

SA-KARATEDO

Documento de Arquitectura del Software

<Versión 1.0>

**Historial de Revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor(es)** |
| 20-12-2015 | 1.0 | Versión Inicial | Minerva Verdugo  Sandra Villamizar  Teddy Sears |

Tabla de contenido

[1. Introducción 4](#_Toc440379624)

[2. Propósito 4](#_Toc440379625)

[2.1 Alcance 5](#_Toc440379626)

[2.2 Personal involucrado 5](#_Toc440379627)

[2.3 Definiciones, acrónimos y abreviaturas 5](#_Toc440379628)

[2.4 Referencias 5](#_Toc440379629)

[2.5 Resumen 6](#_Toc440379630)

[3. Representación Arquitectónica 6](#_Toc440379631)

[4. Metas de la Arquitectura y Restricciones 8](#_Toc440379632)

[4.1 Metas 8](#_Toc440379633)

[4.2 Restricciones 8](#_Toc440379634)

[5. Vista Lógica 9](#_Toc440379635)

[5.1 Vista de Paquetes 9](#_Toc440379636)

[5.2 Diagrama de Clases: 9](#_Toc440379637)

[5.3 Vista de Casos De Uso 9](#_Toc440379638)

[5.4 Casos de Uso 9](#_Toc440379639)

[5.5 Diagramas de Casos de Uso 9](#_Toc440379640)

[5.6 Diagramas de Secuencia 9](#_Toc440379641)

[5.7 Vista de Despliegue 10](#_Toc440379642)

[5.8 Vista de Implementación 10](#_Toc440379643)

[6. Vista de Datos 11](#_Toc440379644)

[7. Diagramas 11](#_Toc440379645)

[7.1 Diagrama E-R de la base de datos 11](#_Toc440379646)

[7.2 Diagrama de Clases Módulo 16 11](#_Toc440379647)

[8. Vista de Seguridad 11](#_Toc440379648)

[9. Calidad 12](#_Toc440379649)

[9.1Requerimientos No Funcionales (RNF) 12](#_Toc440379650)

[9.1.1 Desempeño: 12](#_Toc440379651)

[9.1.2 Seguridad: 12](#_Toc440379652)

[9.1.3 Robustez: 12](#_Toc440379653)

[9.1.4 Mantenibilidad: 12](#_Toc440379654)

[9.1.5 Portabilidad: 12](#_Toc440379655)

[9.1.6 Disponibilidad: 13](#_Toc440379656)

[10. Estándares 13](#_Toc440379657)

[11. Tamaño y Performance 13](#_Toc440379658)

[12. Vistas 15](#_Toc440379659)

[12.1 Vista Lógica 15](#_Toc440379660)

[12.2 Vista de Paquetes 15](#_Toc440379661)

# 1. Introducción

El documento de arquitectura del software para el Sistema de gestión de documentos de análisis SA-KARATEDO, muestra el diseño arquitectónico del proyecto y el conjunto de partes en las que el sistema se dividió para asegurar un sistema más robusto, entendible y escalable, teniendo que cada división va destinado a un grupo de requerimientos funcionales que se deben de cumplir para la completitud del proyecto. El documento sirve de marco de referencia para guiar la construcción del software y estableciendo la estructura, funcionamiento e interacción entre las partes.

# 2. Propósito

El propósito de este documento es mostrar una visión global de la arquitectura usada para el desarrollo del software, así como las diferentes capas de: Datos, Lógica, Dominio, Excepciones, Interfaz y Pruebas Unitarias, así como también las herramientas seleccionadas y estándares.

## 2.1 Alcance

El alcance va vinculado al documento de especificación de requerimientos que describe los requerimientos del sistema y con ello se proporciona una vista arquitectónica que cumpla con todos los requerimientos del documento anterior.

## 2.2 Personal involucrado

|  |  |
| --- | --- |
| **Nombres** | Minerva Verdugo, Sandra Villamizar y Teddy Sears. |
| **Rol** | Grupo 16. |
| **Responsabilidades** | Módulo de Gestión de Compra |

## 2.3 Definiciones, acrónimos y abreviaturas

|  |  |
| --- | --- |
| **Nombre** | **Descripción** |
| **ERS** | Especificación de Requerimientos de Software |
| **DAS** | Documento de Arquitectura de Software |
| **Usuario** | El usuario es empleado de SA-KARATEDO y tiene cierto rol en la empresa de desarrollo |
| **Administrador** | El administrador es un empleado de SA-KARATEDO y tiene un rol de administrador del sistema |
| **Vista** | Es una representación de un área de interés o perspectiva del sistema en alto nivel. |

## 2.4 Referencias

|  |  |  |  |
| --- | --- | --- | --- |
| **Referencia** | **Título** | **Ruta** | **Autor** |
| **IEEE** | Standard IEEE 830 - 1998 | http://ayudantiasubb.files.wordpress.com/2008/06/ieee.pdf | IEEE |
| **ERS** | Especificación de Requerimientos del Software | https://github.com/rosmantorres/sakaratedo | Grupo de Desarrollo SA-KARATEDO |
| **Programación en capas** | Arquitectura de N Capas | http://jtentor.com.ar/post/Arquitectura-de-N-Capas-y-N-Niveles.aspx | Julio Tentor |
| **W3C** | Estándares Webs | http://www.w3c.es/estandares/ | W3C |

## 2.5 Resumen

Este documento explica la arquitectura del software utilizada para el desarrollo del sistema de documentación de software SA-KARATEDO. Describiendo las vistas bajo las cuales el sistema se ha implementado, utilizando diagramas de casos de uso, diagramas de secuencia y cualquier artefacto que facilite dicha descripción. Explicando también como las vistas soportan los requerimientos funcionales y no funcionales que tienen alto impacto en las decisiones de la arquitectura del software.

# 3. Representación Arquitectónica

El sistema fue diseño usando una arquitectura de “n” capas, en este caso fueron una capa de Datos, una capa de Lógica de Negocios, una capa de Interfaz, una capa de Dominio, una capa de Excepciones y una capa de Pruebas Unitarias, aunque esta última funciona como un proyecto aparte que prueba el código del sistema principal, no como una capa funcional del sistema. La capa de Datos es aquella que está encargada de buscar información en la base de datos, cada módulo definió su propia clase para implementar métodos que construían el query para buscar la información necesaria de la base de datos, se estableció una clase que era la encargada de manejar la conexión y de ejecutar los métodos, también se establecieron las clases Parámetro y Respuesta para el manejo de los parámetros de los query y respuesta para almacenar la información extraída de la base de datos. La capa de Lógica de Negocio, es la encargada de manejar la lógica de la aplicación, es la que hacia las llamadas a las clases en la capa de Datos y es la que usaba esa información para realizar las operaciones necesarias de la aplicación, métodos como los de Encriptar, Desencriptar y Enviar Correo se encuentran en esta capa. Al igual que la capa Datos, la capa Lógica estaba separada por módulos, de tal manera que cada módulo tiene al menos una clase en la capa lógica. La capa de la interfaz es la encarga de mostrar la información a través de una interfaz de usuario, en este caso en una página web, y es la encargada de comunicarse con la capa de Lógica. La capa de interfaz debía tener la menor cantidad de código posible, ya que cualquier operación que quiera hacer se debería hacer desde la capa lógica. Las capas de Dominio y de Excepciones, son conocidas por todo el proyecto, de manera que tanto la Interfaz, como la capa Lógica y la capa de Datos conocen de ambas, ya que en las 3 capas se implementa el manejo de objetos del Dominio y el manejo de excepciones. En la capa de Dominio se definen todas las clases, con sus atributos y métodos que cubren los requerimientos del proyecto. En la capa de Excepciones, también separada por modulo, se consiguen las Excepciones de Lógica del proyecto con los posibles errores que podrían ocurrir en tiempo de ejecución y así capturarlos y controlar lo que sucede luego con el sistema.

# 4. Metas de la Arquitectura y Restricciones

La arquitectura se escogió de esta manera para tener un mejor control y orden de código, además de poder distinguir y distribuir las responsabilidades del sistema, también al hacer el proyecto de esta manera garantizamos escalabilidad, usando un código que es fácil de entender y fácil de agregar nuevas funcionalidades y también se consigue un código que es fácil de mantener. Con la delimitación de las responsabilidades conseguimos un proyecto con alta cohesión y bajo acoplamiento.

## 4.1 Metas

La meta principal por la cual se estructuró el proyecto de esta manera es la de conseguir un diseño modular que dividiera las responsabilidades del proyecto en diferentes capas de tal manera que la capa de Interfaz solo conozca la capa Lógica, la capa Lógica conoce la capa de Datos solamente, y la capa de Datos se conecta con la base de datos.

## 4.2 Restricciones

Las restricciones que este tipo de diseño implica es la inhabilidad de poder comunicar dos capas que no corresponden, por ejemplo: la capa lógica no debería conocer la capa de interfaz y la capa de interfaz no debería conocer la capa de datos ni viceversa, lo cual implica que en algunos casos se tenga que extender el tiempo necesario para realizar algunas funcionalidad, lo cual lo contrarresta el hecho de que la arquitectura ofrece una capacidad para la reutilización de código, lo cual al final puede disminuir el tiempo necesario para realizar las funciones.

# 5. Vista Lógica

## 5.1 Vista de Paquetes

A continuación se encuentra un diagrama de los paquetes de alto nivel del proyecto SA-KARATEDO y su relación en el proyecto, como vemos los paquetes pueden darnos una vaga idea de las capas utilizadas en arquitectura de la solución.

## 5.2 Diagrama de Clases:

El diagrama de clases asociado al Módulo 16 del proyecto SA-KARATEDO se encuentra en los anexos.

## 5.3 Vista de Casos De Uso

## 5.4 Casos de Uso

## 5.5 Diagramas de Casos de Uso

## 5.6 Diagramas de Secuencia

Diagrama de Secuencia de Ver Carrito.

Diagrama de Secuencia de Eliminar Item.

Diagrama de Secuencia de Registrar pago.

Diagrama de Secuencia de Consultar Producto.

Diagrama de Secuencia de Detallar Producto.

Diagrama de Secuencia de Consultar Eventos.

Diagrama de Secuencia de Detallar Evento.

Diagrama de Secuencia de Agregar a Carrito.

Diagrama de Secuencia de Consultar Matriculas Morosas.

## 5.7 Vista de Despliegue

El sistema está hecho de tal forma de que las capas de Interfaz, Lógica y Datos puedan funcionar desde lugares diferentes con tal de que exista un web service que permita la conexión entre Interfaz y Lógica y entre Lógica y Datos, para lo cual se tendrían que hacer pocas modificaciones de código, lo ideal siendo que todas las capas se encuentren un mismo servidor. Cuando el servidor acceda a través del navegador web, este se debe conectar con la interfaz, y la capa de Datos debe tener acceso a una base de datos, la cual debería estar en un servidor aparte.

## 5.8 Vista de Implementación



# 6. Vista de Datos

# 7. Diagramas

## 7.1 Diagrama E-R de la base de datos

## 7.2 Diagrama de Clases Módulo 16

# 8. Vista de Seguridad

Como medidas de seguridad se implementó el uso de expresiones regulares para el filtrado de datos de ingreso, específicamente en los campos que requieren el ingreso de correo. Otra medida que se implementó, es establecer un máximo de caracteres posibles en los campos de inputs, se establecieron métodos de encriptamiento con Rijndael para el paso de datos por el URL, encriptado con hash para el almacenamiento de contraseñas en la base de datos, y por último los querys de base de datos fueron hechos con stored procedures parametrizados. Con todo lo anterior podemos garantizar que el sistema desarrollado tiene protección contra ataques cross-scripting, con el filtrado de datos, también se asegura el ataque con SQL-Injection al hacer querys parametrizado y ataques a fuerza bruta con el encriptado de los datos.

# 9. Calidad

## 9.1Requerimientos No Funcionales (RNF)

### 9.1.1 Desempeño:

La aplicación garantizará que los usuarios tendrán una eficiencia de un 98% al usar la aplicación.

### 9.1.2 Seguridad:

* La arquitectura de desarrollo planteada para el sistema de n capas brinda más seguridad.
* El sistema tiene implementadas medidas de seguridad que lo protegen contra ataques comunes al mismo.

### 9.1.3 Robustez:

A través del manejo de excepciones y errores el sistema se comporta de manera robusta, lo que convierte al sistema en un sistema tolerante a fallas.

### 9.1.4 Mantenibilidad:

Gracias a la arquitectura de desarrollo planteada para el sistema el mantenimiento perfectivo y adaptativo, se puede realizar de una manera muy sencilla.

### 9.1.5 Portabilidad:

El sistema SA-KARATEDO corre basado en Web lo que hace que pueda ser utilizado desde cualquier dispositivo que tenga conexión a la red y corra un explorador como Chrome, Firefox o Safari convirtiéndola en una aplicación multiplataforma.

### 9.1.6 Disponibilidad:

La disponibilidad del sistema estará sujeta a la disponibilidad del servidor de producción en donde se aloje.

# 10. Estándares

|  |  |  |
| --- | --- | --- |
| Tipo | Estándar | Ejemplo |
| Clase | PascalCase | Carrito |
| Interfaz | PascaCase, Inicio con ‘I’ | ICarrito |
| Excepción | PascalCase, termina con ‘Exception’ | CarritoException |
| Métodos | PascalCase | ToString() |
| Parámetro | Camel | CalcularValor(Valor valosConseguido) |
| Atributos | Camel | nombreCarrito |
| Variables | Camel | Int contadorCarrito |
| Constantes | Mayúsculas | Final String archivo XML\_PATH |
| Enumeración | PascalCase | CantidadDeItem |

**Acrónimos:**

La primera Letra en mayúscula: Ejemplo: DaoCarrito

# 11. Tamaño y Performance

El sistema SA-KARATEDO está previsto para ser utilizado por pocos usuarios, aunque la arquitectura diseñada permite el crecimiento de la aplicación a grandes escalas. Todo dependerá del equipo donde se aloje la base de datos y la aplicación.

Del lado del cliente como es una aplicación web el crecimiento puede ser masivo y el manejo de la concurrencia en la base de datos está basado en el manejador SQL Server, que permite a los usuarios trabajar de manera continua.

Se requise que el equipo donde será instalada la aplicación tenga al menos 40mb de espacio en disco duro y 4gb de RAM para correr los servicios necesarios.

# 

# 12. Vistas

## 12.1 Vista Lógica

## 12.2 Vista de Paquetes

A continuación se encuentra un diagrama de los paquetes de alto nivel del proyecto SA-KARATEDO y su relación en el proyecto, como vemos los paquetes pueden darnos una vaga idea de las capas utilizadas en arquitectura de la solución.