****

**DOCUMENTO DE DISEÑO DE SOFTWARE (SDD)PARA SISTEMA DE CONTROL DE INVENTARIO DIRIGIDO A LOS ALMACENES TIA**

**UNIVERSIDAD DE GUAYAQUIL**

**FACULTAD**

**CARRERA DE SOFTWARE**

**Autores:**

CHAVARRIA BARRE JEAN CARLOS

GARCIA MORALES JORDAN STEVEN

GUEVARA ESPINOZA JOSUE ANTONIO

GUTIERREZ SUAREZ RONALD JOSEPH

VALENCIA MOYANO IGNACIO ANDRES

VARGAS CASTAÑEDA FELIX ARTURO

**AÑO LECTIVO**

2023 – 2024

**INDICE**

**CONTROL DE VERSIONES DEL DOCUMENTO**

|  |  |  |  |
| --- | --- | --- | --- |
| VERSIÓN | MODIFICADO POR | FECHA DE MODIFICACIÓN | OBSERVACIÓN |
| 1.0.0 | Félix Castañeda | 8/Noviembre/2023 | CREACIÓN |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**INTRODUCCIÓN**

En la actualidad la complejidad de los sistemas de software ha ido en aumento. Los requerimientos del usuario son cada vez mas complejos y extensos, lo cual ha obligado a que los procesos de desarrollo sean mas completos y mejor estructurados, para que puedan asegurar la obtención de un producto que satisfaga los requerimientos en un tiempo de desarrollo aceptable, y que garantice la calidad total del producto.

Dentro del proceso de desarrollo de sistemas, una de las etapas más críticas es la etapa de diseño, ya que en esta etapa se conceptualiza la estructura, organización y propiedades que tendrá la solución a un problema o requerimiento planteado;

Además, esta etapa servirá como punto de partida para etapas futuras, como la construcción e implementación del sistema. Sin embargo, muchas empresas no realizan un proceso de diseño formal, lo cual puede generar un producto mal estructurado, que en el futuro genere costos muy altos en cambios y adecuaciones.

Considerando que al realizar adecuadamente la etapa de diseño s pueden ahorrar costos en tiempo y esfuerzo en etapas posteriores, se ha planteado un procedimiento estándar para la elaboración de esta etapa. Este procedimiento abarca las áreas y tópicos mas importantes del diseño. Mediante una serie de pasos detallados se puede obtener la arquitectura del sistema, y refinarla sucesivamente hasta obtener el diseño detallado; también se considera el diseño del almacenamiento de datos y la obtención de interfaces de usuario, en todas estas actividades se evalúa la calidad del producto mediante métricas propias de esta etapa.

Como resultado al proceso de diseño se obtiene el presente proyecto que plantea una propuesta que describirá el diseño del sistema de control de inventario dirigido para los almacenes TIA.

## Vista Lógica del Sistema

## Arquitectura de software

Según (Blas, Leone , & Gonnet, 2019) “la arquitectura de un producto de software comprende los componentes del software, las propiedades visibles de dichos componentes y las relaciones que existen entre ellos”.

La arquitectura de software proporciona un alto nivel de abstracción para administrar la complejidad del sistema y establecer mecanismos de comunicación y coordinación entre los componentes de diseño, codificación y administración de los datos.

“Su finalidad es brindar solución a un sistema en la parte de la fase de diseño y de requerimientos, ya que reúne todos los casos técnicos y operacionales que son difíciles de cambiar durante el proceso de desarrollo” (Navarro, Aranda, & Moreno, 2018).

### *Arquitectura por capas*

“La arquitectura por capas representa la funcionalidad del sistema organizada en capas separadas e independientes, y cada una se apoya sólo en las peticiones percibidas por la capa de presentación” (Fiallo Viscaino, 2022).

Esta arquitectura consta de tres niveles: nivel de datos o de persistencia, nivel lógico o de negocio, nivel de presentación. Cada una de estas capas funciona de manera diferente, pero tiene un objetivo común, que es mantener el flujo de peticiones de la aplicación.



**Tabla 7.** Niveles de Arquitectura de tres capas

|  |  |
| --- | --- |
| **Niveles de Arquitectura de tres capas** | |
| **Presentación** | Se enfoca en la interfaz de usuario donde se visualiza la funcionales del aplicativo, permitiendo realizar peticiones con la finalidad de obtener un resultado. |
| **Negocio/Lógica** | Mediante las peticiones realizadas por la capa de presentación está realizara el proceso de ejecución mediante sus métodos, realizando validaciones y enviando resultados. |
| **Datos/Persistencia** | Almacena y consultadas los datos percibidos mediante la petición del cliente por parte de la capa de presentación y la lógica de negocios. |

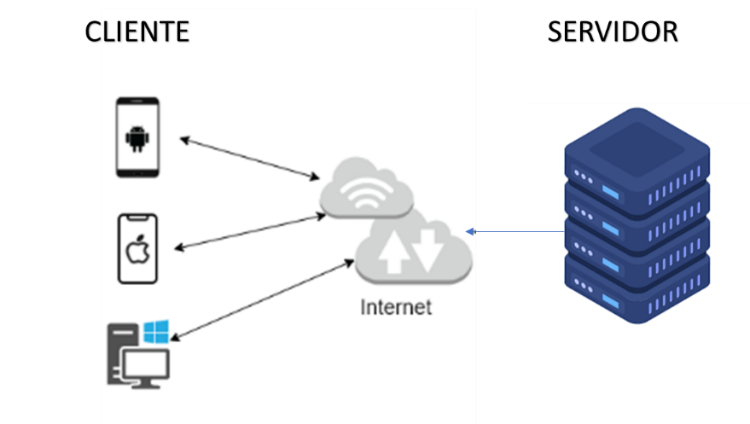
### *Arquitectura Cliente – Servidor*

La arquitectura cliente servidor es un modelo muy utilizado en la actualidad que sirve como proveedor de información, suministrando los recursos a disposición del cliente.

“Esta arquitectura se compone de la capa de clientes, quien se encarga de realizar peticiones por medio de la interfaz, comunicándose con la capa de servidor esperando recibir consultas que fueron solicitadas por el usuario” (Matute Curillo & Yunga Tacuri, 2022).

La arquitectura cliente/servidor funciona cuando la computadora cliente envía una solicitud de recurso o proceso al servidor a través de la conexión de red, que luego se procesa y entrega al cliente. Una computadora servidor puede administrar varios clientes simultáneamente, mientras que un cliente puede estar conectado a varios servidores a la vez, cada uno proporcionando un conjunto diferente de servicios.

Al no distribuir los recursos puede ocasionar que se caiga el servidor por demanda propia, la idea es que se pueda acceder a ellos bajo cierta cantidad de peticiones balanceando las conexiones si necesidad de sobresaturar el servidor.



## Patrones de diseño

Los patrones de diseño son mejores prácticas formalizadas que el programador puede usar para resolver problemas comunes al diseñar una aplicación o sistema, además, permite el código desarrollado sea reutilizable.

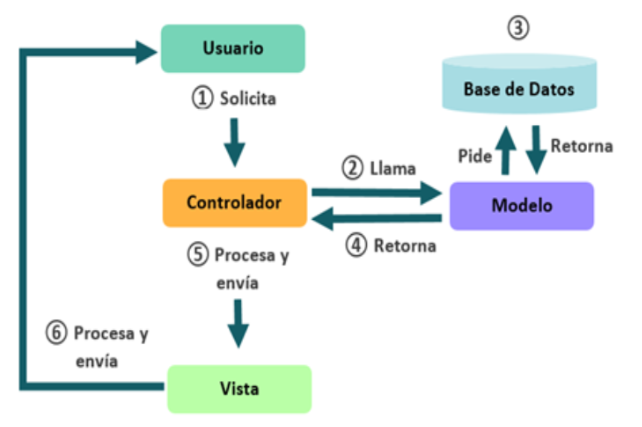
Según (Rivas, 2021), “cada patrón describe un problema que ocurre una y otra vez, describiendo el núcleo de la solución de ese suceso, aparte son independientes de cualquier lenguaje de programación, ya que no se limitan a un lenguaje de programación específico”.

Actualmente existen varios patrones de diseños, pero cada uno de ellos concurre para cumplir un propósito específico, ya sea para transmitir eventos de una aplicación a otra o para consumir mensajes de la aplicación a medida que están disponible, permitiendo corregir errores de código dando estabilidad de desarrollo al mismo.

### *Modelo Vista Controlador*

Es un patrón arquitectónico utilizado en el desarrollo de software. Por lo general, se usa en aplicaciones web y es una forma de separar las correctamente la funcionalidad de cada componente en diferentes partes de una aplicación, que se compone de tres partes: el modelo, la vista y el controlador

Según (Acosta Espinoza, 2022) “el patrón MVC permite crear aplicaciones de calidad, este modifica cada uno de sus componentes sin necesidad de afectar a los demás”.



Como se observa en la **Figura 9**, el funcionamiento de este patrón inicia desde la petición que realiza un usuario desde una interfaz, donde consigo se ejecuta el controlador con la disposición de la lógica de negocios haciendo un llamado al modelo o entidad que tiene relación desde la base de datos por medio de una consulta o ingreso, este retorna con los datos siendo procesados para enviarse nuevamente a la vista y que pueda ser visualizado por el usuario.

**Vista Física del Sistema – Fase de Diseño**

Para el éxito de este proyecto se requiere los siguientes diagramas:

* Modelo de diagrama de clases
* Modelo de entidad relación
* Diagrama de secuencia
* Diagrama de despliegue.

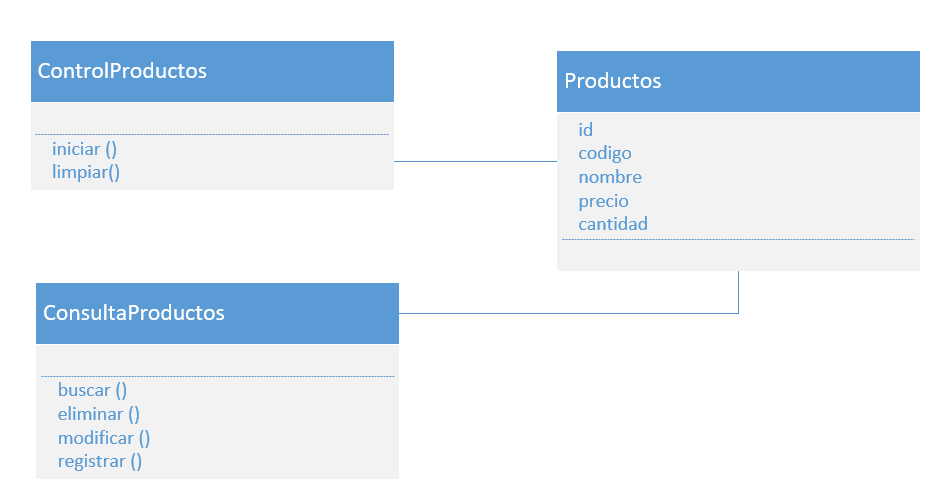
### *Modelo del Diagrama de Clases*

A continuación, se visualiza las relaciones de las entidades más destacadas dentro del modelo lógico del desarrollo del proyecto.



### *Modelo Entidad Relación*

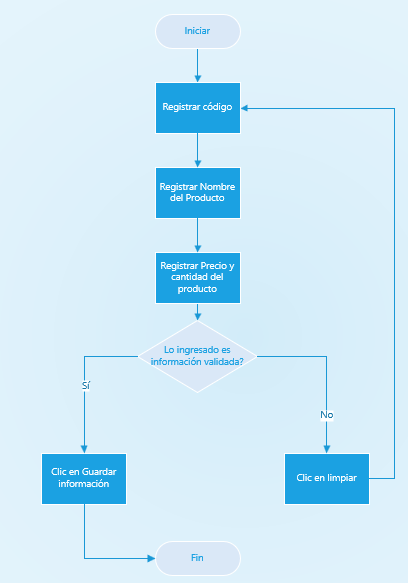
A continuación, el modelo entidad relación del proyecto que facilita la representación de las entidades de una base de datos.



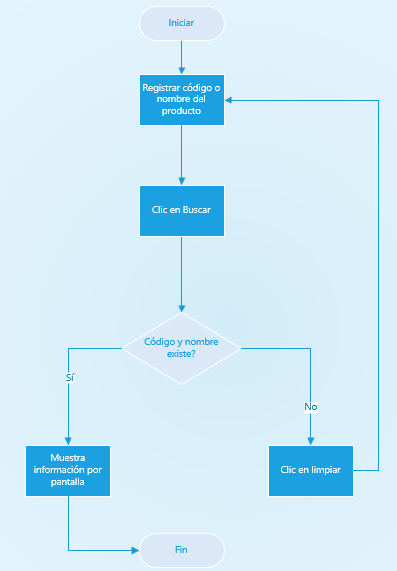
## Diagrama de actividades

Los siguientes diagramas de actividades son diseñados con el fin de poder visualizar de forma más sencilla la lógica del negocio y tener una idea más clara al momento de desarrollar el sistema.

**Diagrama de registro de productos**



**Diagrama de consulta de productos**



## Diagrama de Despliegue

Realizar el diagrama de despliegue permite identificar componentes y una vista general de la topología del sistema, es decir, representar la distribución física de los componentes software.

