diseñar la arquitectura del sistema.

## Identificación de los Casos de Uso relevantes para la arquitectura

## Para el diseño de la plataforma Web de la Talabartería Montenegro, se identifican como casos de uso relevantes desde el punto de vista de la arquitectura, los siguientes:

### Administración de productos:

* Es la funcionalidad principal impuesta por el cliente para la gestión de los productos comercializados en la talabartería.
* Este requerimiento impone la necesidad de contar un lenguaje dinámico que brinde dinamismo y flexibilidad al portal para la administración de los productos por parte del cliente.
* Este componente agrega la complejidad de manejar un framework que brinde seguridad y flexibilidad al momento del desarrollo.

### Autenticación de usuarios:

* Esta funcionalidad es sumamente importante para la gestión de la página por parte del cliente, por lo que se debe tener un nivel de alto seguridad en el modelo de la plataforma.
* Este requerimiento genera la necesidad de implementar modelos de seguridad para garantizar la autenticación de los usuarios.

### Administración de indicadores

* Funcionalidad que debe permitir al cliente modificar características del indicador.
* Esta funcionalidad requiere gemas o importaciones de gran relevancia que modifican el modelo de despliegue.
* Como parte esencial de la ejecución de este requerimiento, debe brindar dinamismo y mostrar buenas prácticas de la talabartería.

## Descripción de los casos de uso relevantes para la arquitectura

### Administración de productos:

Para la administración de los productos por parte del usuario con rol de administrador, se evidencia el siguiente caso de uso:

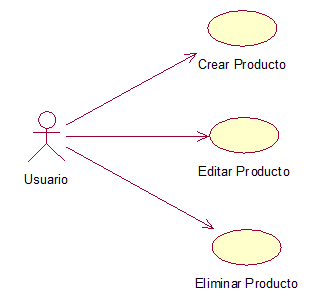


Ilustración 8: Caso de Uso Administrar Producto

### Autenticación de usuarios:

Para la autenticación de los usuarios es necesario registrarse e iniciar sesión como se evidencia en la siguiente ilustración:

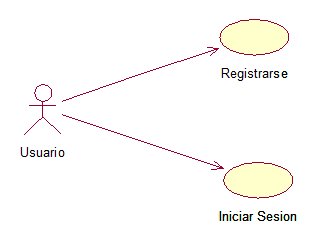


Ilustración 9: Caso de Uso autenticación usuario

### Administración de indicadores

Para la administración de los indicadores por parte del usuario con rol de administrador, se evidencia el siguiente caso de uso:

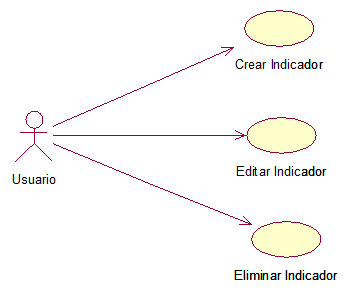


Ilustración 10: Caso de Uso Administrar Indicador

## Especificación de Casos de Uso

A continuación se muestra las especificaciones de los casos de uso anteriormente identificados e ilustrados:

### Administración de productos:

### 

|  |  |
| --- | --- |
| **Nombre** | Administración de productos |
| **Actores** | Usuario (Administrador) |
| **Descripción** | El caso de uso para administrar los productos de la talabartería consta: Crear producto: el usuario administrador podrá crear productos con las características de este, y agregando una imagen para su publicación.  Editar producto: el usuario administrador podrá editar las características de estos productos.  Eliminar producto: el usuario podrá eliminar los productos que ya no desee que se muestren en la plataforma. |

Tabla 2: Especificación administrar productos

### Autenticación de usuarios:

|  |  |
| --- | --- |
| **Nombre** | Autenticación de usuarios |
| **Actores** | Usuario |
| **Descripción** | En el caso de uso para la autenticación de usuarios consta de las siguientes funcionalidades:  Registrarse: el usuario puede llenar un formulario con algunos datos personales para crear una cuenta en la plataforma.  Iniciar sesión: el usuario puede iniciar sesión para crear productos e indicadores o administrar demás aspectos de la plataforma. |

Tabla 3: Especificación autenticación usuarios

### Administración de indicadores

|  |  |
| --- | --- |
| **Nombre** | Administrador de indicadores |
| **Actores** | Usuario (Administrador) |
| **Descripción** | El caso de uso para administrar los indicadores de la talabartería consta: Crear indicador: el usuario administrador podrá crear indicadores de responsabilidad con las características de este, y agregando una imagen para su publicación.  Editar indicador: el usuario administrador podrá editar las características de estos indicadores de responsabilidad.  Eliminar indicador: el usuario podrá eliminar los indicadores de responsabilidad que ya no desee que se muestren en la plataforma. |

Tabla 4: Especificación administrador de indicadores