Universidad InterNaciones

Analisis de proyectos de software

Daniel Madrigal Luttmann

15010004

Proyecto Final

1. Descripcion del proyecto

Proyecto final

Conste de todas las faces del ciclo de vida del proyecto

consiste en una tienda online de articulos de ferreteria

(no existe un punto de venta especifico)

centrado en la parte online de la tienda

tener:

Diseño de hardware

analisis de modelo de negocio

procesos de la tienda

diseñar la base de datos

arquitectura del web server

Analisis:

DERCAS

Diseño:

DFD

DSI

UML

Desarrollo:

Script de base de datos

Repositorio de documentos generados

Script de pruevas de caja negra y caja blanca

mostrar interfaz de la tienda online

2. FASE DE ANÁLISIS

a) DESCRIPCIÓN DEL PROYECTO A DESARROLLAR

EL proyecto a desarrollar será un sistema de compras en línea de una Ferretería; esta no consta de una tienda física.

b) RESUMEN DEL ANÁLISIS

Los usuarios podrán buscar y ver información de los productos en venda y podrán agregar estos productos a su carrito de compras agregando también si desean múltiples cantidades del producto seleccionado.

Para poder comprar dicho producto los usuarios deberán de iniciar sesión pues su usuario contiene la información de envió y de tarjeta de crédito. No podrán hacerse compras sin un usuario y contraseña.

Antes de realizar la compra los usuarios podrán ver un listado de que y cuando están comprando junto con el precio total e individual de su compra. En esta ventana los usuarios podrán agregar nuevos productos, cancelar la compra o finalizarla.

c) SEGURIDAD EN EL SISTEMA

• Identificación de los usuarios que acceden a los módulos de compra y

descarga de información en la aplicación mediante la introducción de

su nombre y su contraseña.

• Acceso restringido a determinados módulos según perfiles que se

asocian los usuarios definidos en el sistema.

• Los usuarios no se eliminan definitivamente. Los usuarios que ya no

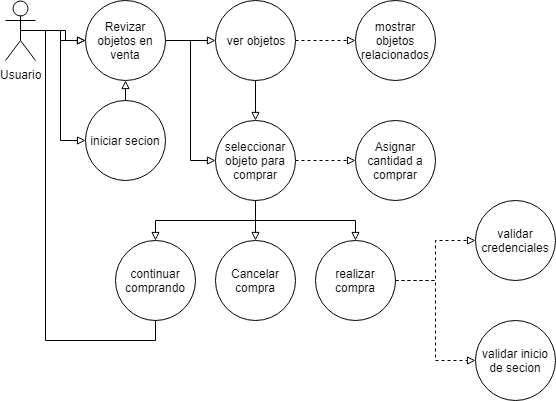
utilizarán el sistema obtienen una baja lógica que evita su acceso sin

eliminar la información de las bases de datos.

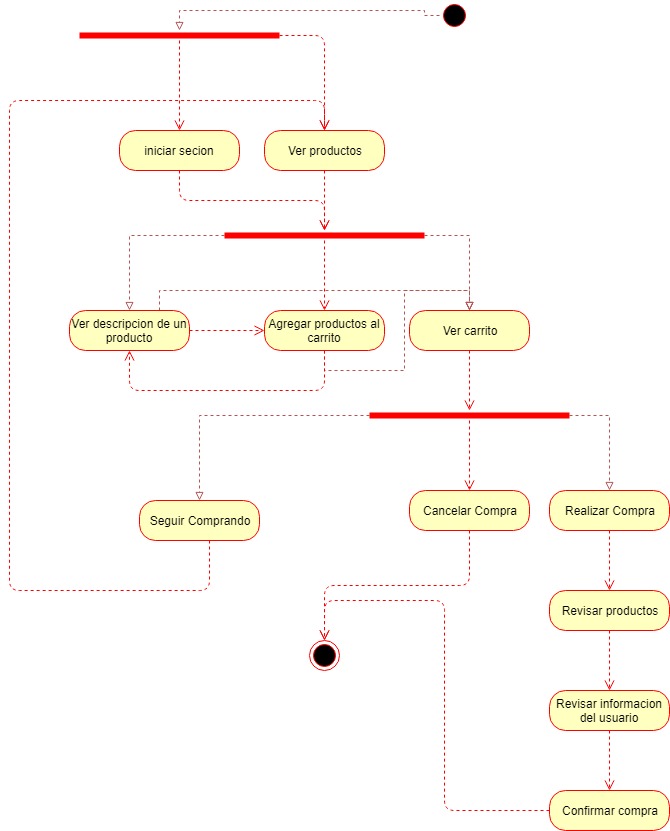
• Solicitud de confirmación antes de completar cualquier acción delicada

para el sistema o de carácter irreversible.

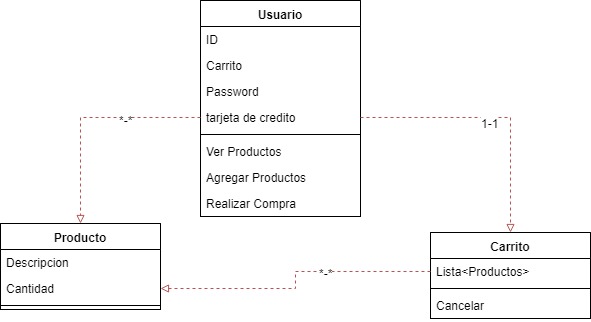
d) DIAGRAMAS Y CASOS DE USO



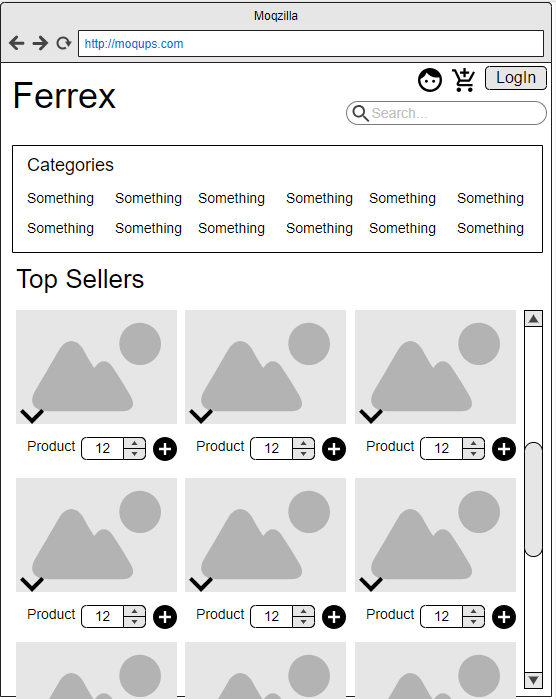
e) DIAGRAMAS DE ESTADO



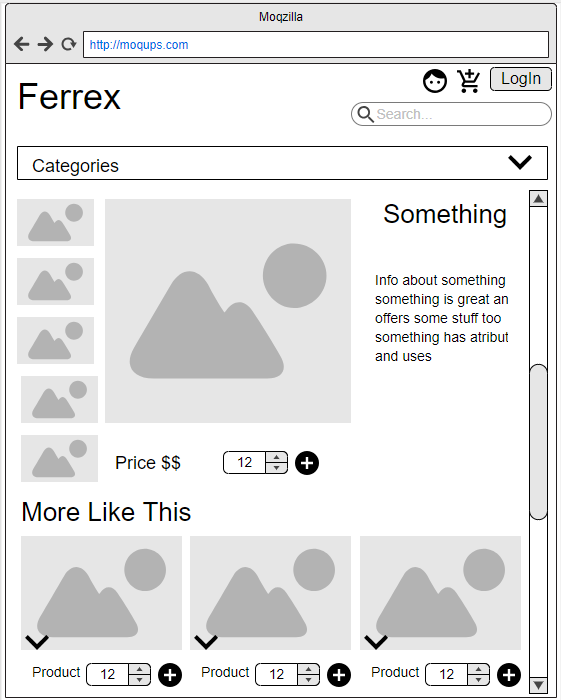
f) DIAGRAMA De Clases



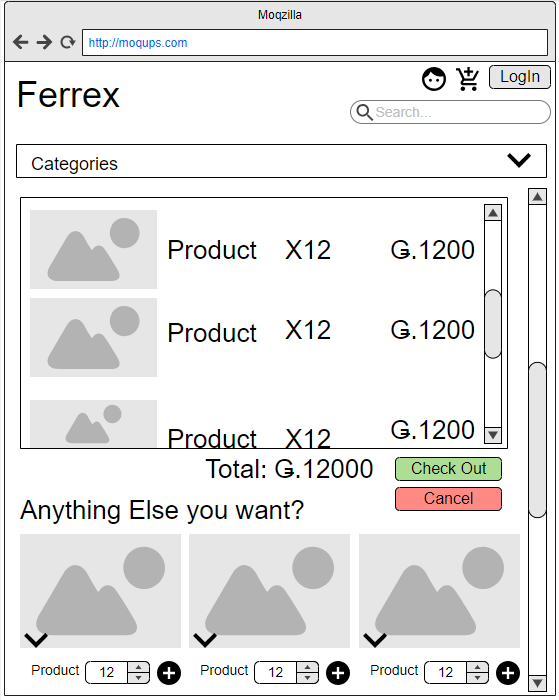
g) DISEÑO DE PANTALLAS



Pantalla de inicio



Información del producto

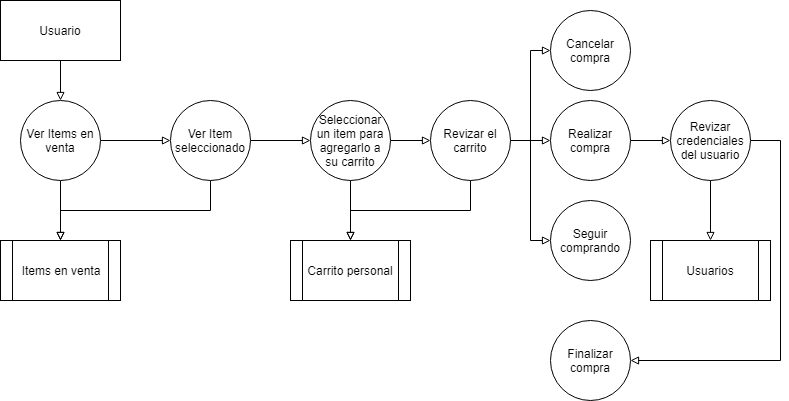


Checkout



Inicio de secion

h)DFD



Para mas información utilizar el documento DERCAS adjunto en el GIT

3. FASE DE DISEÑO

Modelo de venta de los B2C

En las ventas en línea delos B2C, generalmente hay cinco modelos.

1. Vendedores directos: Este es el modelo con el que la gente está más familiarizada- son las tiendas en línea donde los consumidores compran sus productos.

2. Intermediarios en línea: Estos intermediarios ponen a los compradores y a los vendedores juntos sin tener el producto o servicio.

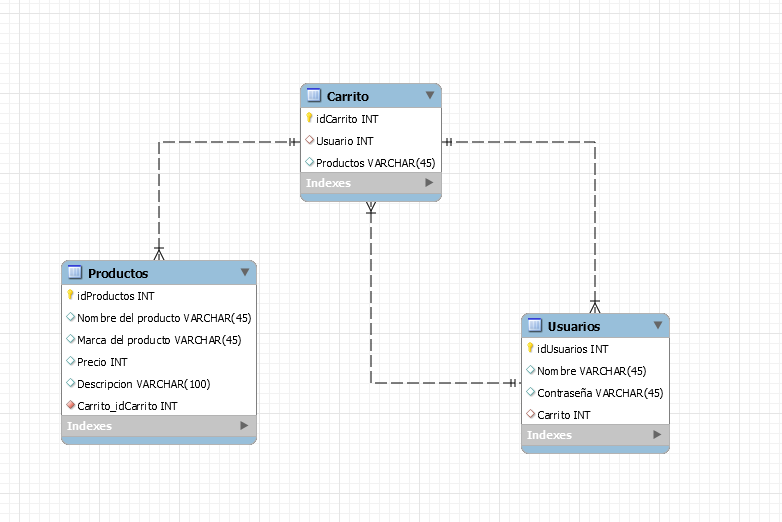
3. Basados en la publicidad: Este enfoque apalanca alto volumen de tráfico web para vender publicidad la cual, sucesivamente, vende producto y servicios al consumidor. Este modelo usa contenido gratis de alta calidad para atraer visitantes al sitio web, que luego encontrarán anuncios en línea.

4. Basados en la comunidad: Este modelo usa las comunidades construidas alrededor de los intereses compartidos para ayudar a los publicistas a publicitar sus productos directamente al sitio de los usuarios. Puede ser un foro en línea de fotografía, gente con diabetes, o miembros de bandas.

5. Basados en una cuota: Estos sitios dirigidos directamente al consumidor cobran una suscripción por el acceso a su contenido. Generalmente incluyen publicaciones que ofrecen un número limitado de contenido gratuito pero cobran por casi todo el contenido.

Los negocios que venden directamente a los consumidores deben tomar en cuenta como su público objetivo le gusta comprar productos similares al de ellos mientras exploran varias opciones de negocio-a-consumidor, ya sea que esas posibilidades envuelvan transacciones personales o en línea.

Utilizar el documento de diseño de sistemas adjunto en el GIT.



4. IMPLEMENTACION

En este caso como no existe código todavía no es posible realizar estas pruebas por lo que las síguete parte es una guía a cómo realizar las pruebas con ejemplos descriptivos

PRUEBA DE CAJA BLANCA

Criterios de cobertura

- Encontrar fragmentos del programa que no son ejecutados por los casos de prueba.

- Crear casos de prueba adicionales que incrementen la cobertura.

- Determinar un valor cuantitativo de la cobertura (que es, de manera indirecta, una medida de la calidad del programa).

1. Cobertura de sentencias:

Comprueba el número de sentencias ejecutables que se han ejecutado. Mantenimiento Avanzado de Sistemas de Información.

En este caso sería revisar paso a paso el código de las funciones de búsqueda

Mostrar información

Agregar al carro de compras

Remover del carro de compras

Completar compra.

2. Cobertura de decisiones:

Comprueba el número de decisiones ejecutadas, considerando que se ha ejecutado una decisión cuando se han recorrido todas sus posible ramas (la que la hace true y la que la hace false, pero también todas las posibles ramas de un switch).

En este caso esta prueba se da por medio de decidir si realizar o no la compra e iniciar o no sesión.

3. Cobertura de condiciones:

Comprueba el número de condiciones ejecutadas, entendiendo que se ha ejecutado una condición cuando se han ejecutado todas sus posibles ramas.

Utilizando el diagrama de casos de uso es posible obtener esta información.

4. Cobertura de condiciones múltiples:

Comprueba el número de condiciones múltiples ejecutadas, considerando que se ha ejecutado una condición múltiple cuando se han ejecutado todas sus correspondientes ramas con todas las posibles variantes de la instrucción condicional.

Esta sección requiere recorrer los diagramas de flujo correspondientes al sistema.

5. Cobertura de condiciones/decisiones:

Comprueba el número de condiciones y decisiones que se han ejecutado.

Manejar una tabla de decisiones para cada función con sus respectivas salidas

6. Cobertura de caminos:

Comprueba el número de condiciones y decisiones que se han ejecutado.

Hacer un árbol de toma de decisiones si es posible

7. Cobertura de funciones:

Comprueba el número de funciones y procedimientos que han sido llamados.

Manejar una tabla de funciones y procedimientos.

8. Cobertura de llamadas:

Comprueba el número de llamadas a funciones y procedimientos que se han ejecutado.

Manejar una tabla de llamada de funciones y procedimientos.

9. Cubrimiento de bucles:

Comprueba el número de bucles que han sido ejecutados cero veces (excepto para bucles do -While), una vez y más de una vez.

Revisar cada ciclo y catalogar cuantas veces es llamado cada uno.

10. Cubrimiento de carrera:

Comprueba el número de tareas o hilos que han ejecutado simultáneamente el mismo bloque de código.

11. Cobertura de operadores relacionales:

Comprueba si se han ejecutado los valores límite en los operadores relacionales (>, <, >=, <=), ya que se asume la hipótesis de que estas situaciones son propensas a errores.

12. Cobertura de tablas:

Comprueba si se ha hecho referencia a todos los elementos de los arrays.

Trabajar con un array de ejemplo para revisar si se utilizan solos los elementos del mismo

PRUEBA DE CAJA NEGRA

1. Particiones de Equivalencia:

La partición de equivalencia es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba.

1. Identificar las clases de equivalencia.

2. Identificar los casos de prueba.

2.Análisis de Valores Límite:

Las condicione límite son aquellas que se hayan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica nos lleva a elegir los casos de prueba que ejerciten los valores límite.

3. Métodos Basados en Grafos.

4. Pruebas de Comparación.

5.Análisis Causa-Efecto.

- si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.

- si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.

- si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.

- si una entrada es booleana, hay 2 clases: si o no. Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

5. Mantenimiento

Plan de Mantenimiento:

Necesidades:

Adaptativo: Modificación de un producto software, después de su entrega, para conseguir que sea utilizable en un nuevo entorno.

Caso:

En caso de tener que reutilizar los componentes de venta.

Correctivo: Modificación reactiva de un producto software, después de su entrega, para corregir defectos detectados. Errores actuales detectados.

Caso:

En caso de bugs que afecten el proceso de compra.

Perfectivo: Modificación de un producto software, después de su entrega, para mejorar su rendimiento o su mantenibilidad.

Caso:

Mantenimiento de las bases de datos y modificación de usuarios.

Preventivo: Modificación de un producto software, después de su entrega, para detectar y corregir defectos latentes antes de que produzcan fallos efectivos.

Caso:

Revisión Anual o semestral de software y feedback para mejorar su uso.

Implementación del proceso:

El proceso para la implementación de un proceso de mantenimiento de software deberá incluir deshabilitar el sistema al momento de implementar la nueva versión. Por el contexto de dicho software estos mantenimientos deberán de hacerse basado en un estudio de tiempos de entrada baja a la página para evitar el mayor impacto a los usuarios. Para los casos de mantenimiento de bases de datos es recomendable mantener un respaldo de las ultimas 3 versiones de las bases de datos en caso de problemas ocurrentes.

Análisis de modificaciones y problemas:

En esta etapa es esencial tener un registro de problemas actuales y previos, en este caso debemos tomar en cuenta que al ser un sistema de compras los problemas causados pueden darse al momento de realizar la aceptación de compras del producto, retirar y agregar productos al carro de compras y buscar y revisar productos específicos, junto con los límites de agregar objetos del mismo tipo al carro. Estos problemas deben de ser tomados en cuenta al momento de realizar cambios para evitar que se repitan o empeoren en el futuro.

Implementación de modificaciones:

Para evitar problemas al momento de implementar modificaciones se debe de tomar en cuenta el momento en el que esta acción es realizada, tanto el día como la hora para evitar choques con acciones, es preferible que dichas implementaciones se realicen tomando en cuenta un análisis previo de uso de la página, revisando visitas diarias y horas.

Siempre utilizar una versión anterior del sistema hasta que este sea aceptado luego de la etapa de revisión.

Revisión y aceptación del mantenimiento:

Para realizar una revisión y aceptación del mantenimiento es recomendando hacer un proceso de compra completo, desde la búsqueda selección y adición de los productos, agregando múltiples copias del producto, y removiendo productos de la lista, revisando la información de productos nuevos y updates de productos anteriores.

Mantener un registro de los cambios en la siguiente tabla:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fecha de revisión | Encargado de revisión | Descripción de los cambios | Pruebas realizadas | Resultados de las pruebas | Anomalías encontradas | Comentarios | Veredicto de revisión | Versión |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Migración:

El proceso de migración del sistema debe realizarse empezando con la base de datos para tener los registros asegurados desde el principio. Siempre asegurándose de mantener un respaldo de dicha información en caso de imprevistos y no utilizar la nueva información hasta que esta sea validada siempre asegurándose de mantener un registro de cambios.