

|  |
| --- |
| Universidad del Cauca |
| Documento de Arquitectura de Software |
| shopping |
| Santiago Hyun Dorado  19-08-2018 |

**Historial de cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 19/08/2018 | 1.0 |  | Santiago Hyun Dorado |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Tabla de contenido**

[1. Introducción 6](#_Toc524762512)

[1.1. Propósito 6](#_Toc524762513)

[1.2. Alcance 6](#_Toc524762514)

[1.3. Definiciones, acrónimos y abreviaturas 6](#_Toc524762515)

[1.4. Referencias 7](#_Toc524762516)

[1.5. Resumen 7](#_Toc524762517)

[2. Objetivos de la Arquitectura 8](#_Toc524762518)

[3. Representación de la Arquitectura 8](#_Toc524762519)

[3.1. Vistas arquitecturales 8](#_Toc524762520)

[3.1.1. Vista de escenarios 9](#_Toc524762521)

[3.1.2. Vista lógica 9](#_Toc524762522)

[3.1.3. Vista de desarrollo 9](#_Toc524762523)

[3.1.4. Vista de proceso 9](#_Toc524762524)

[3.1.5. Vista física 9](#_Toc524762525)

[3.2. Patrones de diseño arquitectónicos 10](#_Toc524762526)

[3.3. Estilo arquitectural 11](#_Toc524762527)

[4. Descomposición de la arquitectura 12](#_Toc524762528)

[4.1. Vista de escenarios (casos de uso) 12](#_Toc524762529)

[4.2. Vista lógica (diseño) 16](#_Toc524762530)

[4.3. Vista de proceso (actividades) 19](#_Toc524762531)

[4.4. Vista de desarrollo (componentes) 19](#_Toc524762532)

[4.5. Vista física (despliegue) 20](#_Toc524762534)

[5. Bibliografía 21](#_Toc524762535)

**Tabla de Figuras**

[Figura 1 Modelo 4+1 vistas 6](#_Toc524763375)

[Figura 2 Patrón MVC 8](#_Toc524763376)

[Figura 3 Estilo arquitectural clásico 9](#_Toc524763377)

[Figura 4 Organización Capas y componentes MVC 10](#_Toc524763378)

[Figura 5 Diagrama de Casos de uso 11](#_Toc524763379)

[Figura 6 Diagrama de clases del componente Configuración 15](#_Toc524763380)

[Figura 7 Diagrama de clases del componente Pedidos 15](#_Toc524763381)

[Figura 8 Diagrama de clases del componente Productos 16](#_Toc524763382)

[Figura 9 Diagrama de clases del componente Tiendas 16](#_Toc524763383)

[Figura 10 Diagrama de clases del componente Usuarios 17](#_Toc524763384)

[Figura 11 Diagrama de secuencia Realizar pedido 17](#_Toc524763385)

[Figura 12 Diagrama de componentes 18](#_Toc524763386)

[Figura 13 Diagrama de despliegue 18](#_Toc524763387)

# Introducción

Este documento tiene como finalidad explicar en profundidad la organización fundamental de los componentes que dan origen a la funcionalidad requerida por los interesados en la construcción del software. En esta sección se define el propósito de este documento, el alcance, algunas definiciones, abreviaturas y acrónimos, y finalmente se muestra el resumen de las secciones posteriores.

## Propósito

La organización fundamental de los componentes de software es una de las primeras tareas que se deben realizar en el desarrollo de software junto con el análisis del negocio. Un buen análisis provee buenas bases para establecer un diseño que se adecúe de manera eficiente a las necesidades del negocio, es por eso que es de gran importancia entender el modelo de negocio y definir cuáles son los puntos en donde se requiere apoyo a nivel arquitectural para complementar la funcionalidad a nivel de calidad del producto. El propósito principal de este documento es preservar las decisiones realizadas a nivel de diseño relacionadas con la arquitectura del software, con el objetivo de tener una base de conocimiento que permita entender o recordar con más agilidad la organización de los componentes y las responsabilidades encargadas de representar la funcionalidad del software.

## Alcance

La definición de arquitectura de software a nivel general cuenta con un alto grado de abstracción, esta se define como la organización fundamental de los componentes que en conjunto forman un sistema. Debido a esto, la arquitectura de software se puede representar de diferentes maneras dependiendo de la perspectiva del observador. Desde el origen del desarrollo de software se pensaron y se estandarizaron algunos meta-modelos que permiten abstraer y modelar la arquitectura a través de diagramas, siendo uno de los lenguajes de modelado más conocidos y utilizados UML. La información contenida en este documento permite entender la estructura del sistema desde diferentes perspectivas, además, permite también describir explícitamente las razones de las decisiones arquitecturalmente importantes implementadas en el proceso de desarrollo. La representación de la arquitectura se expresa en el modelo 4+1 vistas ya que este permite observar la arquitectura desde diferentes perspectivas y para diferentes audiencias, además para cada vista de este modelo existen diagramas bien definidos que facilitan la descripción de cada una de ellas.

Este documento permanece en constante actualización debido a que el sistema se encuentra actualmente en desarrollo y no se ha generado ninguna versión para producción. Sin embargo, este artefacto brinda las bases para establecer la organización de los componentes y definir las decisiones de diseño que se toman respecto a las necesidades a nivel funcional y no funcional.

## Definiciones, acrónimos y abreviaturas

**SAD**: Software Architecture Document

**GUI (Graphic User Interface):** Interfaz Gráfica del Usuario

**POO:** Programación Orientada a Objetos

**ISO (International Organization for Standardization)**: Organización Internacional de Normalización.

**ARAT (Architectural Rationale Annotations Tool):** Es una herramienta software basada en anotaciones de código fuente para documentar las razones arquitecturales de un sistema desarrollado en Java.

**Java**: Es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.

**Stakeholder**: Según N. Rozanski et al [2] un stakeholder en una arquitectura de software “es una persona, grupo o entidad con intereses o preocupaciones sobre la realización de la arquitectura”.

**MVC (Model View Controller)**: Es uno de los patrones de diseño arquitectural más utilizado en el desarrollo web, empezó como un Framework desarrollado por Trygve Reenskaug alrededor de 1970 [3].

**OMG (Object Management Grpup)**: Es un consorcio internacional de estándares tecnológicos sin fines de lucro, de membresía abierta [4].

**UML (Unified Modeling Language)**: El lenguaje de modelado unificado (UML) es un lenguaje gráfico creado por el OMG para visualizar, especificar, construir y documentar los artefactos de un sistema de software [5].

## Referencias

DAS: <https://www.csun.edu/engineering-computer-science>

Java: <https://www.java.com/es/download/faq/whatis_java.xml>

Stakeholder: <https://www.viewpoints-and-perspectives.info/home/stakeholders/>

MVC: <https://msdn.microsoft.com/en-us/library/ff649643.aspx>

ISO: <https://www.iso.org/home.html>

OMG: <https://www.omg.org/>

UML: <http://www.uml.org/>

## Resumen

Este documento se compone de las siguientes secciones:

* Sección 1: Provee una introducción relacionada con la arquitectura de software del Juego N en línea.
* Sección 2: Describe los objetivos a nivel general de la Arquitectura del sistema.
* Sección 3: Describe la representación de la arquitectura en diferentes vistas.
* Sección 4: Desglosa la representación de la arquitectura en diagramas para cada vista.
* Sección 5: Referencias bibliográficas usadas en la creación de este documento.

# Objetivos de la Arquitectura

La arquitectura de este sistema va dirigida a cumplir con requerimientos funcionales que den cumplimiento a las necesidades del negocio expresadas por los interesados del proyecto. Uno de los principales objetivos de la arquitectura es conseguir una aplicación que se adapte a las restricciones y los recursos del proyecto, en este caso es de carácter obligatorio utilizar Java como lenguaje de desarrollo tanto a nivel de Back-End como a nivel de Front-End debido a que la zona en la que se desarrolla el proyecto la oferta de desarrolladores con conocimientos en este lenguaje de desarrollo y sus herramientas relacionadas es alta. Debido a lo anterior la arquitectura planteada se basa en tecnologías como EJB’s, JSF, Primefaces y JPA. Los EJB’s representan la capa de negocio, Java Server Faces junto con PrimeFaces permiten la interacción con el usuario a través de la capa de vista y finalmente con JPA se tiene una última capa a nivel de acceso a datos. Uno de los objetivos principales de la estructura actual de los componentes software es facilitar la transición de una arquitectura en un nivel, a una arquitectura multinivel orientada a microservicios. Con el objetivo de conseguir que el sistema pueda brindar portabilidad hacia futuro, exponiendo los servicios de manera transparente a las aplicaciones clientes que hagan uso de ellos.

# Representación de la Arquitectura

## Vistas arquitecturales

La representación del sistema se puede ver desde diferentes puntos de vista, en el Modelo 4+1 Vistas propuesto por Kruchten p. et al [6] se definen 5 perspectivas que permiten extender el entendimiento de la arquitectura a las diferentes personas involucradas en el desarrollo de un sistema. En la Figura 1 se puede observar la organización de las vistas y su relación.

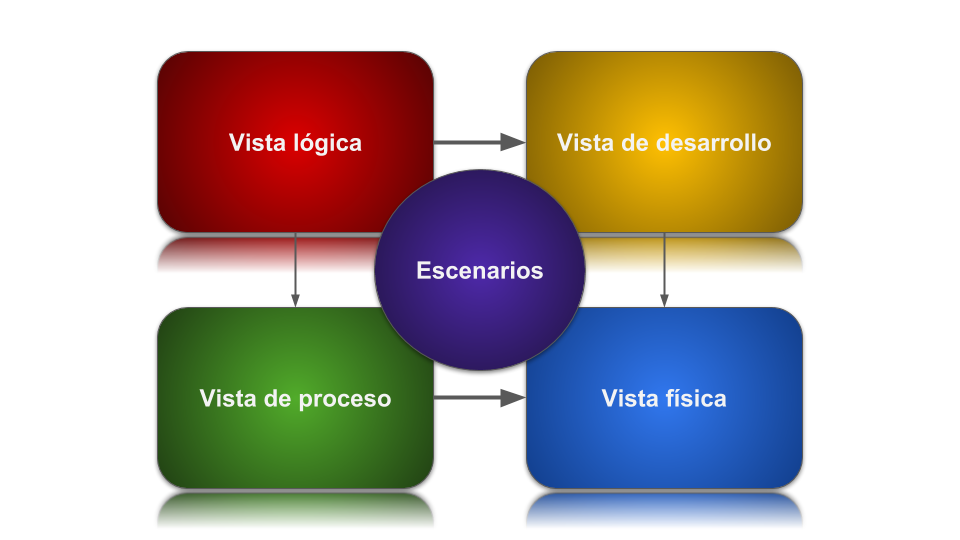


Figura Modelo 4+1 vistas

### Vista de escenarios

Conocida también como Vista de Casos de Uso, brinda información a nivel gráfico sobre los requerimientos funcionales de un sistema. Esta vista permite establecer las acciones y los roles que existen en el desarrollo de un sistema. Cada rol definido en el sistema debe tener una cantidad de casos de uso correspondientes a los requerimientos establecidos.

**Audiencia**: Stakeholders

**Artefactos relacionados**: Diagrama de Casos de Uso

### Vista lógica

La Vista Lógica describe un sistema mediante el paradigma orientado a objetos, esta vista muestra las abstracciones diseñadas para estructurar el sistema junto con sus atributos y las operaciones que definen su comportamiento. Además, muestra cómo se relacionan cada una de ellas para formar partes más grandes de un sistema complejo.

**Audiencia**: Diseñadores

**Artefactos relacionados**: Diagrama de clases

### Vista de desarrollo

Conocida también como Vista de Componentes, representa la arquitectura en términos de componentes software generalmente constituidos por una o más clases, estos componentes se relacionan entre sí para definir el comportamiento esperado por los usuarios finales. Estos componentes suelen ser software, sin embargo, se pueden dar casos donde los algunos de los componentes sean físicos o de hardware.

**Audiencia**: Desarrolladores

**Artefactos relacionados**: Diagrama de componentes, Diagramas de paquetes

### Vista de proceso

Esta vista permite observar las tareas individuales que realizan las abstracciones software para cumplir con un determinado proceso. Los requisitos del usuario se pueden comprender como tareas que un sistema debe cumplir, esta vista permite desglosar esas actividades en partes más pequeñas con el objetivo de determinar los procesos de manera gráfica y ordenada. La complejidad de los procesos determina el nivel de abstracción adecuado para realizar el diseño del modelo.

**Audiencia**: Integradores, Desarrolladores

**Artefactos relacionados**: Diagrama de actividades, Diagrama de secuencia

### Vista física

Es importante conocer la distribución de los componentes físicos que constituyen un sistema, la Vista Física permite observar la topología y las comunicaciones que se establecen entre los nodos físicos que conforman la arquitectura hardware del sistema.

**Audiencia**: Líderes de despliegue

**Artefactos relacionados**: Diagrama de despliegue

En la sección [Descomposición de la arquitectura](#_3j2qqm3) se muestran los diagramas utilizados en el Juego N en línea para cada tipo de vista.

## Patrones de diseño arquitectónicos

El patrón de diseño arquitectural implementado en el desarrollo del Juego N en línea es el patrón *Modelo Vista Controlador MVC*. Por medio de este patrón podemos realizar una separación de la de interacción del usuario con la Interfaz Gráfica en tres diferentes componentes. En la Figura 2 se puede observar la interacción entre los componentes del patrón.

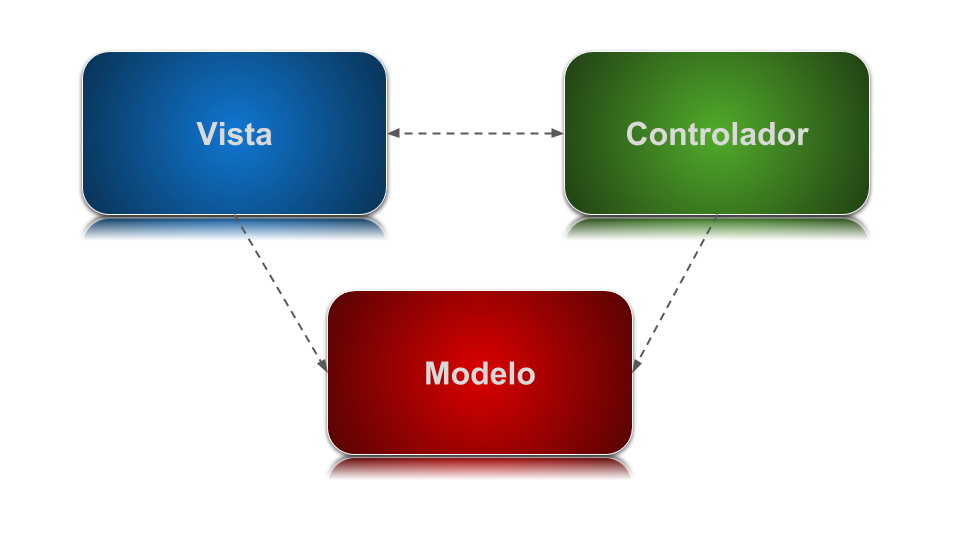


Figura Patrón MVC

**Modelo:** Representa la información sobre el dominio del negocio, los datos y el comportamiento del sistema.

**Vista**: Es un componente que representa la *Interfaz Gráfica del Usuario GUI* con los datos del modelo, este componente captura las acciones y los datos y acude al controlador adecuado para procesarlos.

**Controlador**: Captura los datos ingresados por el usuario, manipula el modelo y actualiza la vista de forma correcta.

En la sección [*Vista lógica*](#_1ci93xb)se muestra con más detalle las clases y los paquetes que componen este patrón de diseño arquitectural.

## Estilo arquitectural

Los estilos arquitecturales son soluciones para resolver un problema relacionado con requerimientos NO funcionales o atributos de calidad, cada estilo arquitectural representa la experiencia de los diseñadores en un dominio de un problema específico. Las necesidades de calidad cambian de un sistema a otro por lo tanto los estilos arquitecturales son muy diversos y es muy difícil que contemplen todos los atributos de calidad requeridos [7]. Para el desarrollo del sistema Shopping los requisitos NO funcionales de mayor prioridad son la portabilidad y la mantenibilidad. La portabilidad es la característica de un sistema de ejecutar las instrucciones en diferentes dispositivos y la mantenibilidad según la Norma ISO 25000 [8], es la facilidad para efectuar cambios, reutilizar componentes o agregar funcionalidad al sistema.

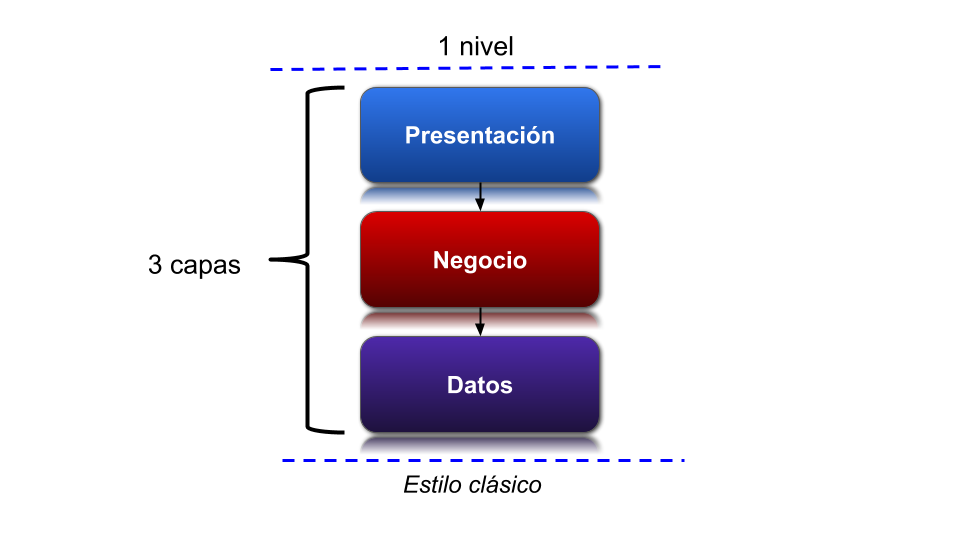


Figura Estilo arquitectural clásico

En la Figura 3 se puede observar que el Juego N en línea sigue el estilo clásico de un nivel y tres capas propuesto por G. Florijn et al [7].

**Capa de Presentación**: Se encarga de controlar el flujo de trabajo entre el usuario y la GUI, también es responsable por mostrar la información del modelo. En el Juego N en línea esta capa contiene los componentes *Vista* y *Controlador* del *Patrón MVC.*

**Capa de Negocio**: Define los atributos y el comportamiento de las abstracciones de software en términos del contexto del negocio. En esta capa se puede encontrar las entidades que representan el *Modelo* del *Patrón MVC*.

**Capa de datos**: Esta capa es la encargada de acceder a los datos y establecer la persistencia de la información en el sistema.

En la Figura 4 se puede observar la organización de las capas y los componentes del *MVC.*

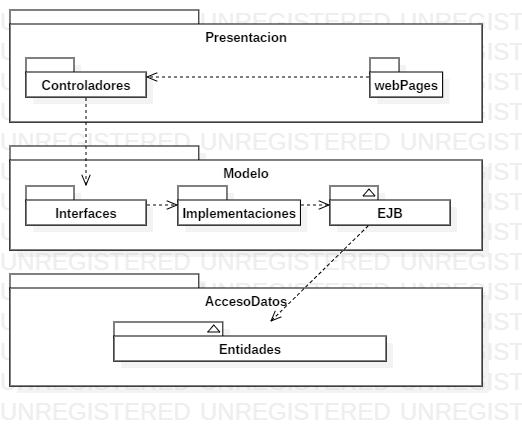
**

Figura Organización Capas y componentes MVC

# Descomposición de la arquitectura

La representación de la arquitectura se puede definir desde diferentes vistas adecuadas para una determinada audiencia. Estas vistas están relacionadas con uno o más diagramas UML que detallan con más información la arquitectura del sistema.

## Vista de escenarios (casos de uso)

Las funcionalidades a nivel general se describen en la Figura 5. El jugador en el contexto del juego real es la persona que utiliza un tablero con n filas por m columnas y una cantidad x de fichas con el objetivo de realizar una secuencia de fichas continua del mismo color. Esta persona es la encargada de ubicar la ficha en una columna del tablero y soltarla, la evaluación del ganador se hace de manera visual, rectificando que haya una cantidad x de fichas contiguas del mismo color.



Figura Diagrama de Casos de uso

|  |  |
| --- | --- |
| **ID** | CU01 |
| **Nombre** | Iniciar sesión |
| **Actores** | Usuario registrado |
| **Sinopsis** | Inicio de sesión en el aplicativo |
| **Pre-condición** | Tener usuario y contraseña activos en el sistema |
| **Curso típico de eventos** | 1. Digitar nombre de usuario 2. Digitar contraseña 3. Ingreso al sistema |
| **Cursos alternativos** | En caso de que el sistema valide los datos ingresados por la persona y no encuentre ningún usuario con esas credenciales se debe notificar al usuario por medio de un mensaje. |
| **Extensiones** | Ninguno |
| **Prioridad** | Media |
| **% Completado** | 100 |

|  |  |
| --- | --- |
| **ID** | CU02 |
| **Nombre** | Gestionar tiendas |
| **Actores** | Configurador |
| **Sinopsis** | Configurar la información de las diferentes tiendas(sucursales) de la empresa |
| **Pre-condición** | Debe haber iniciado sesión (CU01) |
| **Curso típico de eventos** | 1. Después de que el configurador haya iniciado sesión se muestra la pantalla principal con el menú de opciones en la parte izquierda. 2. El configurador selecciona en el menú configurar tienda y se despliega una lista de tiendas asociadas al sistema 3. El configurador tiene las opciones de Crear, Editar, Eliminar o Ver la información de las tiendas. |
| **Cursos alternativos** | El configurador decide no realizar ningún cambio y cierra sesión mediante un botón ubicado en la parte superior derecha de la pantalla. |
| **Extensiones** | Crear tienda, Editar tienda, Buscar tienda. |
| **Prioridad** | Baja |
| **% Completado** | 100 |

|  |  |
| --- | --- |
| **ID** | CU03 |
| **Nombre** | Gestionar Usuarios |
| **Actores** | Administrador |
| **Sinopsis** | Gestionar los usuarios que acceden al sistema, asignarles permisos y definir el rol respectivo para cada usuario. |
| **Pre-condición** | Se debe haber iniciado sesión (CU01) |
| **Curso típico de eventos** | 1. Después de iniciar sesión el sistema muestra la pantalla principal, en la que se muestra el menú de opciones en la parte izquierda. 2. Usuario selecciona la opción ‘Gestionar Usuarios’ y se despliega una lista con las opciones de ‘Gestionar roles’, donde se definen los roles que hay en el sistema, la ‘Gestión de permisos’ donde se asignan los permisos a los cuáles cada usuario tiene acceso y finalmente la ‘Gestión de usuarios’ donde se Agregan, Modifican, Visualizan o Eliminan a los usuarios del sistema. |
| **Cursos alternativos** |  |
| **Extensiones** | Gestionar Roles, Gestionar Permisos, Agregar, Editar, Buscar y Eliminar Usuarios |
| **Prioridad** | Baja |
| **% Completado** | 70 |

|  |  |
| --- | --- |
| **ID** | CU04 |
| **Nombre** | Configurar tienda |
| **Actores** | Inventarista, Configurador |
| **Sinopsis** | El usuario relaciona los productos que tiene cada tienda |
| **Pre-condición** | Deben existir productos y tiendas configurados previamente |
| **Curso típico de eventos** | 1. El sistema despliega la lista de Tiendas que existen en la base de datos 2. El usuario selecciona una de las tiendas y se despliega un diálogo dónde se muestra el nombre de la tienda y el producto respectivo que se quiere agregar 3. El sistema relaciona los productos a la tienda. |
| **Cursos alternativos** | El usuario no modifica relaciona productos a ninguna tienda y cierra sesión |
| **Extensiones** |  |
| **Prioridad** | Baja |
| **% Completado** | 60 |

|  |  |
| --- | --- |
| **ID** | CU05 |
| **Nombre** | Gestionar productos |
| **Actores** | Inventarista, Configurador |
| **Sinopsis** | El Inventarista/Configurador/Cocinero pueden gestionar las diferentes categorías de los productos así como también pueden relacionar los diferentes ingredientes que tiene un producto. |
| **Pre-condición** | Deben existir categorías e ingredientes previamente configurados |
| **Curso típico de eventos** | 1. El usuario selecciona del menú la opción ‘Gestionar productos’ 2. El sistema despliega un submenú del cual se tienen las opciones ‘Gestionar Categorías’, ‘Gestionar Ingredientes’ y ‘Gestionar productos’. 3. El usuario selecciona alguno de las anteriores opciones y el sistema muestra un listado de elementos según la opción seleccionada. 4. El usuario puede Crear, Buscar, Editar o eliminar Categorías, Ingredientes o Productos. |
| **Cursos alternativos** | El usuario decide no realizar ninguna opción y cierra sesión. |
| **Extensiones** | Gestionar Categorías, Gestionar Ingredientes |
| **Prioridad** | Baja |
| **% Completado** | 60 |

|  |  |
| --- | --- |
| **ID** | CU06 |
| **Nombre** | Gestión de Clientes |
| **Actores** | Cajero |
| **Sinopsis** | El Cajero puede gestionar la información de los clientes habituales. |
| **Pre-condición** | Se debe haber iniciado el juego (CU01) |
| **Curso típico de eventos** | 1. El cajero ingresa a registrar la información de los clientes 2. El sistema le muestra una lista de clientes 3. El usuario puede agregar, editar, buscar o eliminar clientes. |
| **Cursos alternativos** |  |
| **Extensiones** | Agregar cliente, Editar cliente, Buscar cliente, Eliminar Cliente |
| **Prioridad** | Baja |
| **% Completado** | 100 |

|  |  |
| --- | --- |
| **ID** | CU07 |
| **Nombre** | Gestión de Pedidos |
| **Actores** | Cajero |
| **Sinopsis** | El Cajero puede gestionar la información de los pedidos que se realizan en el sitio. |
| **Pre-condición** | Se debe haber iniciado el juego (CU01) |
| **Curso típico de eventos** | 1. El sistema muestra una lista de pedidos al usuario. 2. El cajero puede agregar, editar, modificar o cancelar pedidos |
| **Cursos alternativos** |  |
| **Extensiones** | Agregar , Editar, Buscar, Eliminar pedido |
| **Prioridad** | Alta |
| **% Completado** | 70 |

## Vista lógica (diseño)

En esta vista se muestra la estructura lógica del sistema en términos de abstracciones y entidades que dan forma al sistema. La composición de las clases se forma de los siguientes elementos:

* Interfaz: En la cual se definen los métodos relacionados con un componente
* Implementación: Clase que implementa los métodos definidos en la interfaz de algún componente
* Ejb: Entidad que manipula las entidades para cumplir con las reglas de negocio
* Entidad: Abstracción que representa los objetos reales en el dominio del negocio.

A continuación, se muestran las jerarquías de los componentes definidos hasta ahora, en cada interfaz se definen todas las acciones que se pueden realizar sobre un componente:

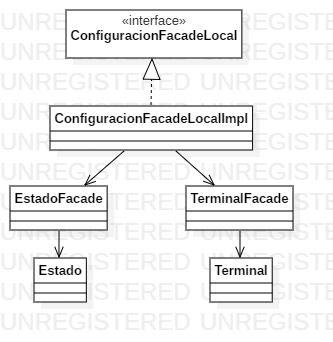
Figura Diagrama de clases del componente Configuración

Figura Diagrama de clases del componente Pedidos



Figura Diagrama de clases del componente Productos



Figura Diagrama de clases del componente Tiendas



Figura Diagrama de clases del componente Usuarios

## Vista de proceso (actividades)

En esta vista se muestra la secuencia de actividades que son prioritarias en el sistema, en la Figura 11 se puede observar la interacción cuando un usuario quiere realizar un pedido.

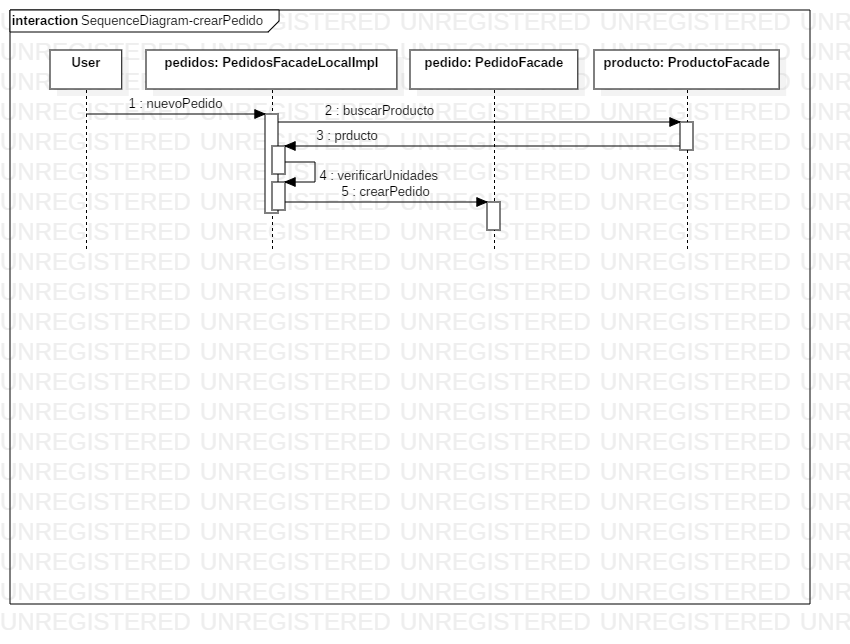


Figura Diagrama de secuencia Realizar pedido

## Vista de desarrollo (componentes)

En esta vista se pueden observar los 5 componentes principales que conforman el sistema, las interfaces por las cuáles se comunican los componentes y las relaciones que se configuran entre sí.

## 

Figura Diagrama de componentes

## Vista física (despliegue)

El sistema Shopping se empaqueta en un archivo .war el cual es instalado en un Servidor Glassfish, este se conecta mediante jdbc con puerto 1521 a la base de datos instalada en un servidor temporal para asistir de repositorio de datos.

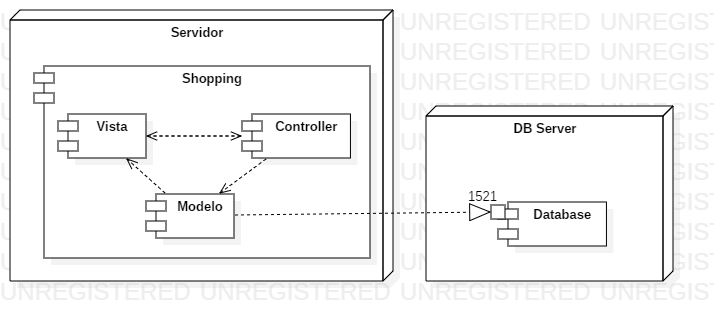


Figura Diagrama de despliegue

# Bibliografía

[1] E. Robson and E. Freeman, *Head First Design Patterns Poster*. 2005.

[2] N. Rozanski and E. Woods, *Software Systems Architecture*. 2005.

[3] M. Fowler, *Patterns of Enterprise Application Architecture*, vol. 48, no. 2. 2002.

[4] OMG, “Object Management Group.” [Online]. Available: https://www.omg.org/.

[5] Object Management Group, “Unified Modeling Language UML.” [Online]. Available: http://www.uml.org/.

[6] P. Kruntchen, “Architectural blueprints–the” 4+ 1” view model of software architecture,” *IEEE Softw.*, vol. 12, no. November, pp. 42–50, 1995.

[7] G. Florijn, “Architectural styles and patterns,” 2015.

[8] Portal ISO, “ISO/IEC 25010.” [Online]. Available: http://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&limitstart=0.