UNIVERSIDAD MAYOR REAL Y PONTIFICIA DE

SAN FRANCISCO XAVIER DE CHUQUISACA

FACULTAD DE CIENCIAS Y TECNOLOGÍA

CARRERA DE INFORMÁTICA



MONOGRAFÍA

IMPLEMENTACIÓN DE SISTEMA DE GESTIÓN DE PEDIDOS PARA RESTAURANTES EN TIEMPO REAL

UNIVERSITARIO: GARY DAVID GUZMÁN MUÑOZ

SUCRE - BOLIVIA

2021

ÍNDICE

[1. CAPÍTULO I – INTRODUCCIÓN 1](#_Toc61450063)

[1.1 ANTECEDENTES 1](#_Toc61450064)

[1.2 JUSTIFICACIÓN 3](#_Toc61450065)

[1.3 PLANTEAMIENTO DEL PROBLEMA 5](#_Toc61450066)

[1.4 FORMULACIÓN DEL PROBLEMA 5](#_Toc61450067)

[1.5 OBJETIVO GENERAL 5](#_Toc61450068)

[1.6 OBJETIVOS ESPECÍFICOS 6](#_Toc61450069)

[2 CAPÍTULO II – DESARROLLO 6](#_Toc61450070)

[2.1 DEFINICIÓN DE LA INGENIERÍA DE SOFTWARE 6](#_Toc61450071)

[2.2 DEFINICIÓN DE INGENIERÍA DE REQUISITOS 6](#_Toc61450072)

[2.3 DEFINICIÓN DE LA INGENIERÍA DE REQUERIMIENTOS 7](#_Toc61450073)

[2.4 ACTIVIDADES DE LA INGENIERÍA DE REQUERIMIENTOS 8](#_Toc61450074)

[2.4.1 EXTRACCIÓN (ANÁLISIS DEL PROBLEMA) 8](#_Toc61450075)

[2.4.2 ANÁLISIS (EVALUACIÓN Y NEGOCIACIÓN DE LOS REQUERIMIENTOS) 9](#_Toc61450076)

[2.4.3 ESPECIFICACIÓN 9](#_Toc61450077)

[2.4.4 VALIDACIÓN 9](#_Toc61450078)

[2.4.5 EVOLUCIÓN DE LOS REQUERIMIENTOS 10](#_Toc61450079)

[2.5 PROCESO DE DESARROLLO DE SOFTWARE 11](#_Toc61450080)

[2.5.1 FASES DEL PROCESO DE DESARROLLO DE SOFTWARE 11](#_Toc61450081)

[2.6 CICLO DE VIDA EN CASCADA 14](#_Toc61450082)

[2.7 LENGUAJE UNIFICADO DE MODELADO (UML) 15](#_Toc61450083)

[2.7.1 ¿QUÉ ES Y PARA QUÉ SIRVE UML? 15](#_Toc61450084)

[2.7.2 TIPOS DE DIAGRAMAS EN UML 15](#_Toc61450085)

[2.8 ARQUITECTURA CLIENTE/SERVIDOR 16](#_Toc61450086)

[2.8.1 COMPONENTES 17](#_Toc61450087)

[2.9 DEFINICIÓN DE SOFTWARE 18](#_Toc61450088)

[2.10 FRAMEWORK 20](#_Toc61450089)

[2.10.1 CUATRO RAZONES PARA UTILIZAR UN FRAMEWORK A LA HORA DE PROGRAMAR 20](#_Toc61450090)

[2.10.1 FACTORES A CONSIDERAR A LA HORA DE ELEGIR UN FRAMAWORK Y OTRO 21](#_Toc61450091)

[2.11 BASE DE DATOS NO RELACIONAL 22](#_Toc61450092)

[2.11.1 TIPOS DE BASES DE DATOS NO RELACIONALES 23](#_Toc61450093)

[2.12 MONGODB 23](#_Toc61450094)

[2.13 BIBLIOGRAFÍA 24](#_Toc61450095)

## CAPÍTULO I – INTRODUCCIÓN

Este proyecto tiene como objetivo desarrollar una aplicación en formato web adaptable para computadora de escritorio, tableta y celular, capaz de dar soporte a la gestión de pedidos de restaurante. El proyecto incluye un entorno de trabajo para un administrador con varios módulos que ayudarán a realizar un buen mantenimiento del sistema. Nuestro alcance comprenderá la creación de un pedido y sus posteriores fases, hasta su extracto de cuenta del pedido.

La situación habitual en un restaurante en cuanto a la apertura de caja, gestión de pedidos, tiempo de espera, impresión de cuenta, entre otros, no es la más ideal en la mayoría de los casos, lo que hace que resulte difícil dar un buen servicio al cliente, sobre todo durante las horas de mayor concurrencia del local.

Los recursos con los que cuenta en un local de este tipo (restaurante, bar, etc.) son escasos, y esto obliga al personal a tener que desplazarse un gran número de veces de un lugar a otro para poder cumplir con su labor, ocasionando deficiencias en el servicio, olvido de órdenes, retardos, y equivocaciones en los pedidos, debido a que el sistema que se utiliza es manual.

Todo lo anteriormente explicado conlleva a pérdidas económicas y de clientela que pueden determinar el éxito o fracaso del negocio. Es por ello que se propone diseñar e implementar un sistema que brinde flexibilidad gracias al uso de terminales táctiles para los meseros, ofrecerá información precisa, garantizada por la aplicación, llevará a cabo el control de caja, la integración de los procesos del negocio por medio de los distintos módulos del sistema, optimizando los mismos, debido a que el sistema reduce el tiempo de ejecución de los procesos y usabilidad.

## ANTECEDENTES

Actualmente, en el mercado se puede encontrar numerosas soluciones, las cuales se dedican a la gestión de restaurantes, cumpliendo con las necesidades más básicas mencionadas previamente. Además, se pretende una mayor participación del cliente en el proceso de la gestión de pedidos, ya que con esa participación se ahorrarán tiempos y malentendidos, además que será de fácil manejo. (Hernanz, 2010)

Después de haber realizado una búsqueda de diferentes soluciones y aplicaciones que cumplen con los requerimientos básicos como ser:

* **EposNow**

Es un sistema de Punto de Ventas (“Point of Sales” o POS por sus siglas en inglés) que se adapta a la mayoría de restaurantes, de cualquier tipo y tamaño. Es fácil de instalar y de usar, e incluye un periodo de prueba de 30 días. Periodo de prueba como este te ayudarán a probar el software, y si no es la mejor opción para ti, lo puedes cancelar fácilmente.

**Características**

* + Te permite manejar tu inventario fácilmente.
  + Mejora el rendimiento de tu personal.
  + Puedes realizar reportes por medio de un panel personalizado.

**Impacto para el restaurante**

Con Epos Now accederás a información a tiempo real de productos y ventas, y así evitarás malgastar dinero en artículos que no necesitas. También podrás estar seguro de tener siempre provisiones de los productos que más vendes. (Eposnow)

* **Avero Slingshot**

Es una plataformabasada en la nube y un software como servicio, o abreviadamente ScuS (de “Software as a Service” o “SaaS” en inglés). Puede convertir cifras y otra información confusa en datos concretos y procesables. Además, ofrece varios módulos de software inteligentes para restaurantes, que te servirán para manejar el costo de alimentos, la productividad, las ganancias, y más. En definitiva, es una herramienta útil que te ayudará a resolver problemas específicos.

**Características**

* + Acceso instantáneo a datos a tiempo real.
  + Asistencia para planificar, comprar y preparar el trabajo.
  + Es una plataforma clara y fácil de usar.

**Impacto para el restaurante**

Avero Slingshot te puede ayudar a controlar los gastos de tu restaurante y a aumentar las ventas. Pero, sobre todo, te ayudará en tu proceso de toma de decisiones, y evitará que tengas que estar haciendo conjeturas. (Avero, 2002)

* **Simple Order**

Es un software para restaurantes que te ayuda a manejar tu inventario y las operaciones de trastienda de tu negocio. No requiere una instalación compleja, ni resulta complicado usarlo.

**Características**

* + Te permite manejar y controlar tu inventario.
  + Te muestra precios de alimentos a tiempo real.
  + Maneja compras, pedidos, crédito y devoluciones de todos los abastecedores.

**Impacto para el restaurante**

SimpleOrder te ayudará a eliminar tareas repetitivas y que consumen mucho tiempo, como aceptar entregas de pedidos, hacer pedidos, procesar papeleo, y más. Además, te ayudará a aumentar los beneficios reduciendo los pedidos excesivos y manteniendo un nivel óptimo de inventario. (SimpleOrder)

Entre otros complacen las necesidades básicas, incluso disponer de un sitio donde almacenan dichos pedidos y el coste de los mismos, pero ninguno de ellos cumple el requisito de la participación con el cliente a la hora de gestionar el pedido o de tener una comunicación en tiempo real con la parte de la cocina para la realización de dichos pedidos.

## JUSTIFICACIÓN

El presente proyecto a través de una investigación, análisis, diseño e implementación busca dar solución al inconveniente planteado, con la creación del proyecto para los restaurantes donde se quisiera implementar un sistema de generación de pedidos y emisión de cuenta de pago.

Es habitual que en este tipo de local (restaurante, bar, etc.) el proceso de atención no es el más adecuado. ¿Cuántas veces nos quejamos porque se tarda demasiado tiempo en ser atendidos? o, por el contrario, ¿aún no hemos decidido qué elegir y el encargado de registrar los pedidos ya está dispuesto a tomar nota?

Las ventajas que obtendrá la empresa con la utilización de un sistema que automatice la gestión de pedidos son múltiples. Las más importantes se pueden resumir en:

* El sistema realizará la gestión de una parte del manejo económico de la empresa: cada pago de un pedido se registrará, permitiendo conocer cuánto dinero ha de haber en la caja, al final del día, del mes o durante un determinado periodo de tiempo.
* Ahorrar en personal, ya que la automatización de procesos, hace que intervengan menos personas, y que se puedan dedicar a otros procesos internos.
* Colaborar con el medio ambiente ahorrando en papel, ya que cada vez que se necesite actualizar la carta con los productos ofertados, o que quieran imprimir el pedido para mandar a cocina, no será requisito si se cuenta con pantallas para la visualización de estos.
* Realizar estadísticas y estudios, ya que se dispondrá de información almacenada en la base de datos, relacionada a los pedidos realizados por los clientes.

Las desventajas que obtendrá la empresa con la utilización de un sistema que automatice la gestión de pedidos son muy pocas. Las más importantes se pueden resumir:

* **Inversión inicial**, ya que, al ser un sistema desarrollado a medida, supone un coste más amplio.
* **Hardware**, dependiendo del que se disponga, hay que hacer una inversión económica mayor o menor.

Automatizar un sistema o procesos conlleva a una serie de ventajas y desventajas, como hemos demostrado, en este caso las ventajas son mayores que las desventajas, por lo que realizar esta automatización tendrá como resultado un beneficio con respecto al sistema sin automatizar.

## PLANTEAMIENTO DEL PROBLEMA

Desarrollar un sistema de gestión de información de pedidos basado en módulos configurables, que permite automatizar parte del proceso generado por un cliente: ordenar su pedido, atenderla, entregar su cuenta, etc. Además, el sistema permitirá manejar la información asociada a las distintas comidas o bebidas ofrecidas.

El proyecto está pensado para restaurantes que no dispongan de un sistema automatizado para la gestión de sus pedidos. Después de haber analizado distintas necesidades de restaurantes que no disponen de ningún sistema automatizado de gestión, a continuación, se exponen las necesidades de mejora más importantes de dichos sistemas:

* Gestión de pedidos.
* Gestión de caja.
* Control de productos ofertados o promocionados.
* Control de cuentas parcial.

## FORMULACIÓN DEL PROBLEMA

La gerencia o dueños del restaurante al no contar con un control adecuado y rápido de la gestión de pedidos. Formula la siguiente interrogante al planteamiento del problema:

¿Cómo lograr un control y optimización de designación para los involucrados en la gestión de pedidos, eliminando el arduo trabajo manual y pérdida de tiempo para una mejor calidad y eficiencia en la atención al cliente?

## OBJETIVO GENERAL

Desarrollar un sistema de control que permita gestionar las fases de pedidos, consiguiendo mejorar el tiempo del proceso de gestión de los mismos.

## OBJETIVOS ESPECÍFICOS

* Identificar la gestión actual de los pedidos realizados por los clientes.
* Agilizar y mejorar el proceso de la información centralizada.
* Identificar las mesas ocupadas por los comensales.
* Obtener lista de pedidos por mesas ocupadas.
* Obtener lista de pedidos para llevar.
* Generar una cuenta al finalizar el servicio.
* Generar un reporte diario en Excel de los pedidos vendidos.

# CAPÍTULO II – DESARROLLO

## DEFINICIÓN DE LA INGENIERÍA DE SOFTWARE

Ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software, y el estudio de estos enfoques, es decir, la aplicación de la ingeniería al software. Es la aplicación de la ingeniería al software, ya que integra matemáticas, ciencias de la computación y prácticas cuyos orígenes se encuentran en la ingeniería (Ingeniería de software es la aplicación de un enfoque sistemático, 2015)

Se pueden citar otras definiciones enunciadas por prestigiosos autores:

* Ingeniería de software es el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas software (Zelkovitz, 1978) (Carhuaricra)
* Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software (Bohem, 1976) (Madrid, 2010)

## DEFINICIÓN DE INGENIERÍA DE REQUISITOS

En la [ingeniería de sistemas](http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_sistemas) y la [ingeniería de software](http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software), la Ingeniería de requisitos o Ingeniería de requerimientos comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requisitos de los inversores, que pueden entrar en conflicto entre ellos.

Muchas veces se habla de requerimientos en vez de requisitos; esto se debe a una mala traducción del inglés. La palabra **requirement** debe ser traducida como requisito, mientras que requerimiento se traduce al inglés como **request**.

La razón principal para escoger este tema se fundamentó en la gran cantidad de proyectos de software que no llegan a cumplir sus [objetivos](http://www.monografias.com/trabajos16/objetivos-educacion/objetivos-educacion.shtml). En nuestro país somos partícipes de este problema a diario, en donde se ha vuelto común la compra de sistemas extranjeros, para luego "personalizarlos" supuestamente a la medida de las [empresas](http://www.monografias.com/trabajos11/empre/empre.shtml). (virtual.itca)

La ingeniería de requerimientos es un proceso que comprende todas las actividades para crear y mantener los requerimientos de un sistema.

Comprende cuatro actividades de alto nivel:

1. Estudio de factibilidad

2. Obtención y análisis de requerimientos

3. Validación de requerimientos

4. Administración de requerimientos (Querétaro, 2005)

## DEFINICIÓN DE LA INGENIERÍA DE REQUERIMIENTOS

La Ingeniería de Requerimientos (IR) cumple un papel primordial en el proceso de producción de software, ya que se enfoca un área fundamental: la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, las necesidades de los usuarios o clientes; de esta manera, se pretende minimizar los problemas relacionados por la mala gestión de los requerimientos en el desarrollo de sistemas.

Normalmente, un tema de la Ingeniería de Software tiene diferentes significados. De las muchas definiciones que existen para requerimiento, a continuación se presenta la definición que aparece en el [glosario](http://www.monografias.com/trabajos13/glosadm/glosadm.shtml) de la IEEE .

(1) Una condición o necesidad de un usuario para resolver un problema o alcanzar un [objetivo](http://www.monografias.com/trabajos16/objetivos-educacion/objetivos-educacion.shtml). (2) Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un [contrato](http://www.monografias.com/trabajos6/cont/cont.shtml), estándar, especificación u otro documento formal. (3) Una representación documentada de una condición o capacidad como en (1) o (2). (johany)

La ingeniería de requerimientos comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer un software nuevo o modificado, por su flexibilidad y orientación al usuario.

## ACTIVIDADES DE LA INGENIERÍA DE REQUERIMIENTOS

La ingeniería de requerimientos se divide en tres actividades:

* En licitar los requerimientos de las diversas fuentes individuales.
* Asegurar que las necesidades de todos los usuarios son consistentes y factibles.
* Validar que los requerimientos que se derivaron son un reflejo exacto de las necesidades del usuario.

Esta clasificación de actividades no puede definirse por medio de una simple progresión a través de la relación entre adquisición, expresión, análisis y especificación. (johany)

Se dice que dentro de la IR existen cuatro actividades básicas que se tienen que llevar a cabo para completar el proceso. Estas actividades ayudan a reconocer la importancia que tiene para el desarrollo de un proyecto de software realizar una especificación y administración adecuada de los requerimientos de los clientes o usuarios. Las cinco actividades son: extracción, análisis, especificación, validación y evolución, y serán explicadas a continuación cada una de ellas.

### EXTRACCIÓN (ANÁLISIS DEL PROBLEMA)

Esta fase representa el comienzo de cada ciclo. Extracción es el nombre comúnmente dado a las actividades involucradas en el descubrimiento de los requerimientos del sistema. Aquí, los analistas de requerimientos deben trabajar junto al cliente para descubrir el problema que el sistema debe resolver, los diferentes servicios que el sistema debe prestar, las restricciones que se pueden presentar y otros. Es importante, que la extracción sea efectiva, ya que la aceptación del sistema dependerá de cuán bien éste satisfaga las necesidades del cliente (johany).

### ANÁLISIS (EVALUACIÓN Y NEGOCIACIÓN DE LOS REQUERIMIENTOS)

Sobre la base de la extracción realizada previamente, comienza esta fase en la cual se enfoca en descubrir problemas con los requerimientos del sistema identificados hasta el momento. Usualmente se hace un análisis luego de haber producido un bosquejo inicial del documento de requerimientos; en esta etapa se leen los requerimientos, se conceptúan, se investigan, se intercambian ideas con el resto del equipo, se resaltan los problemas, se buscan alternativas y soluciones, y luego se van fijando reuniones con el cliente para discutir los requerimientos (johany).

### ESPECIFICACIÓN

En esta fase se documentan los requerimientos acordados con el cliente, en un nivel apropiado de detalle. En la práctica, esta etapa se va realizando conjuntamente con el análisis, se puede decir que la especificación es el "pasar en limpio" el análisis realizado previamente aplicando técnicas y/o estándares de documentación, como la notación UML (Lenguaje de Modelado Unificado), que es un estándar para el modelado orientado a objetos, por lo que los casos de uso y la obtención de requerimientos basada en casos de uso se utiliza cada vez más para la obtención de requerimientos (johany).

### VALIDACIÓN

La validación es la etapa final de la IR. Su objetivo es, ratificar los requerimientos, es decir, verificar todos los requerimientos que aparecen en el documento especificado para asegurarse que representan una descripción, por lo menos, aceptable del sistema que se debe implementar. Esto implica verificar que los requerimientos sean consistentes y que estén completos. Se puede apreciar que el proceso de ingeniería de requerimientos es un conjunto estructurado de actividades, mediante las cuales se obtiene, se valida y se logra dar un mantenimiento adecuado al documento de especificación de requerimientos, que es el documento final, de carácter formal, que se obtiene de este proceso. Es necesario recalcar que no existe un proceso único que sea válido de aplicar en todas las organizaciones. Cada organización debe desarrollar su propio proceso de acuerdo al tipo de producto que se esté desarrollando, a la cultura organizacional, y al nivel de experiencia y habilidad de las personas involucradas en la ingeniería de requerimientos. Hay muchas maneras de organizar el proceso de ingeniería de requerimientos y en otras ocasiones se tiene la oportunidad de recurrir a consultores, ya que ellos tienen una perspectiva más objetiva que las personas involucradas en el proceso (johany).

### EVOLUCIÓN DE LOS REQUERIMIENTOS

Los requerimientos son una manera de comprender mejor el desarrollo de las necesidades de los usuarios y cómo los objetivos de la organización pueden cambiar, por lo tanto, es esencial planear posibles cambios a los requerimientos cuando el sistema sea desarrollado y utilizado. La actividad de evolución es un proceso externo que ocurre a lo largo del ciclo de vida del proyecto. (profesores.elo, 2002)

En el proceso de IR son esenciales diversas actividades. En este documento serán presentadas secuencialmente, sin embargo, en un proceso de ingeniería de requerimientos efectivo, estas actividades son aplicadas de manera continua y en orden variado.

Dependiendo del tamaño del proyecto y del modelo de proceso de software utilizado para el ciclo (johany).

A pesar de las diferentes interpretaciones que cada desarrollador tenga sobre el conjunto de actividades mostradas en la tabla anterior, podemos identificar y extraer cinco actividades principales que son:

* Análisis del Problema
* Evaluación y Negociación
* Especificación
* Validación
* Evolución

## PROCESO DE DESARROLLO DE SOFTWARE

Desarrollo de software. Desarrollar un [software](https://www.ecured.cu/Software) significa construirlo simplemente mediante su descripción. Esta es una muy buena razón para considerar la actividad de desarrollo de software como una [ingeniería](https://www.ecured.cu/Ingenier%C3%ADa_de_Software). En un nivel más general, la relación existente entre un software y su entorno es clara ya que el software es introducido en el mundo de modo de provocar ciertos efectos en el mismo.

Aquellas partes del mundo que afectarán al software y que serán afectadas por él será el [Dominio de Aplicación](https://www.ecured.cu/index.php?title=Dominio_de_Aplicaci%C3%B3n&action=edit&redlink=1). Es allí donde los [usuarios](https://www.ecured.cu/Usuarios) o [clientes](https://www.ecured.cu/Clientes) observarán si el desarrollo del software ha cumplido su propósito.

Una de las mayores deficiencias en la práctica de construcción de software es la poca atención que se presta a la discusión del problema. En general los desarrolladores se centran en la solución dejando el problema inexplorado. El problema a resolver debe ser deducido a partir de su solución.

Esta aproximación orientada a la solución puede funcionar en campos donde todos los problemas son bien conocidos, clasificados e investigados, donde la innovación se ve en la detección de nuevas soluciones a viejos problemas.

Pero el desarrollo de software no es un campo con tales características. La versatilidad de las [computadoras](https://www.ecured.cu/Computadora) y su rápida evolución hace que exista un repertorio de problemas en constante cambio y cuya solución software sea de enorme importancia (BORONAT).

### FASES DEL PROCESO DE DESARROLLO DE SOFTWARE



**Análisis de requisitos:** Extraer los requisitos de un [producto](https://www.ecured.cu/Producto) de software es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere de [habilidad](https://www.ecured.cu/Habilidad) y experiencia en la ingeniería de software para reconocer requisitos incompletos, ambiguos o contradictorios. El resultado del [análisis](https://www.ecured.cu/An%C3%A1lisis) de requisitos con el cliente se plasma en el documento ERS, Especificación de Requerimientos del Sistema, cuya estructura puede venir definida por varios estándares, tales como CMM-I. Asimismo, se define un [diagrama](https://www.ecured.cu/Diagrama) de Entidad/Relación, en el que se plasman las principales entidades que participarán en el desarrollo del software. La captura, análisis y especificación de requisitos (incluso pruebas de ellos), es una parte crucial; de esta etapa depende en gran medida el logro de los objetivos finales. Se han ideado modelos y diversos procesos de trabajo para estos fines. Aunque aún no está formalizada, ya se habla de la Ingeniería de Requisitos. La IEEE Std. 830-1998 normaliza la creación de las Especificaciones de Requisitos Software (Software Requirements Specification).

**Diseño y arquitectura:** Se refiere a determinar cómo funcionará de forma general sin entrar en detalles. Consiste en incorporar consideraciones de la [implementación tecnológica](https://www.ecured.cu/index.php?title=Implementaci%C3%B3n_tecnol%C3%B3gica&action=edit&redlink=1), como el [hardware](https://www.ecured.cu/Hardware), la [red](https://www.ecured.cu/Red), etc. Se definen los casos de uso para cubrir las funciones que realizará el sistema, y se transforman las entidades definidas en el análisis de requisitos en clases de diseño, obteniendo un modelo cercano a la [programación orientada a objetos](https://www.ecured.cu/Programaci%C3%B3n_orientada_a_objetos).

**Programación:** Reducir un diseño a código puede ser la parte más obvia del [trabajo](https://www.ecured.cu/Trabajo) de ingeniería de software, pero no es necesariamente la porción más larga. La complejidad y la duración de esta etapa está íntimamente ligada al o a los [lenguajes de programación](https://www.ecured.cu/Lenguaje_de_Programaci%C3%B3n) utilizados.

**Pruebas:** Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación. Una [técnica](https://www.ecured.cu/T%C3%A9cnica) de prueba es probar por separado cada módulo del software, y luego probarlo de forma integral, para así llegar al objetivo. Se considera una buena práctica el que las pruebas sean efectuadas por alguien distinto al desarrollador que la programó, idealmente un [área](https://www.ecured.cu/%C3%81rea) de pruebas; sin perjuicio de lo anterior el programador debe hacer sus propias pruebas. En general hay dos grandes formas de organizar un área de pruebas, la primera es que esté compuesta por personal inexperto y que desconozca el tema de pruebas, de esta forma se evalúa que la documentación]entregada sea de calidad, que los procesos descritos son tan claros que cualquiera puede entenderlos y el software hace las cosas tal y como están descritas. El segundo enfoque es tener un área de pruebas conformada por programadores con experiencia, personas que saben sin mayores indicaciones en que condiciones puede fallar una aplicación y que pueden poner atención en detalles que personal inexperto no consideraría.

**Documentación:** Todo lo concerniente a la documentación del propio desarrollo del software y de la gestión del [proyecto](https://www.ecured.cu/Proyecto), pasando por modelaciones (UML), [diagramas](https://www.ecured.cu/Diagramas), pruebas, [manuales de usuario](https://www.ecured.cu/Manual_de_usuario), manuales técnicos, etc; todo con el propósito de eventuales correcciones, [usabilidad](https://www.ecured.cu/Usabilidad), mantenimiento futuro y ampliaciones al sistema.

**Mantenimiento:** Mantener y mejorar el software para enfrentar errores descubiertos y nuevos requisitos. Esto puede llevar más tiempo incluso que el desarrollo inicial del software. Alrededor de 2/3 de toda la ingeniería de software tiene que ver con dar mantenimiento. Una pequeña parte de este trabajo consiste en arreglar errores, o [bugs](https://www.ecured.cu/Bugs). La mayor parte consiste en extender el sistema para hacer nuevas cosas. De manera similar, alrededor de 2/3 de toda la [Ingeniería civil](https://www.ecured.cu/Ingenier%C3%ADa_Civil), [Arquitectura](https://www.ecured.cu/Arquitectura) y trabajo de construcción es dar mantenimiento.

Se puede decir que con la mejora continua garantiza la [calidad](https://www.ecured.cu/Calidad) del [producto](https://www.ecured.cu/Producto), ya que el estarla aplicando día con día es la mejor decisión que puede llegar a tener cualquier empresa, porque de esta manera evita grandes problemas en la elaboración o desarrollo de los productos. Esto es fundamental para todas las empresas ya que se vuelven competitivas, con mayor productividad y eficiencia. No hay que olvidar que la mejora se da porque el cliente es el rey y hay que satisfacer todas y cada una de sus necesidades siempre garantizando la [calidad](https://www.ecured.cu/Calidad) (BORONAT).

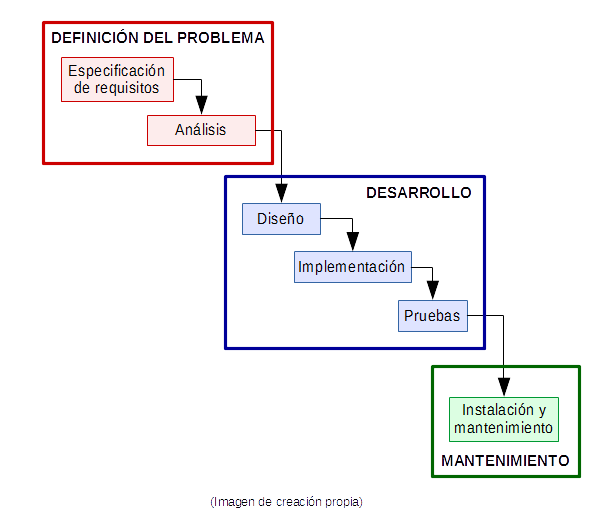
## CICLO DE VIDA EN CASCADA

El modelo en cascada es un enfoque clásico en el desarrollo de software que describe un método de desarrollo lineal y secuencial. Consta de cinco a siete fases, cada fase está definida por diferentes tareas y objetivos, por lo que la totalidad de las fases describe el ciclo de vida del software hasta su entrega. Una vez finalizada una fase, sigue el siguiente paso de desarrollo y los resultados de la fase anterior pasan a la siguiente fase (Ryte Wiki, 2016).

Este modelo asume que todo se lleva a cabo y tiene lugar tal y como se había planeado en la fase anterior, y no es necesario pensar en asuntos pasados que podrían surgir en la siguiente fase. Este modelo no funcionará correctamente si se dejan asuntos de lado en la fase previa. La naturaleza secuencial del modelo no permite volver atrás y deshacer o volver a hacer acciones.

Este modelo es recomendable cuando el desarrollador ya ha diseñado y desarrollado aplicaciones similares con anterioridad, es decir, tiene la experiencia suficiente para terminar con una etapa y comenzar la siguiente.

Son tres las fases en que se agrupan las etapas de este tipo de ciclo de vida:

* Definición del problema, que incluye tanto la especificación de requisitos como el análisis del sistema.
* Desarrollo, que abarca el diseño, implementación y pruebas del sistema.
* Mantenimiento, es decir, la instalación y el mantenimiento del sistema (agrega.juntadeandalucia.es, 2017).

## LENGUAJE UNIFICADO DE MODELADO (UML)

Una imagen vale más que mil palabras. Es por eso que se creó la generación de diagramas con el Lenguaje Unificado de Modelado (UML): para forjar un lenguaje visual común en el complejo mundo del desarrollo de software que también fuera comprensible por los usuarios de negocios y quienquiera que desee entender un sistema. Aprende lo básico de los diagramas UML, además de sus orígenes, usos, conceptos, tipos y pautas sobre cómo dibujarlos usando nuestra (Lucidchart, 2020).

### ¿QUÉ ES Y PARA QUÉ SIRVE UML?

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software, p. ej., en el flujo de procesos en la fabricación.

Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene.

UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML. UML guarda una relación directa con el análisis y el diseño orientados a objetos (Lucidchart, 2020).

### TIPOS DE DIAGRAMAS EN UML

Usando UML se pueden construir numerosos tipos de diagramas. Vamos a citar algunos:

**Diagramas de casos de uso:** representan a los actores y casos de uso (procesos principales) que intervienen en un desarrollo de software.

**Diagramas de clases:** para UML una clase es una entidad, no una clase software. Un diagrama de clases UML puede ser un diagrama del dominio o representación de conceptos que intervienen en un problema, o también un diagrama de clases software. El sentido de un diagrama UML se lo da la persona que lo construye.

**Diagramas de secuencia:** suelen usarse para representar objetos software y el intercambio de mensajes entre ellos, representando la aparición de nuevos objetos de izquierda a derecha.

**Diagramas de colaboración:** suelen usarse para representar objetos o clases y la forma en que se transmiten mensajes y colaboran entre ellos para cumplir un objetivo.

**Diagramas de estados:** suelen usarse para representar cómo evoluciona un sistema (cómo va cambiando de estado) a medida que se producen determinados eventos.

**Otros diagramas:** diagramas de actividad, diagramas de paquetes, diagramas de arquitectura software, etc (aprenderaprogramar.com, 2006).

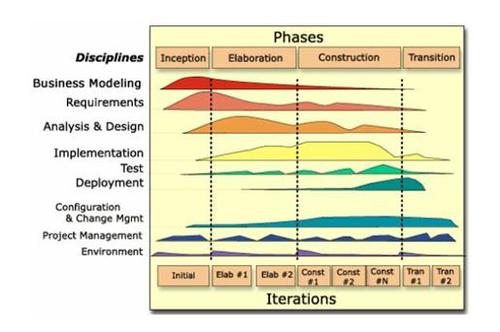
## METODOLOGÍA PROCESO UNIFICADO ÁGIL (AUP)

El proceso unificado ágil (AUP) es una versión simplificada de RUP desarrollada por Scott Ambler. Describe un enfoque simple, fácil de entender, del desarrollo de software de aplicación de negocios usando técnicas y conceptos ágiles. AUP aplica técnicas ágiles incluyendo desarrollo orientado a pruebas, modelado ágil, gestión de cambios ágil y refactorización de bases de datos para mejorar la productividad. La naturaleza en serie de AUP se presenta en cuatro fases:

* **Inicio:** el objetivo es identificar el alcance inicial del proyecto, una arquitectura potencial para el sistema y obtener fondos y aceptación por parte de las personas involucradas en el negocio.
* **Elaboración:** el objetivo es probar la arquitectura del sistema.
* **Construcción:** el objetivo es construir software operativo de forma incremental que cumpla con las necesidades de prioridad más altas de las personas involucradas en el negocio.
* **Transición:** el objetivo es validar y desplegar el sistema en el entorno de producción.

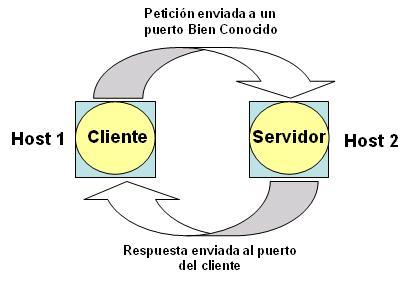
### DISCIPLINAS (AUP)

1. **Modelado:** Entender el negocio de la organización, tratar el dominio del problema e identificar una solución viable para tratar el dominio del problema.
2. **Implementación:** Transformar el modelo en código ejecutable y realizar un nivel básico de pruebas, en particular pruebas unitarias.
3. **Pruebas:** Realizar una evaluación objetiva para asegurar calidad. Esto incluye encontrar defectos, validar que el sistema funciona como fue diseñado y verificar que se cumplen los requisitos.
4. **Despliegue:** Planificar el despliegue del sistema y ejecutar el plan para poner el sistema a disposición de los  usuarios finales.
5. **Gestión de configuración:** Gestión de acceso a los artefactos del proyecto. Esto no sólo incluye el seguimiento de las versiones de los artefactos sino también controlar y gestionar los cambios en ellos.
6. **Gestión de proyecto:** Dirección de las actividades que tienen lugar dentro del proyecto. Esto incluye gestionar  riesgos, dirigir a las personas y coordinar las personas y sistemas fuera del alcance del proyecto para asegurar que se entrega a tiempo y dentro del presupuesto.
7. **Entorno:** El objetivo de esta disciplina es apoyar el resto de los esfuerzos por garantizar que el proceso sea el adecuado, la orientación (normas y directrices), y herramientas (hardware, software, etc.) estén disponibles para el equipo según sea necesario. (Scott W. Ambler)



## ARQUITECTURA CLIENTE/SERVIDOR

**Arquitectura Cliente servidor**. Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el [servidor](https://www.ecured.cu/Tipos_de_servidores_de_red)) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola [computadora](https://www.ecured.cu/Computadora) es más ventajosa en un [sistema operativo](https://www.ecured.cu/Sistema_operativo) multiusuario distribuido a través de una [red de computadoras](https://www.ecured.cu/Red_de_computadoras). La interacción cliente-servidor es el soporte de la mayor parte de la [comunicación](https://www.ecured.cu/Comunicaci%C3%B3n) por [redes](https://www.ecured.cu/Redes_Inform%C3%A1ticas). Ayuda a comprender las bases sobre las que están construidos los algoritmos distribuidos.



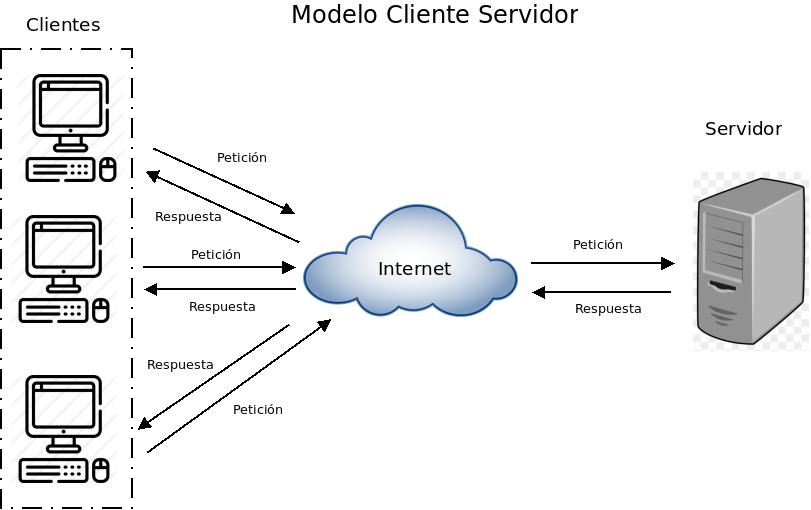
La **arquitectura cliente servidor** tiene dos partes claramente diferenciadas, por un lado la parte del servidor y por otro la parte de cliente o grupo de clientes donde lo habitual es que un [servidor](https://blog.infranetworking.com/que-es-un-servidor/) sea una máquina bastante potente con un hardware y software específico que actúa de depósito de datos y funcione como un sistema gestor de base de datos o aplicaciones.

En esta arquitectura **el cliente suele ser estaciones de trabajo** que solicitan varios servicios al servidor, mientras que **un servidor es una máquina que actúa como depósito de datos** y funciona como un sistema gestor de base de datos, este **se encarga de dar la respuesta demandada por el cliente** (ecured.cu).

### COMPONENTES

Para entender este modelo vamos a nombrar y definir a continuación algunos conceptos básicos que lo conforman.

* **Red:** Una red es un conjunto de clientes, servidores y base de datos unidos de una manera física o no física en el que existen protocolos de transmisión de información establecidos.
* **Cliente:** El concepto de cliente hace referencia a un demandante de servicios, este cliente puede ser un ordenador como también una aplicación de informática, la cual requiere información proveniente de la red para funcionar.
* **Servidor:** Un servidor hace referencia a un proveedor de servicios, este servidor a su vez puede ser un ordenador o una aplicación informática la cual envía información a los demás agentes de la red.
* **Protocolo:** Un protocolo es un conjunto de normas o reglas y pasos establecidos de manera clara y concreta sobre el flujo de información en una red estructurada.
* **Servicios:** Un servicio es un conjunto de información que busca responder las necesidades de un cliente, donde esta información pueden ser mail, música, mensajes simples entre software, videos, etc.
* **Base de datos:** Son bancos de información ordenada, categorizada y clasificada que forman parte de la red, que son sitios de almacenaje para la utilización de los servidores y también directamente de los clientes (Schiaffarino).



Podemos decir que esta arquitectura necesita tres tipos de software para su correcto funcionamiento:

* Software de gestión de datos: Este software se encarga de la manipulación y gestión de los datos almacenados y requeridos por las diferentes aplicaciones. Normalmente este software se aloja en el servidor.
* Software de desarrollo: este tipo de software se aloja en los clientes y solo en aquellos que se dedique al desarrollo de aplicaciones.
* Software de interacción con los usuarios: También reside en los clientes y es la aplicación gráfica de usuario para la manipulación de datos, siempre claro a nivel usuario (consultas principalmente) .

A parte de estos existen más aplicaciones software para el correcto funcionamiento de esta arquitectura pero ya están condicionados por el tipo de sistema operativo instalado, el tipo de red en la que se encuentra, etc (Alvarez, 2007).

## DEFINICIÓN DE SOFTWARE

Software es un término informático que hace referencia a un programa o conjunto de programas de cómputo, así como datos, procedimientos y pautas que permiten realizar distintas tareas en un sistema informático.

Comúnmente se utiliza este término para referirse de una forma muy genérica a los programas de un dispositivo informático, sin embargo, el software abarca todo aquello que es intangible en un sistema computacional.

Software es un término procedente del idioma inglés, que es aceptado por la RAE y que no posee una traducción que se ajuste al español (significados.com).

El Software son los programas de aplicación y los sistemas operativos que permiten que la computadora pueda desempeñar tareas inteligentes, dirigiendo a los componentes físicos o hardware con instrucciones y datos a través de diferentes tipos de programas.

El software, según las funciones que realiza, puede ser clasificado en:

* Software de Sistema Operativo
* Software de Programación
* Software de Aplicación (informaticamilenium.com.mx)

**Software de sistema:** Es el programa responsable de la ejecución de todas las aplicaciones necesarias para que un sistema opere correctamente. Suele confundirse con el sistema operativo pero va más allá, ya que también incluye las herramientas de optimización, los controladores de dispositivo y los servidores, entre otros componentes.

**Software de programación**: Son todas las herramientas que permiten el desarrollo de nuevo software. Por ello, no suelen ser utilizadas por el usuario final sino por los programadores. Incluye, por mencionar los más importantes, los siguientes:

* **Compiladores** (programas que analizan y ejecutan otros programas).
* **Intérpretes** (software que "traduce" los diferentes lenguajes de programación).
* **Editores de texto** (crean y modifican archivos de texto, lenguaje de programación o código fuente).

**Software de aplicación:** Un software de aplicación es un programa diseñado para facilitar algunas tareas específicas para cualquier medio informático, ya sea en computadores, tabletas o celulares.

Los videojuegos, los programas de diseño asistido (como CAD), el software utilizado en las telecomunicaciones, las aplicaciones de productividad empresarial o educativa son algunos ejemplos de este tipo de programas (significados.com).

## FRAMEWORK

Un **framework,**según [wikipedia](https://es.wikipedia.org/wiki/Framework), es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Vamos, una manera de hacernos más fácil la programación. Pero… **¿Por qué debemos elegir un framework a la hora de ponernos a programar?** (Systems, 2015).

### CUATRO RAZONES PARA UTILIZAR UN FRAMEWORK A LA HORA DE PROGRAMAR

**1.- Evitar escribir código repetitivo:** La mayoría de los proyectos tienen partes comunes necesarias para el funcionamiento como, por ejemplo, acceso a base de datos, validación de formularios o seguridad. Un framework nos evita tener que programar estas partes, de esta manera nos resulta más fácil centrarnos en programar la aplicación.

**2.- Utilizar buenas prácticas:** Los frameworks están basados en patrones de desarrollo, normalmente MVC (Modelo-Vista-Controlador) que ayudan a separar los datos y la lógica de negocio de la interfaz con el usuario. Vamos, que gracias a ellos, lo tenemos todo más ordenado.

**3.- Permitir hacer cosas avanzadas que tú no harías:** Está claro que un framework siempre te va permitir hacer cosas de una manera fácil y segura, que  para ti serían imposibles o al menos te costaría mucho tiempo hacerlas.

**4.- Desarrollar más rápido:** Si tenemos en cuenta los puntos anteriores, sabremos que desarrollar una aplicación con un framework nos permite hacerlo más rápido, más limpio y más seguro (Systems, 2015).

### FACTORES A CONSIDERAR A LA HORA DE ELEGIR UN FRAMAWORK Y OTRO

**Soporte de comunidad:** Es muy importante que el framework elegido tenga una comunidad activa y participativa. De esta manera te va a ser más fácil conseguir ayuda en los momentos difíciles. Visita la web de la comunidad y asegúrate que obtendrás consejos y soluciones cuando los necesites.

**Documentación:** La documentación de un framework debe estar actualizada y ser lo bastante amplia para que te ayude con los problemas que te puedan surgir. Una herramienta bien documentada es mucho más fácil de utilizar.

**Simplicidad y potencia:** Algunos framework son muy potentes pero difíciles de entender. El código generado por el framework debe ser claro y limpio, que no tengas que preocuparte de las peculiaridades internas del framework.

**Arquitectura MVC:** Como hemos comentado anteriormente, este tipo de arquitectura nos ayuda a tener separado la lógica de negocios, los datos y la presentación. De esta manera nuestro código esta mucho más ordenado, nos hace más fácil cualquier tipo de modificación y actualización y facilita mucho el trabajo en equipo.

**Reutilización:** Es imprescindible que los elementos desarrollados puedan ser reutilizados en varios proyectos.

**Patrón Active Record:** Es interesante tener una clase especifica que se encargue de realizar las operaciones de modificación y consulta de una determinada tabla de una base de datos. De esta forma nuestra aplicación quedará aislada del trabajo con SQL.

**Posicionamiento:** Dependiendo del proyecto puede ser que no nos interese estar bien posicionado, pero si no es así, es interesante que el framework ofrezca funcionalidades como url’s amigables, etc., para no tener que rehacer el proyecto más adelante.

**Seguridad:** Puede ser un tema que en un principio pasamos por alto y luego ya es demasiado tarde. Asegúrate que el framework este desarrollado tomando la seguridad como punto de partida. Si un framework es muy potente pero no es seguro no nos sirve de nada.

Después de todos estos consejos lo último que te queda es probar, ya que es la única manera de saber si un framework se adapta a las necesidades de un proyecto (Systems, 2015).

## BASE DE DATOS NO RELACIONAL

Las bases de datos NoSQL están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Las bases de datos NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala. Esta página incluye recursos que lo ayudan a comprender mejor las bases de datos NoSQL y comenzar a usarlas (amazon.com).

Es muy común entre desarrolladores de aplicaciones encontrarse en una situación de**tener que elegir si se va a usar una base de datos relacional o no relacional**. La mayoría no se lo piensa demasiado y opta por la opción que más conocen y con la que más cómodos trabajan. Tampoco es una decisión catastrófica; en realidad, ya sea la base de datos relacional o no, se puede construir cualquier cosa.

Vale, entonces, **¿por  qué es importante saber en qué se diferencian y cuál deberíamos usar en cada caso?** Pues porque un buen diseño de base de datos con la tecnología apropiada indudablemente aporta calidad al proyecto. Dependiendo de la naturaleza de la aplicación, interesa que la base de datos tenga unas características u otras (Rendón, 2019).



### TIPOS DE BASES DE DATOS NO RELACIONALES

Es muy probable que a menudo oigas hablar de las bases de datos NoSQL para referirse en realidad a las bases de datos NO relacionales. Se emplea ese término porque se asocia muy estrechamente SQL a las bases de datos relacionales, sin embargo, esto no es correcto, podemos encontrar bases de datos relacionales que utilizan otros lenguajes diferentes a SQL y que por tanto son NoSQL pero relacionales.

Por ello, utilizaré el término NO relacionales en lugar de NoSQL, para referirme a las bases de datos que no siguen la estructura relacional propuesta por Codd. Sencillamente me parece más correcto.

Entre este tipo de bases de datos encontramos:

* Clave-valor
* Orientadas a documentos
* Orientadas a columna
* Orientadas a grafo (jesgargardon.com, 2020)

## MONGODB

Dentro de las bases de datos NoSQL, probablemente una de las más famosas sea MongoDB. Con un concepto muy diferente al de las bases de datos relacionales, se está convirtiendo en una interesante alternativa.

Pero cuando uno se inicia en MongoDB se puede sentir perdido. No tenemos tablas, no tenemos registros y lo que es más importante, no tenemos SQL. Aun así, MongoDB es una seria candidata para almacenar los datos de nuestras aplicaciones.

MongoDB es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección - concepto similar a una tabla de una base de datos relacional -, pueden tener esquemas diferentes.

### ¿Dónde se puede utilizar MongoDB?

Aunque se suele decir que las bases de datos NoSQL tienen un ámbito de aplicación reducido, MongoDB se puede utilizar en muchos de los proyectos que desarrollamos en la actualidad.

Cualquier aplicación que necesite almacenar datos semi-estructurados puede usar MongoDB. Es el caso de las típicas aplicaciones CRUD o de muchos de los desarrollos web actuales.

Eso sí, aunque las colecciones de MongoDB no necesitan definir une esquema, es importante que diseñemos nuestra aplicación para seguir uno. Tendremos que pensar si necesitamos normalizar los datos, de normalizarlos o utilizar una aproximación híbrida. Estas decisiones pueden afectar al rendimiento de nuestra aplicación. En definitiva el esquema lo definen las consultas que vayamos a realizar con más frecuencia.

MongoDB es especialmente útil en entornos que requieran escalabilidad. Con sus opciones de replicación y sharding, que son muy sencillas de configurar, podemos conseguir un sistema que escale horizontalmente sin demasiados problemas (RUBENFA, 2014).

# CAPÍTULO III – RESULTADOS

## METODOLOGÍA (AUP)

Según la ingeniería de software, en el desarrollo de todo producto software existen una serie de actividades que deben realizarse en un orden determinado y que abarcan no solo su producción, sino también su explotación y mantenimiento, esto se denomina de ciclo de vida o proceso de desarrollo del software.

El Proceso Unificado Agil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). (ingenieriadesoftware, 2021)

## DESARROLLO DE LA APLICACIÓN

Se define la metodología AUP acorde a las fases, necesidades y requerimientos del caso.

Se desarrolla la arquitectura y el diseño del prototipo mediante iteraciones la cual nos permite facilitar la elaboración del tema.

### DOCUMENTOS ENTREGABLES

A continuación, se define los documentos entregables que se crearon como resultado del sistema web con la metodología AUP.

*Tabla 1 Documentos Entregables*

|  |  |  |
| --- | --- | --- |
| Fase | Actividad | Documento |
| **Inicio** | Identificación de requerimientos. | Se desarrolla el sistema de acuerdo al registro y seguimiento de información y pedidos.   * Identificación de interesados. * Descripción de posibles soluciones. * Visión generalizada del sistema. |
| **Elaboración** | Arquitectura | * Aplicación. * Requerimientos tecnológicos. * Arquitectura del software. |
| Análisis | * Especificaciones de caso de uso. * Identificación de actores. * Modelo de clases. |
| Diseño | * Modelo de navegación. * Modelo de presentación. * Modelo entidad relación. |
| **Construcción** | Implementación | * Herramientas. * Pantallas del sistema. |
| **Transición** | Prueba | * Prueba del sistema. * Solución a los casos de prueba |

Fuente: Elaboración propia

El proceso de requerimientos marca el inicio de desarrollo del sistema, a definir y determinar cada función, es decir “Quien lo inicia y a quien afecta” se puso en manifiesto lo que se investigó en el ámbito de trabajo dentro del restaurante, a fin de lograr la aceptación de los usuarios a los requerimientos y dar paso al diseño de sistema.

Modelo de Prueba

#### **Diseño**

## Implementación

**Prueba**

#### **Análisis**

Modelo de Casos de Uso

Modelo de Diseño y Desarrollo

Modelo de Análisis

Modelo de Implementación

## Requerimientos

Diagrama de Casos de Uso

Diagramas Colaboración

### Diagrama de Componentes

Diagrama

de Objetos

Diagrama de Secuencia

Diagrama de

Clases

Diagrama de Actividades

Modelo de Diseño

Modelo de Casos de Uso

Modelo de Análisis

Modelo de Desarrollo

### Diagrama de Despliegue

Modelo de

Implementación

Modelo de Prueba

Caso de Estudio y prueba a la funcionalidad del sistema.

### REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales que se pudo identificar para el sistema son los siguientes:

* Registro de Clientes: El software permitirá registrar a los clientes para poder realizar descuentos en el momento del pedido.
* Registrar menú: El software permitirá registrar los platos del menú, así también como las porciones extras tomando en cuenta la categoría en la que pertenece y sus diferentes detalles para poder llenar nuestro menú.
* Registrar precios: El software permitirá registrar los precios de los platos del menú y porciones extras, como también las rebajas por porcentaje de los platos que se desee registrar.
* Apertura y cierre de caja: Registrar múltiples veces en lo largo del día la apertura de caja.
* Registrar pedidos: El software tendrá un diseño por mesas, en la cual mostrará la cantidad de comensales en dichas mesas en tiempo real, a partir de estas mesas se podrá registrar los pedidos en forma ordenada, en automático imprimirá una comanda para llevarlo a cocina.
* Listar pedidos: Una vez concluido el pedido el software podrá visualizar una lista de pedidos para poder atenderlas desde la cocina, con la opción de eliminar el pedido si así se requiera.
* Reporte: El software permitirá descargar un reporte diario introduciendo la fecha, el reporte estará en formato Excel para mayor flexibilidad si la gerencia decide hacer cálculos extras.

### REQUERIMIENTOS NO FUNCIONALES

### DEFINICIÓN DE ACTORES

## PROCESO DE REQUERIMIENTOS

## BIBLIOGRAFÍA

*profesores.elo.* (27 de 08 de 2002). Obtenido de http://profesores.elo.utfsm.cl/~agv/metitulo/LilloGrimblatt/avance2/Plantillas/Requerimientos%20Software.doc

*aprenderaprogramar.com.* (2006). Obtenido de https://www.aprenderaprogramar.com/index.php?option=com\_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46&Itemid=163

*Ingeniería de software es la aplicación de un enfoque sistemático.* (27 de 10 de 2015). Obtenido de https://issuu.com/kelvin91/docs/ingenier\_\_a\_de\_software\_es\_la\_aplic

*Ryte Wiki.* (2016). Obtenido de es.ryte.com: https://es.ryte.com/wiki/Modelo\_en\_Cascada#:~:text=El%20modelo%20en%20cascada%20es,del%20software%20hasta%20su%20entrega.

*agrega.juntadeandalucia.es.* (2017). Obtenido de http://agrega.juntadeandalucia.es/repositorio/20022017/6b/es-an\_2017022012\_9122843/51\_ciclo\_de\_vida\_clsico\_o\_en\_cascada.html

*jesgargardon.com.* (16 de 03 de 2020). Obtenido de https://jesgargardon.com/blog/las-bases-de-datos-no-relacionales/

*Lucidchart.* (2020). Obtenido de https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml

Alvarez, S. (30 de 08 de 2007). *DesarrolloWeb.com.* Obtenido de https://desarrolloweb.com/articulos/arquitectura-cliente-servidor.html

*amazon.com.* (s.f.). Obtenido de https://aws.amazon.com/es/nosql/

Avero. (2002). Obtenido de https://averoinc.com/products-page/

BORONAT, A. (s.f.). *EcuRed.* Obtenido de ecured.cu: https://www.ecured.cu/Desarrollo\_de\_software

Carhuaricra, J. (s.f.). *academia.* Obtenido de https://www.academia.edu/7074168/INGENIER%C3%8DA\_DEL\_SOFTWARE#:~:text=INGENIER%C3%8DA%20DEL%20SOFTWARE%20%E2%80%A2%20Ingenier%C3%ADa,software%20de%20modo%20rentable%20que

*ecured.cu.* (s.f.). Obtenido de https://www.ecured.cu/Arquitectura\_Cliente\_Servidor

Eposnow. (s.f.). Obtenido de https://www.eposnow.com/us/

Hernanz, F. R. (2010). *Sistema de Gestión de Pedidos*. Obtenido de https://ddd.uab.cat/pub/trerecpro/2013/hdl\_2072\_206893/RodriguezHernanzFranciscoR-ETIGa2009-10.pdf

*informaticamilenium.com.mx.* (s.f.). Obtenido de https://www.informaticamilenium.com.mx/es/temas/que-es-software.html

johany. (s.f.). *monografias.com.* Obtenido de https://www.monografias.com/trabajos6/resof/resof.shtml

Madrid, U. C. (12 de 02 de 2010). *Introducción a la Ingeniería del.* Obtenido de http://ocw.uc3m.es/ingenieria-informatica/principios-de-ingenieria-informatica/introduccion-a-la-ingenieria-del-software

Querétaro, S. d. (01 de 07 de 2005). Obtenido de https://mena.com.mx/gonzalo/maestria/ingreq/presenta/procesos\_ir/

Rendón, Y. A. (28 de 05 de 2019). *Academia pragma.* Obtenido de https://www.pragma.com.co/academia/lecciones/bases-de-datos-relacionales-vs.-no-relacionales

RUBENFA. (03 de 20 de 2014). *genbeta.com.* Obtenido de https://www.genbeta.com/desarrollo/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no

Schiaffarino, A. (s.f.). *infranetworking.* Obtenido de https://blog.infranetworking.com/modelo-cliente-servidor/

*significados.com.* (s.f.). Obtenido de https://www.significados.com/software/

SimpleOrder. (s.f.). Obtenido de https://simpleorder.com/

Systems, O. (12 de 07 de 2015). *orix.es.* Obtenido de https://www.orix.es/que-es-un-framework-y-para-que-se-utiliza#:~:text=Un%20framework%2C%20seg%C3%BAn%20wikipedia%2C%20es,hacernos%20m%C3%A1s%20f%C3%A1cil%20la%20programaci%C3%B3n.

*virtual.itca.* (s.f.). Obtenido de https://virtual.itca.edu.sv/Mediadores/stis/11\_\_\_\_ingeniera\_de\_sistemas\_productos\_y\_requisitos.html