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

Patrones

Historial de revisión

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 05-Oct-2016 | 1.0 | Identificación patrones de diseño | Valentina Correa Romero  Cristian Camilo Ramirez  Andrés Felipe Gómez B. |
| 06-Oct-2016 | 1.0 | Identificación patrones de arquitectura, implementación y usabilidad | Valentina Correa Romero  Cristian Camilo Ramirez  Andrés Felipe Gómez B. |

Tabla de contenidos

[1. Introducción 4](#_Toc463559466)

[1.1 Propósito 4](#_Toc463559467)

[1.2 Alcance 4](#_Toc463559468)

[1.3 Definiciones, Acronimos y abreviaturas 4](#_Toc463559469)

[2. Patrones de diseño 5](#_Toc463559470)

[2.1 Patrón de diseño Builder 5](#_Toc463559471)

[2.1.1 Problemas Identificados 5](#_Toc463559472)

[2.1.2 Diagrama de clases 5](#_Toc463559473)

[2.1.3 Consecuencias 6](#_Toc463559474)

[2.2 Patrón de diseño Composite (Compuesto) 6](#_Toc463559475)

[2.2.1 Problemas identifiados 6](#_Toc463559476)

[2.2.2 Diagrama de clases 7](#_Toc463559477)

[2.2.3 Consecuencias 8](#_Toc463559478)

[2.3 Patrón de diseño Cadena de responsabilidad 8](#_Toc463559479)

[2.3.1 Problemas identificados 8](#_Toc463559480)

[2.3.2 Diagrama de clase 8](#_Toc463559481)

[2.3.3 Consecuencias 9](#_Toc463559482)

[3. Patrones de Arquitectura, Usabilidad y análisis 9](#_Toc463559483)

[3.1 Patrón de aquitectura Active Record 9](#_Toc463559484)

[3.2 Patrón de usuabilidad 10](#_Toc463559485)

[3.3 Patrón de análisis 10](#_Toc463559486)

# Introducción

Patrón de diseño: Base para la búsqueda de soluciones a problemas comunes en el desarrollo de software.

Para que una solución sea considerada un patrón debe poseer ciertas características:

* Efectivo resolviendo problemas similares en ocasiones anteriores.
* Reutilizable: aplicable a diferentes patrones de diseño en distintas circunstancias.

Los patrones de diseño pretenden:

* Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
* Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
* Formalizar un vocabulario común entre diseñadores.
* Estandarizar el modo en que se realiza el diseño.
* Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento existente.

A continuación se mostrará una pequeña descripción de cómo se desea implementar estos patrones de diseño, arquitectura, usabilidad y analisis en el proyecto para la Talabartería Montenegro.

## Propósito

EL propósito de este documento es evidenciar la implementación de algunos patrones que ayudaran a la reutilización de código, aumentar la modularidad entre otros aspectos importantes para el desarrollo del proyecto de la Talabatería Montegro.

## Alcance

El alcance de este documento es la identificación y descripción de patrones de diseño, arquitectura, usabilidad y análisis para la implementación en el proyecto, teniendo en cuanta como ambiente de desarrollo ruby on rails.

## Definiciones, Acronimos y abreviaturas

* Patrón de diseño: Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores.

# Patrones de diseño

## Patrón de diseño Builder

El patrón builder es utilizado para la creación de objetos complejos que admiten diversas configuraciones, este separa la construcción de un objeto de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.

### Problemas Identificados

|  |  |
| --- | --- |
| **Problema de diseño** | **Solución** |
| El usuario requiere publicar sus productos en el portal, estos tienen diferentes características y métodos de construcción. | Aplicar patrón de diseño Build para el manejo de los objetos teniendo en cuanta la complejitdas de los productos y sus diferentes características. |
| Los productos identificados en la Talabartería Montenegro pueden tener diferentes métodos, caracteríticas y modelos de construcción, por lo que dificulta su representación. | Usar el patrón de diseño Build con el fin de implementar una construcción concreta a la hora de crear un producto específico. |

Table 1: Problema de diseño 1

### Diagrama de clases

Para la aplicación de este patrón de diseño dentro del proyecto Talabartería Montenegro, se tiene en cuenta en el manejo de objetos de tipo Producto, ya que este objeto posee partes concretas de diferente tipo para su representación. En la siguiente **Tabla 2: Patrón Build** se puede observar un ejemplo de dicha aplicación.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Director | |  | | ConstruirProducto(): producto;  Director(Builder):void; | | |  | | --- | | Builder | | Producto producto; | | ConstruirCuero();  ConstruirBastos();  ConstruirJuste();  ConstruirHerraje(); | | |  | | --- | | TerequeBuilder | | Producto producto; | | ConstruirCuero();  ConstruirBastos();  ConstruirJuste();  ConstruirHerraje(); | |
|  | |  | | --- | | CorreaBuilder | | Producto producto; | | ConstruirCuero();  ConstruirHerraje(); | | |  | | --- | | Producto | |  | |  | |

Tabla 2: Patrón Build

En el anterior diagrama se puede observar la clase “Director” que construye un objeto utilizando el Builder. El Builder es una interfaz abstracta para crear los productos. CorreaBuilder y TerequeBuilder son un Builder concreto, que se encarga de reunir y costruir las partes necesarios para crear un producto específico. Y la clase “Producto” representa el objeto complejo que será construido.

### Consecuencias

El uso de este patrón en el pryecto de la Talabartería Montenegro tendrá las siguientes consecuencias enumeradas acontinuación:

* Reduce la dependencia en los componentes del software.
* Cada ConcreteBuilder tiene el código específico para crear y modificar una estructura específica del objeto.
* Permite un mayor control en el proceso de creación de los objetos.
* Las clases concretas que tratan las representaciones no forman parte de la interfaz del builder.

## Patrón de diseño Composite (Compuesto)

El patrón de diseño Composite o compuesto permite construir objetos complejos partiendo de otroas más sencillos basados en una composición recursiva y a una estructura en forma de árbol.

### Problemas identifiados

|  |  |
| --- | --- |
| **Problema de diseño** | **Solución** |
| Algunos acciones que se deben realizar en diferentes productos lo que hace más complejo la implementación. | Aplicar el patrón Composite o Compuesto para una implementación más sencilla y reusar el código en los diferentes productos. |
| Teniendo en cuanta la complejidad de los objetos “Producto” en el proyecto, es necesario una implementación sencilla que sea facilmente mantenible a futuro. | Usar patrón Composite o Compuesto para un diseño simple y reutilizable para el desarrollador. |

Tabla 1: Problema de diseño 2

### Diagrama de clases

La aplicación de este patrón de diseño en el proyecto es de gran utilidad ya que permite manejar simplemente la representación de los productos que son componentes complejos debido a sus gran distribución, esta representación esta de distribuida para administrar eficazmente la responsabilidades de las clases, pero se ven y se tratan de manera única.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Producto | |  | | ConstruirProducto();  ModificarProducto();  EliminarProducto();  BuscarProducto(); | | | |  | | --- | | ProductoInterface | | String : Cuero; | | <<Override>> ConstruirProducto();  <<Override>> ModificarProducto();  <<Override>> EliminarProducto();  <<Override>> BuscarProducto(); | |
| |  | | --- | | Tereque | | String : Basto;  String : Juste; | | <<Override>> ConstruirProducto();  <<Override>> ModificarProducto();  <<Override>> EliminarProducto();  <<Override>> BuscarProducto(); | |  | |  | | --- | | Correaje | | String : Herraje; | | <<Override>> ConstruirProducto();  <<Override>> ModificarProducto();  <<Override>> EliminarProducto();  <<Override>> BuscarProducto(); | |
| |  | | --- | | Carriel | | String : Herraje; | | <<Override>> ConstruirProducto();  <<Override>> ModificarProducto();  <<Override>> EliminarProducto();  <<Override>> BuscarProducto(); | | | |

Tabla 2: Patrón Composite

En el anterior diagram se puede evidenciar la implementación del patrón de diseño composite o compuesto, en este el objeto producto es abstracto y heredan algunos de los productos que se comercializan en la talabartería, la super clase implementa la interfaz “ProductoInterface” para la reutilización y representación de características especificas de algunos productos y acciones que realiza cada uno de estos.

### Consecuencias

El uso de este patrón en el proyecto de la Talabartería Montenegro tendrá las siguientes consecuencias enumeradas acontinuación:

* Mantenibilidad de código a futuro
* Vuelve el código reusable
* Permite jerarquía de objetos complejas
* Es posible añadir nuevos componentes fácilmente
* El diseño se vuelve algo general

## Patrón de diseño Cadena de responsabilidad

El patrón de diseño Cadena de responsabilidades permite establecer una cadena de objetos receptores a través de los cuales se pasa una petición formulada por un objeto emisor. Cualquiera de los objetos receptores puede responder a la petición en función de un criterio establecido.

### Problemas identificados

|  |  |
| --- | --- |
| **Problema de diseño** | **Solución** |
| Las acciones emitidas por parte del usuario deben ser atendidas por distintos tipos de objetos. | Aplicar el patrón cadena de responsabilidad para permitir el tipo de objeto que debe atender la acción. |
| Un conjunto de objetos, en este caso productos, debe responder una acción, y esta se debe especificar dinámicamente. | Usar el patrón cadena de responsabilidad para evitar el uso excesivo de condicionales en el código fuente. |

### Diagrama de clase

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Usuario | | |  | | --- | | Manejador | |  | | ManejarAccion(); | |  |
| |  | | --- | | ManejadorModificacion | |  | | ManejarAccion(); | |  | |  | | --- | | ManejadorBusqueda | |  | | ManejarAccion(); | |

Tabla 3: Patrón Cadena de responsabilidad

En el diagrama anterior se puede ver que el usuario interactua con el manejador, para manejar donde se esta manejando el evento, y asi dirigir la acción que desea realizar el usuario, en el caso de la interfaz de la talabartería, se puede aplicar a modificacion y búsqueda de un producto.

### Consecuencias

El uso de este patrón en el proyecto de la Talabartería Montenegro tendrá las siguientes consecuencias enumeradas acontinuación:

* Añade Flexibilidad en la asignación de responsabilidades de los objetos. Se puede variar la asignación de responsabilidades añadiendo nuevos manejadores o modificándolos.
* La recepción no esta asegurada. Si la cadena no esta bien configurada puede no cubrir todas las peticiones.
* Reduce el acoplamiento. El patrón permite que un objeto envie una petición y sepa que va a ser tratada correctamente, pero tanto el receptor como el emisor no conocen nada el uno del otro.

# Patrones de Arquitectura, Usabilidad y análisis

## Patrón de aquitectura Active Record

El patrón Active Record facilita la creación y el uso de objetos de negocio cuyos datos requiere un almacenamiento persistente a una base de datos, este patrón es encontrado en las bases de datos relacionales. Active record es un enfoque para acceso de datos en una base de datos. Una tabla de la base de datos o vista (view) está envuelta en una clase. Por lo tanto, una instancia de un objeto está ligada a un único registro (tupla) en la tabla. Después de crear y grabar un objeto, un nuevo registro es adicionado a la tabla. Cualquier objeto cargado obtiene su información a partir de la base de datos. Cuando un objeto es actualizado, un registro correspondiente en la tabla también es actualizado. Una clase de envoltura implementa los métodos de acceso (setter e getter) o propiedades para cada columna en la tabla o vista. Su implementación es encontrada regularmente en Frameworks de desarrollo de software, para el caso del proyeto de la Talabartería Montenegro, se utilizará el ambiente de programación Ruby On Rails.

Para la implementación de este patrón se tomará como ejemplo los productos de la talabartería, estos estarán representados con la clase Producto, que posee los atributos nombre, tipo, valor, entre todos, este patrón al ser implementado, creará cada registro en la base de datos por cada objeto que se agregue en la aplicación, tomando los atributos como columnas de la tabla. De la misma forma una clase puede ser usada para hacer una consulta de los productos en la base de datos, esta consulta creará un objeto tipo producto con las características del registro de la base de datos. En la siguiente **Tabla 4: Active Record** se puede evidenciar esta implementación del patrón Active Record en el proyecto utilizando como referencia la clase producto.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | << Active Record >>   |  | | --- | | Nombre | | Tipo | | Valor |  |  | | --- | | Insertar | | Modificar | | Buscar | | Eliminar | | | |  |  |  |  | | --- | --- | --- | --- | | << Tabla >>  Producto   |  | | --- | | Nombre | | Tipo | | Valor | | |

Tabla 4: Active Record

## Patrón de usuabilidad

“Los patrones de usabilidad representan varios mecanismos, por ejemplo, deshacer, cancelar, múltiples idiomas, etc., que mejoran la usabilidad del sistema final y que tienen un efecto en el diseño del sistema software donde se implante. Junto con estos patrones de usabilidad, también se presentan posibles soluciones de diseño para incorporar los mecanismos de usabilidad correspondientes en el diseño del sistema software”

En la el caso del proyecto de la Talabartería Montenegro, se desea aplicar los patrones de Usabilidad en los siguientes ítems:

* Subida de productos a la página principal por parte del Administrador: El sistema contara con la opción de cancelar cambios realizados en la sección de productos
* Manejo de múltiples idiomas: La página web de la talabartería Montenegro, contará con opción de mostrar su contenido en distintos idiomas, esto lo definirá el Administrador de acuerdo al público al que se lo desee mostrar
* Ayuda en la búsqueda de productos: En la sección de productos, se implementara una pestaña de búsqueda que permitirá autocompletar o dar varias opciones al cliente que interactúa con la página

## Patrón de análisis

Estos patrones están más orientados a la fase de análisis y toma de requerimientos. Cuando uno trabaja como Analista, realiza una toma de requerimientos apoyándose en los casos de uso que el Cliente nos comunica.

Con este patrón de análisis se realizó el levantamiento de requerimientos, en el cual se utilizaron los modelos establecidos por el cliente, los cuales fueron fundamentales para la toma de decisiones.

Con estos modelos también fue posible identificar las necesidades del cliente, con las cuales nos basamos para exponer una buena solución al problema que está enfrentando actualmente.

Después de llegar a un acuerdo con el cliente y poder establecer los requerimientos necesarios para este proyecto se procedió a la etapa de análisis de desarrollo.

Ya sabemos que tenemos que hacer, ahora nos toca buscar la forma de cómo hacerlo.

Lo que buscamos aplicar a nuestro proyecto en la parte de desarrollo es poder dar una solución óptima, que permita adecuarse a las necesidades del cliente, además que no sea muy complicado de manejarlo, ya que nuestro cliente no cuenta con experiencia a la hora de manejar artículos electrónicos, se busca entregar un producto fácil de manejar, rápido, eficiente y con una interfaz agradable.