Análisis de sistemas II – Parcial I

Proceso unificado de desarrollo (PUD):

Cuando hablamos de PUD nos referimos a una técnica de trabajo fundamentalmente teórico de desarrollo en donde se define el "Quien hace qué y cuando se hace", de esta forma se construye un esquema de procesos en el que podemos gestionar y consolidar los cimientos del sistema, permitiendo documentar el proceso y entender cómo será el proyecto, cuáles serán sus comportamientos, pero sobre todo definir de manera explícita, los objetivos y las actividades a lo largo de su ciclo de vida, es decir, desde sus inicios hasta que finalmente se entrega el producto terminado al cliente. Lo que proporciona un alto nivel de organización y estructura. Este marco teórico está compuesto por ciclos que tiene como características principales:

Ser *iterativo e incremental* esto que quiere decir que se dividirá en ciclos de pequeños proyectos donde eventualmente el sistema continuará expandiéndose y mejorando en cada una de sus versiones, al crecer gradualmente brinda la posibilidad de detectar errores antes de pasar a la siguiente etapa. Una vez aprobado, podrá continuar con el siguiente para seguir con el progreso del proyecto.

Dirigido por casos de uso, cuando hablamos de CU nos referimos a las funcionalidades que tendrá el sistema al momento de interactuar con él usuario, estudiadas en base a las necesidades del cliente. El conjunto de estos casos de uso, conforman un modelo de casos de usos en los que junto con otras herramientas como la base de datos y los distintos diagramas de UML conformaran lo que será el sistema.

Centrado en su arquitectura, Este es un aspecto más técnico que teórico ya que describe como será el despliegue del sistema, interfaces, en que plataformas de utilizar, etc. Nos referimos a la estructura que tendrá el proyecto, pueden existir varios modelos, pero como filosofía base se buscará que este pueda ser escalable en el tiempo al proporcionar una base sólida para su continua actualización a medida que pasen los ciclos iterativos y pueda adaptarse a nuevas demandas y/o modificaciones.

Basados en componentes, una de las ventajas que podemos tener para mejorar la eficiencia al momento de escribir el código es poder reutilizar algunos componentes para las distintas funcionalidades del sistema. Es como crear una planilla en donde podremos reutilizarla como estructura para una función distinta de la que fue creada originalmente. Por ejemplo: Un componente llamado "Persona" que es utilizado para registrar a los empleados, también pueda utilizarse en el momento de registrar a los clientes o incluso a los proveedores.

Que, a su vez, estos ciclos están compuestos por 4 fases: Inicio, elaboración, construcción y transición.

Podría compararse con los planos en un proyecto de arquitectura, en donde el arquitecto al confeccionar el plano de una casa, describe que tan grande será, cuáles serán sus limitaciones, como serán cada una de sus habitaciones, las funciones que tendrá cada una, los distintos servicios que podría necesitar, como también los tiempos de construcción en un orden especifico, cuáles podrían ser los riesgos y costos, cuáles serán los materiales que se utilizaran en la construcción, etc. Esto ayuda a cimentar y organizar la base que sostendrá toda la estructura.

Este modelo teórico esta fuertemente vinculado al "Lenguaje unificado de modelado". En donde el PUD ofrece una metodología de trabajo para la gestión, UML brinda las herramientas graficas para poder visualizar, documentar y comunicar mediante sus diagramas el "Como" se llevará a cabo el proyecto. Es una guía para los equipos de trabajo y desarrolladores al momento de codificar el sistema.

Esta metodología posee 4 fases:

1) *Inicio:*

Es donde se realiza una investigación para obtiene una primera visión integral de lo que será el sistema solicitado, ya que al conocer las necesidades del cliente se podrá definir cuál será su alcance y hasta donde llegara el proyecto, conocer los requisitos principales que podría solicitar, y analizar cuáles son los realmente funcionales que serán de utilidad en el proyecto a largo plazo, cuales podrían ser los riesgos y si en realidad son viables. En esta etapa

inicial es donde se comienzan a modelar los primeros casos de uso, los principales para comenzar con la construcción de un modelo.

2) *Elaboración*:

En esta fase se refina la primera visión obtenida de lo que será el proyecto, ya que se comienza a definir lo que va ser la arquitectura, es decir cómo se construirá el sistema, se perfeccionan los requisitos definidos, analizar y comprender si los casos de uso hasta el momento encontrados son los solicitados por el cliente, también se analiza una solución para aquellos riesgos que podrían presentarse, y se entrega un plan de proyecto con los casos de uso de la arquitectura gracias a la utilización del análisis y el diseño para organizar estos CU de manera que puedan ser implementados.

3) Construcción:

Aquí es donde comienza finalmente la construcción del sistema con la arquitectura que venimos definiendo anteriormente, desde la codificación, hasta los manuales de usuarios, las versiones betas que se van entregando, las pruebas que deben realizarse en cada una de las funcionalidades con el fin de cerciorarse que sean correctas y que realmente funcionen. Que sea una copia fiel de lo que solicito el cliente, teniendo en cuenta el criterio del analista al momento de desarrollar el sistema, es decir más allá de lo que el cliente, proponga como analista, debemos saber que es lo mas conveniente para el sistema dentro de los requisitos en los que se trabaje, esto es con el fin de evitar cualquier error humano que pueda afectar al sistema. Como podría ser el ingresar manualmente el legajo de un empleado, el nombre de alguna provincia donde el sistema sea sensible a la utilización de las mayúsculas, que el mismo usuario ingrese mal escrito alguna palabra o incluso una fecha de nacimiento. Algo tan simple como eso puede generar un error.

4) Transición:

En esta etapa donde se prepara el sistema para su entrega, se llevan a cabo las correcciones correspondientes, las pruebas con los clientes, se capacitan a los usuarios para el uso del sistema con un manual, y se asegura que cumpla con las expectativas impuestas.

Proyecto: Aplicación de Gestión de Pedidos para una Cafetería

«Este sistema permitirá a una cafetería gestionar los pedidos de sus clientes a través de una aplicación móvil. Los usuarios podrán visualizar el menú, hacer pedidos, pagar en línea y elegir si desean retirar en el local o recibir el pedido a domicilio.»

El objetivo de este sistema es el poder brindarle al cliente una manera más rápida y ágil de realizar su pedido, poder pagar y darle la facilidad del envió o retiro del producto comprado. Algo muy importante es el sistema sea simple e intuitivo, de esta forma será mas amigable al momento de utilizarlo. Como también reduce drásticamente el proceso para la cafetería ya que estarían optimizando el tiempo, la atención y los recursos requeridos para llevar a cabo las tareas que se desean implementar en el sistema.

Inicio:

Como primera medida se tomarían las entrevistas pertinentes al cliente, conocer cuáles son sus expectativas con respecto al sistema, que requisitos son los que considera necesarios y fundamentales, comprender que tan robusto debería ser el proyecto a llevar a cabo y las dimensiones de este según las necesidades del negocio. Como también una entrevista o cuestionarios con posibles usuarios, para poder hacer un reconocimiento de que resultaría practico en el momento de utilizar la aplicación e incluso, de tener experiencias con otras aplicaciones de este tipo, cuales son las que consideran innecesarias. Siempre bajo la discreción del analista con el objetivo de reunir la información y los requisitos necesarios para poder llevar a cabo el modelo de negocios, una vez obtenida los primeros resultados de la investigación, se comienza a analizar y escribir el primer bosquejo de lo que será el sistema.

Como usuario de la aplicación: debería ser capaz de gestionar su ingreso al sistema mediante un registro de usuario donde se le otorgara un alias o identificación de usuario para realizar el pedido a su nombre, junto con sus datos básicos. Tendrá acceso al menú disponible ya sea para consultar o al momento de realizar el pedido, visualizar el precio, los detalles del producto, seleccionar el deseado y posteriormente elegir cuál será su medio de pago ya sea transferencia, tarjeta o abonarlo al momento de retirar el pedido, como también poder cancelar el pedido por cualquiera sea la razón antes de confirmar la solicitud. Deberá poder elegir si desea retirarlo en el local o prefiere que sea enviado a su dirección.

Como administrador de la aplicación: debería ser capaz de ingresar su usuario, donde se verificaría el nivel de acceso al sistema y a partir de allí, poder cargar promociones, actualizaciones del menú disponible, de ser necesario agregar o quitar artículos de él, verificar el listado de pedidos pendientes, realizados y cancelados. Tener un recuento de los usuarios que se registraron en la aplicación, junto con la validación de cada uno de ellos. Llevar un control de las ventas realizadas, ser capaz de visualizar si el pago fue procesado correctamente, analizar y comprender las métricas del negocio, cuales fueron los productos más vendidos o los menos solicitados.

Elaboración:

Según los requisitos captados anteriormente se refinarían los casos de usos identificados, donde se detallarían cual seria su enfoque dentro del sistema, que acciones tendría este, como también que tipo de relaciones y se comienza armar lo que será la arquitectura. Cada una de las funcionalidades solicitadas serán analizadas y cuestionadas sobre su viabilidad dentro del sistema. Utilizando las herramientas que nos brinda UML, comenzando con:

Un diagrama de casos de uso, donde se expondrían las principales funciones y como estas se relacionan con el actor, en este caso con el cliente, ya que será una manera eficiente de ilustrar como deberá comportarse el sistema al interactuar con el usuario.

Un diagrama de clases, mostraría los atributos y métodos que los CU tendrían, y como se relacionarían entre ellos con el fin de marcar como seria su flujo de acción. Como en el caso de “Usuario” que como atributo tendría los datos personales del usuario como su nombre, dirección, teléfono y como método realizar un “Pedido” en el cual seria otro CU que estaría vinculado.

Un diagrama de estados seria necesario para comprender cual seria el flujo de acción y cuáles serían los distintos estados en los que podría encontrarse un CU dado un evento que provoque el cambio, como en el caso de un pedido online o el pago.

Construcción:

Comienza a materializarse el proyecto con toda la información reunida a través de la metodología y herramientas UML que proporcionan una guía para los desarrolladores. En cada iteración que se vaya haciendo a medida que se construye el sistema, se irán agregando y mejorando nuevas funciones en el sistema, como una nueva función de un menú especializado (vegano, vegetariano, etc.), incluir un nuevo método de pago, o que el usuario sea capaz de realizar un pedido para un determinado momento del día, si no que también se ira documentando el proceso paso a paso. Sometiendo al sistema a diversas pruebas para verificar la efectividad de cada nueva funcionalidad ya sea de manera individual o conjuntas y si se detecta algún error poder solucionarlo antes de finalmente entregar una nueva versión.

Transición:

Finalmente se realizan las últimas pruebas dentro del entorno de producción validando que su funcionamiento sea el correcto y cumpla realmente con las expectativas del cliente, para entregar la última versión del sistema construido, junto con las documentaciones y los manuales de usuario especificando en ellos los aspectos prácticos en torno al sistema, como poder manipularlo, las funciones que tiene.

Mapeo de Base de Datos.

Una base de datos no es una herramienta de UML, si no que se crea a través de una como son los diagramas de clases que contienen las distintas entidades que serán convertidas en tablas relacionales con el fin de poder transformar esa información en un esquema de base de datos relacional, estas tablas son confeccionadas a partir de filas que son las instancias de una relación y columnas que es donde se almacenan las características de las mismas. Cada una de estas tablas son identificadas a través de las claves primarias, y es gracias a las claves foráneas que se logra evidenciar, no solo si existe una relación entre ellas, sino que también el tipo que las vincula. El objetivo principal de una base de datos es mantener una fuerte organización de la información que guarda lo que permite poder acceder de manera rápida y eficiente a los diversos datos que están relacionados entre sí, ya sea para consulta o extracción de estos, como también mantiene la integridad de los datos que son almacenados.

Como elaborar un mapeo de base de datos:

1) Para realizar un mapeo de base de datos es fundamental poder identificar de manera clara y precisa las entidades en un diagrama de clases, en ellos no solo muestran las clases que componen el diagrama, sino que también los atributos que lo acompañan. Por cada clase identificada esta se ve reflejada en la base de datos como una tabla que contendrá los datos. Al momento de elaborar estas tablas en importante especificar el tipo de dato en cada atributo. (int, varchar, date, etc.).

2) Es sumamente importante destacar que cada tabla tendrá un atributo obligatorio denominado “Clave primaria” que la identifique de manera única y singular de todas las demás, el atributo que se escoja para esta tarea debe ser exclusivo, como lo sería el DNI, un legajo, o incluso generar un ID particular para cada una de estas tablas, ya que no podríamos poner como PK un nombre o apellido cuando podrían existir dos personas que lo compartan. Al momento de establecer una referencia o relación en otra tabla se utilizarán las “Claves foráneas” en las cuales se escriben como atributo en la tabla que está referenciando, cabe aclarar que no necesariamente cada tabla tiene que tener una FK. Esto es fundamental para ayudar a mantener la integridad de los datos, es decir que cada tabla sea única y pueda ser identificada de manera clara e inconfundible.

Por ejemplo: las tablas “Medico” y “Paciente” pueden tener una clave foránea que haga referencia a la clave primaria de la tabla “Persona” estableciendo una relación. Otro ejemplo podría ser una tabla denominada “tipo_menú” que contenga la información de los distintos tipos de menús que podría ofrecer un restaurant (vegetariano, aptos celiacos, veganos, etc.) y otra tabla “Plato” que describa los diferentes platos. Esta ultima no tendrá una FK como atributo.

3) Otro aspecto a tener en cuenta cuando realizamos las tablas es la manera en la que se asocian, el tipo de relación que puedan tener entre ellas. Pueden ser:

“1:1” (uno a uno) → Es cuando una fila de una tabla se relaciona únicamente con una fila de otra tabla.

Ejemplo: tabla “Persona” y otra “Teléfono”. Cada persona va a estar vinculada a un solo número de teléfono.

“1” (uno a muchos) → Es cuando una fila de una tabla se relaciona con mas de una fila en otra tabla.

Ejemplo: tabla “Cliente” y tabla “Pedidos”. Un Cliente puede tener más de un pedido, pero cada pedido va a tener un solo cliente.

“M:M” (muchos a muchos) → Este tipo de relación es un poco más compleja ya que aparece una tabla intermedia que contendrá como claves foráneas las dos claves primarias que identifican a las tablas principales que están siendo relacionadas.

Ejemplo: Si tenemos las tablas “Autores” y” Títulos” tendríamos que crear una tabla intermedia denominada “Titulo autor” en donde se relacionaría un autor con su correspondiente título.

4) La normalización en una base de datos es clave para poder llevar un sistema organizado y eficiente. Nos permite tener una estructura clara y precisa, evita la duplicación de datos y la redundancia al momento de crear las tablas que contendrán dichos datos. Consiste en crear tablas más pequeñas que llevaran datos específicos de las distintas tablas que tuviera la base y relacionar estas mediante las claves. La única desventaja de este método es que nuestro de base de datos tendrá más tablas al dividirlas, es decir que será más mucho más grande pero las ventajas a la hora de implementarlo superan con creces.

Existen tres tipos de normalización más conocidos, pero que en esencia las tres son reglas fundamentales a tener en cuenta:

La primera forma normalizada esta enfocada en eliminar la duplicidad de los datos, ya que nos indica que cada atributo debe ser atómico, esto quiere decir que en una columna no puede haber un nombre Y apellido solo debe integrar un único dato en la misma, por lo que debería haber dos columnas, una para nombre y otra para apellido. También nos dice que en cada fila no puede haber más de un dato, de ser el caso que un mismo nombre tiene dos productos, se debe crear dos filas para ese mismo nombre, eso es con el fin de mantener la claridad y el orden en el registro. Además, cada atributo debe ser homogéneo, es decir que en una columna de nombres no puede haber un apellido o producto.

Esto nos deja como resultado: tres columnas, una para “Nombre”, otra para “Apellido” y una ultima para “Producto”, y en cada una de esas tablas solo pueden obtener un solo valor cuyos valores deben ser únicamente correspondientes con el tipo de dato que se desea guardar.

Nombre	Producto
Maria Costa, Analia Carrera	Cuaderno
Analia Carrera, Mateo Lopez	Hojas, lapiz

Nombre	Apellido	Producto
Maria	Costa	Cuaderno
Analia	Carrera	Hojas
Analia	Carrera	Cuaderno
Mateo	Lopez	Lapiz
Mateo	Lopez	Hojas

La segunda forma normalizada además de eliminar la duplicidad está enfocada en evitar la redundancia de los datos, ya nos dice que además de incluir FN1, también se debe evitar la dependencia parcial que es cuando tenemos una tabla con claves compuestas (Es decir, dos claves primarias) y los atributos de dicha tabla solo dependen de una de estas claves y no de las dos. Por ejemplo: si tenemos ID_Profesor, Nom_Profesor; ID_Curso y Nom_curso.

“Nom_profesor” solo depende de la ID_Profesor, y no de la clave compuesta ID_profesor e ID_Curso, asi como tambien sucede en el caso de “Nom_Curso” que solo depende de ID_Curso y no de ID_Profesor. En este caso se dividen la tabla en dos.

Esto quiere decir que el encargado de dirigir el ID_Curso “15” es ID_Profesor “Jose Rosso”. Mediante una clave foránea de ID_Profesor a la clave primaria de la tabla ID_Curso para consultar cual es el correspondiente.

ID_Profesor	Nom_profesor	ID_Curso	Nom_Curso
1	Jose Rosso	15	Matematicas
2	Mariana Blazz	12	Lengua

ID_Curso	ID_Profesor	Nom_profesor
15	1	Jose Rosso
12	2	Mariana Blazz

ID_Curso	Nom_Curso
15	Matematicas
12	Lengua

La tercera forma normalizada además de eliminar la duplicidad y la redundancia de los datos al eliminar la dependencia parcial, nos permite eliminar también la dependencia transitiva que se crea cuando un atributo depende de otro que no es clave primaria.

ID_Marca	Nombre	Pais
1	Aston Martin	Inglaterra
2	Volkswagen	Alemania
3	Tesla	Estados Unidos
4	Suzuki	Japon
5	Audi	Alemania

ID_Marca	Nombre	ID_Pais
1	Aston Martin	1
2	Volkswagen	2
3	Tesla	3
4	Suzuki	4
5	Audi	2

ID_Pais	Nombre
1	Inglaterra
2	Alemania
3	Estados Unidos
4	Japon

Por ejemplo: el atributo “País” depende del atributo “Nombre” de la tabla ID_Marca. En realidad, el atributo “País” no depende directamente del ID_Marca. Ya que dependiendo del nombre de la marca determina de que país es.

5) Asi como en el diagrama de clases existe la herencia, podemos convertir esas relaciones en un esquema de base de datos:

Tabla única: es cuando tenemos una clase padre que comparte sus atributos con dos subclases hijas, la manera de representar esas dos subclases es agregar en la tabla una columna particular que albergue el tipo de clase que es.

Por ejemplo: si tenemos una clase “Persona” que tiene como atributo el nombre, teléfono y DNI. Naturalmente sus clases hijas “Profesor” y “Estudiante” compartirán esos atributos. Por lo que, en el momento de realizar una tabla relacional, se agregara una columna denominada “Tipo_persona” donde serán indicados como datos profesor y estudiante.

Tabla por clase: En este caso vamos a tener tres tablas distintas, en donde la tabla “Persona” contendrá los atributos que seria compartidos por las subclases hijas y luego las subclases “Profesor” y” Estudiante” que serán relacionadas mediante el ID de cada una para saber cuál será el estudiante y cual el profesor

Tabla por jerarquía: Es similar a la tabla por clase, con la diferencia que tanto la tabla “Profesor” como la de “Estudiante” podrían contener sus propios métodos, además de los heredados por su clase padre “Persona”.

6) Como ultimo ítem para poder realizar un mapeo, es comprender y saber cómo se refleja la agregación y composición de un diagrama de clases en un esquema relacional.

Fundamentalmente ambas representan que “El todo” está conformado por “las partes”, pero tienen una notables diferencia y es su dependencia. Mientras que en la agregación las entidades relacionadas con la entidad padre son complemente independientes y se representa en la base de datos como “1:M”, en la composición las entidades relacionadas con la entidad padre SI son dependientes y en una base de datos se ve como una relación de “1:M” con eliminación en cascada. Esto quiere decir que, si se elimina la clase padre, las sub clases también serán eliminadas.

Ejemplos:

Agregación: Una biblioteca (Clase padre) puede tener muchos libros (Clase hija), pero también los libros pueden existir en otras bibliotecas independientemente o no existir en ninguna.

Composición: Un libro (Clase padre) puede tener muchos capítulos (Clase hija), pero si se elimina ese libro los capítulos que lo componen se eliminarían también.

La base de datos es fundamental para la fase de desarrollo porque como venimos comentando, proporciona una estructura de datos sólida, organizada y confiable de los datos que serán utilizados por el sistema, refleja con claridad los roles dentro del esquema evidenciando quien se relaciona con qué y de qué manera puede hacerlo. Gracias a la normalización de estos se logra evitar la redundancia y la duplicidad lo que optimiza el trabajo al momento de realizar las consultas. Permite también la escalabilidad a futuro ya que podrían agregarse mas datos sin causar grandes dificultades en dentro del sistema ya existente gracias a la diversificación de tablas especializadas que contiene.

Caso de estudio: Tienda online de productos electrónicos.

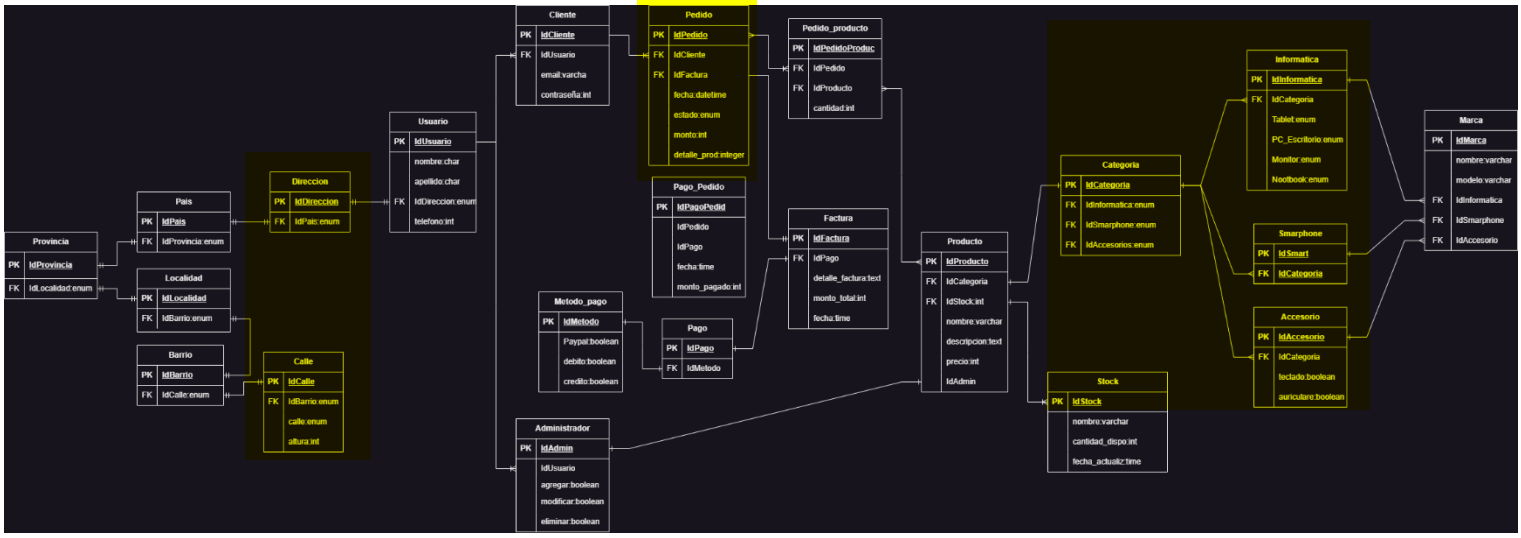
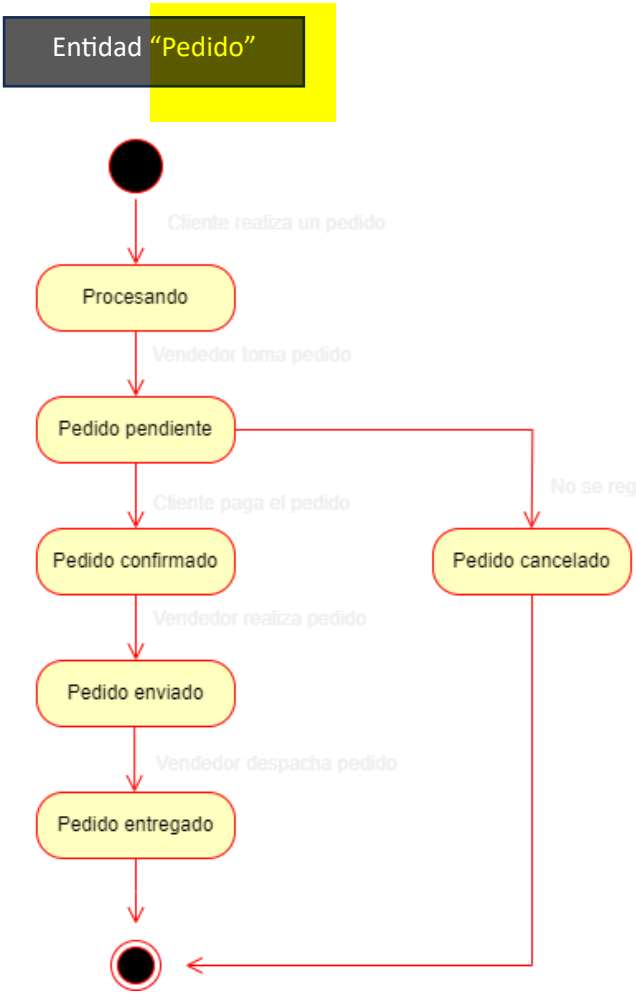


Diagrama de Estado



Dentro del caso de estudio de la tienda online de tecnología, la entidad "Pedido" es una de las más importantes dado que se relaciona con varios componentes dentro del mapeo de base de datos, como puede ser "productos" o "pago". Ya que un cliente puede realizar un pedido, pero si no se encuentra disponible el producto no podría obtenerlo o si al realizar el pago resultara rechazado ya sea por un problema en el sistema de pago o por razones externas al sistema (que el cliente no posea el monto indicado en su medio de pago). Es importante saber cuáles podrían ser los estados en los que podrían encontrarse en respuesta a un evento, para conocer y entender como es su ciclo de vida y de qué manera debería responder.

Fase de Pruebas e Implementación.

Pruebas:

Comenzaría con las pruebas unitarias en los distintos componentes, pero principalmente a las entidades "Pedido", "Pago" y "Stock". Saber si el cliente al seleccionar un producto dentro de la categoría este es realmente seleccionado y agregado al carrito de compras, como también verificar que sea capaz de registrar la cantidad de productos si es que fuera más de uno.

En la entidad "Pago" verificar que efectivamente brinde las distintas opciones y no se produzca un error al realizar un cobro.

En la entidad "Stock" verificar si la cantidad que muestra en el detalle de fecha corresponde actual, y si esta se actualiza a medida que se vayan vendiendo los productos. Otro detalle, es ver si el administrador del sistema pueda modificar tanto los precios como el stock mismo si es que agrega o retira algún producto.

Al finalizar con esas pruebas unitarias, que son entidades en las que el cliente podría interactuar (ya sea al ingresar su número de tarjeta, alias o incluso realizar el mismo pedido) realizaría una prueba integral del sistema. Simulando desde el inicio de la creación de un usuario, comprobar que se autentifique este, hasta la realización del pedido donde debería poder visualizar y seleccionar los productos de las diversas categorías y marcas, hasta finalmente la realización del pago y emisión de la factura.

Implementación:

Una vez realizadas las pruebas, en la fase de implementación comenzaría con la instalación del software y configuración del sistema, otorgando los permisos correspondientes según el nivel de administración del personal, también es importante la capacitación sobre el nuevo sistema, ya que no todos podrían comprender como utilizarlo. Enseñar las nuevas funcionalidades y como estas deberían implementarse, de qué manera podrían generar la factura, o solicitar un resumen de facturación con las ventas mensuales, como actualizar o modificar el stock, llegado al caso modificar la lista de clientes, registrados como agregar nuevos productos.

Brindarles soporte incluso después de ser entregado el sistema por cualquier inquietud o problema que surja, y solicitar un reporte para saber si cumplen realmente con sus expectativas.