Proyecto Certificador de Desarrollo de SW 1

Tema Nº13:

#### Diseño de Prototipos del Sistema – Pruebas - Planificar y probar el Diseño

Indicador de logro Nº13:Diseña los prototipos de un sistema de información y elabora un plan de pruebas de software orientada a aplicaciones de escritorio.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº13:**

DISEÑO DE PROTOTIPOS DEL SISTEMA – PRUEBAS

Objeto De La Experiencia

* Conoce la versión inicial de la solución esbozando un prototipo en un breve periodo de tiempo y mejorándolo en varias iteraciones, probando y evaluando la eficacia del diseño del prototipo culminando con instalación final del sistema.
* Elabora el Diseño de Prototipos de formularios de un proyecto informático

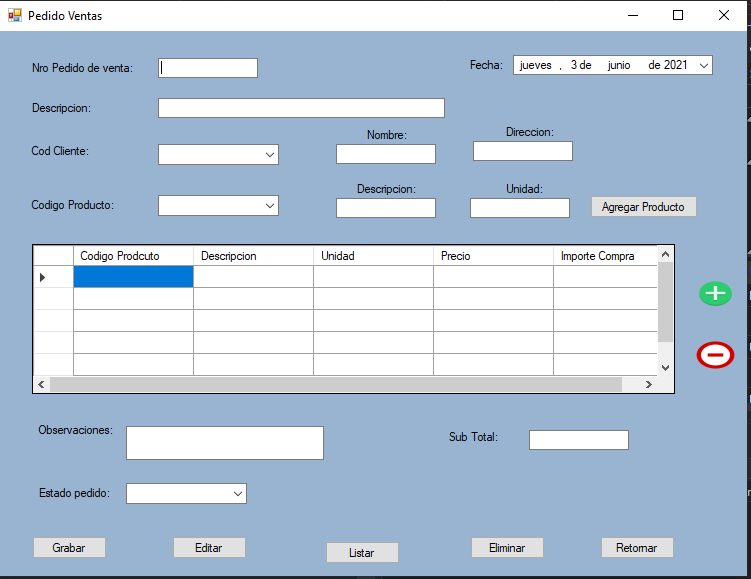
Marco Teórico

1. Diseño dePrototipo de Sistema

Concepto

Los prototipos de Software son implementaciones realizadas con técnicas de programación del sistema interactivo propuesto que reproducen el funcionamiento de una parte importante de las funcionalidades con el objetivo de probar determinados aspectos del sistema final.

Representación Gráfica

****

El paradigma de construcción de prototipos

* Los prototipos de Software son implementaciones realizadas con técnicas de programación del sistema
* Escuchar al cliente. Recolección de requisitos. Se encuentran y definen los objetivos globales, se identifican los requisitos conocidos y las áreas donde es obligatorio más definición.
* Construir y revisar la maqueta (prototipo).
* El cliente prueba la maqueta (prototipo) y lo utiliza para refinar los requisitos del software

Este modelo es útil cuando

* El cliente no identifica los requisitos detallados.
* El responsable del desarrollo no está seguro de la eficiencia de un algoritmo, sistema operativo o de la interface hombre-máquina.
  1. Ventajas y desventajas del Modelo dePrototipo

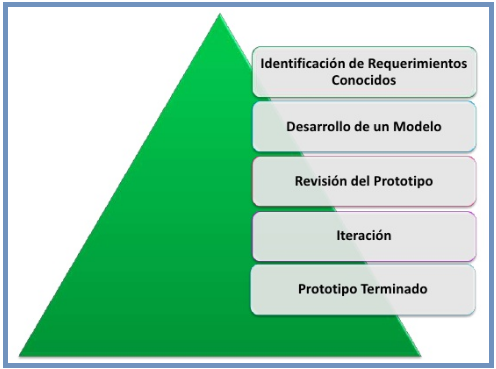
Ventajas

* Permite la construcción del sistema con requisitos poco claros o cambiantes
* El cliente recibe una versión del sistema en muy poco tiempo, por lo que lo puede evaluar, probar, y empezar a utilizarlo
* Se pueden introducir cambios en las funcionalidades del sistema en cualquier momento
* Involucra al usuario en la evaluación de la interfaz de usuario
* Se reduce el riesgo y la incertidumbre sobre el desarrollo
* Genera signos visibles de progreso, que se utilizan cuando existe una demanda en la velocidad del desarrollo
* Permite entender bien el problema antes de la implementación final

Desventajas del modelo de prototipo

* El cliente puede quedar convencido con las primeras versiones y, quizás, no vea la necesidad de completar el sistema o rediseñarlo con la calidad necesaria
* Requiere trabajo del cliente para evaluar los distintos prototipos y traducirlo en nuevos requisitos
* Requiere un tiempo adicional para definir adecuadamente el sistema
* No se sabe exactamente cuánto será el tiempo de desarrollo ni cuantos prototipos se tienen que desarrollar
  1. **Etapas de construcción del modelo de prototipo**

Normalmente, el prototipo sirve como mecanismo para identificar los requisitos del software, y su construcción suele llevar las siguientes etapas:

****

* 1. **Identificación de requerimientos conocidos**

La determinación de los requerimientos de una aplicación es tan importante para el método de desarrollo de prototipos como lo es para el ciclo de desarrollo de sistemas o análisis. Por consiguiente, antes de crear un prototipo, los analistas y usuario deben de trabajar juntos para identificar los requerimientos conocidos que tienen que satisfacer.



* 1. Desarrollo de un Modelo

Los desarrolladores explican a los usuarios:

* El método
* Las actividades a realizar
* La secuencia en que se llevará a cabo
* El proceso de construcción del prototipo se debe iniciar con el desarrollo de un plan general que permita conocer el proceso de desarrollo.
  1. Revisión del Prototipo

Se realiza la evaluación y con la información obtenida se levantan las características que debe llevar la siguiente versión del prototipo:

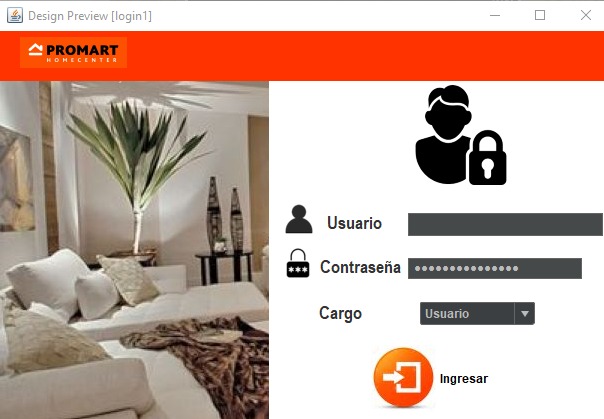
* La evaluación permite profundizar los rasgos de los usuarios y los de la organización que tienen influencia sobre la aplicación y en su implementación.
* Los cambios en el prototipo son planificados con los usuarios antes de llevarlos a cabo por el analista.
  1. Iteración

Se produce un proceso de refinamiento en el que el prototipo es refinado para que satisfaga las necesidades del cliente, al tiempo que facilita al desarrollador de software un mejor conocimiento del sistema.

* 1. Prototipo Terminado

Los pasos anteriores se repiten varias veces hasta cuando los usuarios y desarrolladores están de acuerdo en que el sistema ha evolucionado lo suficiente, e incluye todas las características necesarias.

Cuando el prototipo está terminado, el paso que sigue a continuación es tomar la decisión sobre cómo proceder con su instalación



* 1. El prototipado ayudará a:
* Testear, validar nuevas ideas rápidamente
* Entender las claves de la interacción del usuario
* Mejorar la experiencia de uso e iterar sobre ello
* Demostrar la capacidad de un producto
* Validar el detalle del diseño de la apariencia: Definir el propósito de tamaños, colores, sombras, etc.

1. Requerimientos funcionales y no funcionales
   1. Requerimientos funcionales

Los requerimientos funcionales hacen referencia a la descripción de las actividades y servicios que un sistema debe proveer. Normalmente este tipo de requerimientos están vinculados con las entradas, las salidas de los procesos y los datos a almacenar en el sistema.

* 1. Requerimiento no funcional

Los requerimientos no funcionales engloban características como rendimiento, facilidad de uso, presupuestos, tiempo de entrega, documentación, seguridad y auditorías internas.

Clasificación de requerimientos no funcionales

Requerimientos no funcionales de producto

* Requerimientos de usabilidad: La usabilidad se define como el esfuerzo que necesita hacer un usuario para aprender, usar, ingresar datos e interpretar los resultados obtenidos de un software de aplicación.
* Requerimientos de eficiencia: Relacionado con desempeño en cuanto a tiempo de respuesta, número de operaciones por segundo, entre otras mediciones, así como consumo de recursos de memoria, procesador, espacio en disco o red.
* Requerimientos de dependibilidad: Engloba varios atributos

Disponibilidad: Disposición del sistema para prestar servicio correctamente.

Confiabilidad: Continuidad del servicio prestado por el sistema.

Seguridad industrial: Ausencia de consecuencias catastróficas para el usuario o el ambiente.

Integridad: Ausencia de alteraciones inadecuadas al sistema.

Mantenibilidad: Posibilidad de realizar modificaciones o reparaciones a un proceso sin afectar la continuidad del servicio.

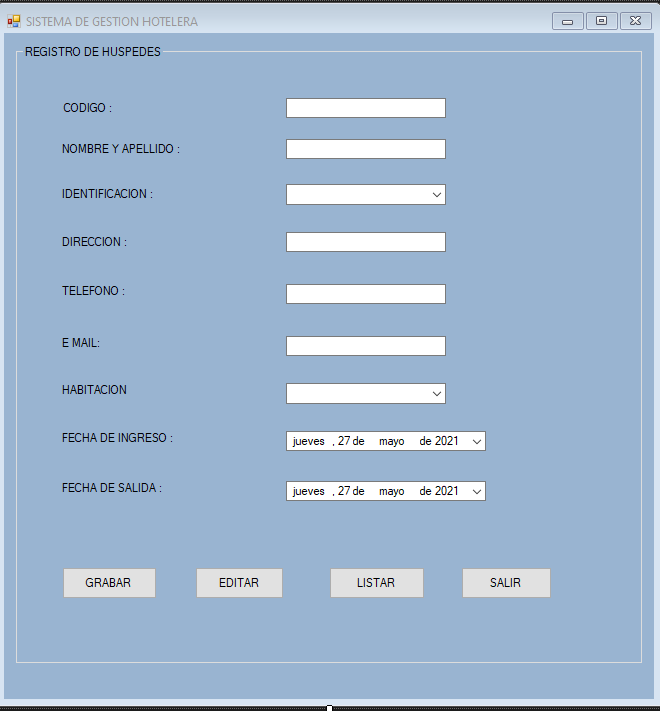
* Requerimientos de seguridad: Capacidades funcionales o no funcionales que debe tener un sistema para cumplir atributos en el área de seguridad de tecnología de información, seguridad de datos, seguridad lógica, control de acceso a información (restricciones de acceso), autenticidad de la información, privacidad, entre otros aspectos.

1. Caso Propuesto Nro. 4

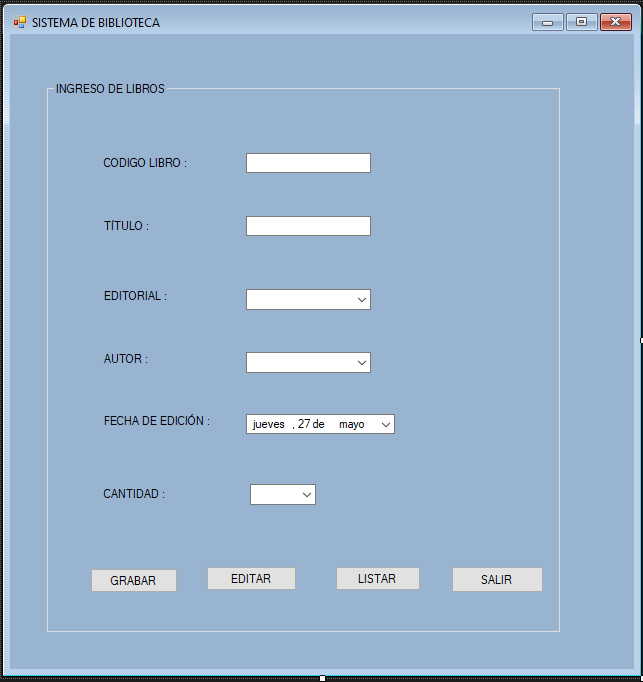
3.1 Diseñar la Interfaz de usuario para la captura de los datos del registro de un Huésped en un hotel.

3.2 Diseñar la interfaz de usuario para la captura de los datos de un libro que debe ser registrado en la Biblioteca.

Prototipo de la Solución 1



Prototipo de la Solución 2



PRUEBAS DE SOFTWARE

Objeto De La Experiencia

Conoce la disciplina que proporciona orientación sobre cómo evaluar y valorar la calidad del producto de software.

Marco Teórico

Disciplina - Prueba

Tiene como finalidad

La prueba de disciplina actúa como proveedor de servicios de otras disciplinas en muchos aspectos. Las pruebas se centran principalmente en la evaluación o la valoración de la Calidad del producto, hecho que se lleva a cabo mediante las prácticas:

* Buscar y documentar los defectos en la calidad del software.
* Opinar sobre la calidad percibida del software.
* Validar y demostrar las suposiciones efectuadas en las especificaciones de diseño y requisitos con una demostración concreta.
* Validar que el producto de software funciona según lo diseñado.
* Validar que los requisitos se han implementado de forma adecuada.

Existe una interesante diferencia entre Probar y las otras disciplinas de RUP - básicamente, las tareas de Probar son encontrar y exponer los puntos flacos del producto de software. Es interesante porque, para obtener mayores beneficios, necesita una filosofía general diferente de la que se utiliza en las disciplinas de Requisitos, Análisis & diseño e Implementación. Una sutil diferencia es que estas tres disciplinas se centran en la completitud, mientras que la Prueba se centra en la falta de ésta.

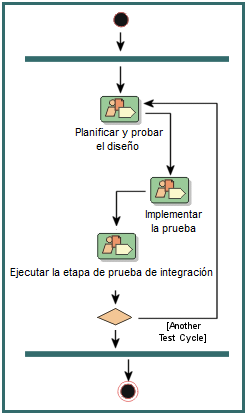
Un buen esfuerzo de prueba se orienta con preguntas como las siguientes:

* ¿Cómo puede fallar este software?
* ¿En qué posibles situaciones podría fallar este software de forma previsible?

La prueba desafía las presuposiciones, los riesgos y las dudas inherentes al trabajo de otras disciplinas y dirige estas preocupaciones hacia la utilización de una demostración concreta y una evaluación imparcial. Lo que desea es evitar dos posibles extremos:

* Un enfoque que no desafíe adecuada ni eficazmente el software, y que exponga sus problemas o debilidades inherentes
* Un enfoque que resulte inapropiadamente negativo o destructivo; adoptando tal enfoque negativo, puede resultar imposible considerar el producto de software de calidad aceptable y se podría distanciar el esfuerzo de prueba de otras disciplinas

**1.1 Flujo de Trabajo**



1. **Planificar y Probar el Diseño**
   1. **Planear la Prueba**

|  |
| --- |
| **Propósito**   * Recopilar y organizar información sobre planificación de pruebas. * Para crear el plan de prueba. |
| **Pasos**   * Identificar los requisitos para la prueba * Evaluar el riesgo * Desarrollar estrategia de prueba * Identificar recursos * Crear horario * Generar plan de prueba |
| **Artefactos de entrada**   * Especificaciones complementarias (Definido anteriormente) * Modelo de diseño (Definido anteriormente) * Modelo de implementación (Definido anteriormente) * Modelo de caso de uso (Definido anteriormente) * Modelo de caso de uso (Definido anteriormente) |

|  |
| --- |
| **Artefactos resultantes**  Plan de prueba. - La definición de las metas y objetivos de las pruebas dentro del alcance de la iteración (o proyecto), los elementos a los que se dirige, el enfoque a tomar, los recursos requeridos y los entregables que se producirán. |
| **Ro**l  Probador. - Es responsable de las actividades centrales del esfuerzo de prueba, que implica realizar las pruebas necesarias y registrar los resultados de esas pruebas. Esto cubre:   * Identificar el enfoque de implementación más apropiado para una prueba determinada * Implementando pruebas individuales * Configurar y ejecutar las pruebas * Registro de resultados y verificación de la ejecución de la prueba * Analizar y recuperarse de errores de ejecución |

* 1. **Diseñar la Prueba**

|  |
| --- |
| **Propósito**   * Identificar un conjunto de casos de prueba verificables para cada compilación. * Identificar los procedimientos de prueba que muestran cómo se realizarán los casos de prueba. |
| **Pasos**   * Identificar y describir casos de prueba * Identificar y estructurar los procedimientos de prueba * Revisar y evaluar la cobertura de la prueba |
| **Artefactos de entrada**   * Componente (Definido anteriormente) * Modelo de implementación (Definido anteriormente) * Especificaciones complementarias (Definido anteriormente) * Plan de prueba (Definido anteriormente) * Caso de uso (Definido anteriormente) |
| **Artefactos resultantes**  Casos de prueba. - La definición (generalmente formal) de un conjunto específico de entradas de prueba, condiciones de ejecución y resultados esperados, identificados con el propósito de realizar una evaluación de algún aspecto particular de un elemento de prueba objetivo. |
| **Ro**l  Probador (Definido anteriormente) |

1. **Implementar la prueba.**
   1. **Diseñan las Clases de Pruebas**

|  |
| --- |
| **Propósito**  Para diseñar funcionalidad específica de prueba |
| **Pasos**   * Identificar clases específicas de pruebas * Interfaz de diseño para herramienta de prueba automatizada * Comportamiento del procedimiento de prueba de diseño |
| **Artefactos de entrada**:   * Modelo de diseño (Definido anteriormente) * Caso de prueba (Definido anteriormente) |
| **Artefactos resultantes:**  Clase de prueba, Un estereotipo de clase en el modelo de diseño |
| **Rol:**  Diseñador. - El rol de diseñador define las responsabilidades, operaciones, atributos y relaciones de uno o varios componentes, y determina cómo se ajustarán al entorno de implementación. |

* 1. **Implementar los componentes de prueba**

|  |
| --- |
| **Propósito**  Para implementar la funcionalidad específica de la prueba |
| **Pasos**   * Controladores / talones de pruebas unitarias y de implementos * Interfaz de prueba de implementación y unidad para herramientas de prueba automatizadas * Implementar y poner a prueba el comportamiento del procedimiento de prueba |
| **Artefactos de entrada:**   * Construir (Definido anteriormente) * Componente (Definido anteriormente) * Clase de prueba (Definido anteriormente) * Caso de prueba (Definido anteriormente) |
| **Artefactos resultantes:**  Componente de prueba.- Un estereotipo de componente en el modelo de implementación. |
| **Rol:**  **Implementador (Definido anteriormente)** |

1. **Ejecutar la etapa de prueba de integración.**
   1. **Ejecutar la prueba**

|  |
| --- |
| **Propósito**  Para ejecutar pruebas y capturar resultados de pruebas. |
| **Pasos**   * Ejecutar procedimientos de prueba * Evaluar la ejecución de la prueba * Verificar los resultados de la prueba * Recuperarse de pruebas detenidas |
| **Artefactos de entrada:**  Construir (Definido anteriormente) |
| **Artefactos resultantes**:   * Resultados de la prueba. - Una recopilación de información resumida determinada a partir del análisis de uno o más registros de prueba y solicitudes de cambio, que proporciona una evaluación relativamente detallada de la calidad de los elementos de prueba objetivo y el estado del esfuerzo de prueba. A veces se lo denomina un repositorio más grande de muchos resultados de pruebas. * Defecto. - La depuración implica ejecutar el compilador en el software para detectar defectos de implementación en el nivel del código fuente. Estos defectos suelen estar relacionados con el diseño detallado o con la redacción real del código fuente. El artefacto de defectos enumera las anomalías encontradas a través de las pruebas. El artefacto del defecto contiene una descripción e información precisas sobre las anomalías y cómo se pueden volver a crear para que los probadores puedan asegurarse de que los defectos estén bien definidos. |
| **Rol:**  Probador (Definido anteriormente) |

* 1. **Evaluar la prueba**

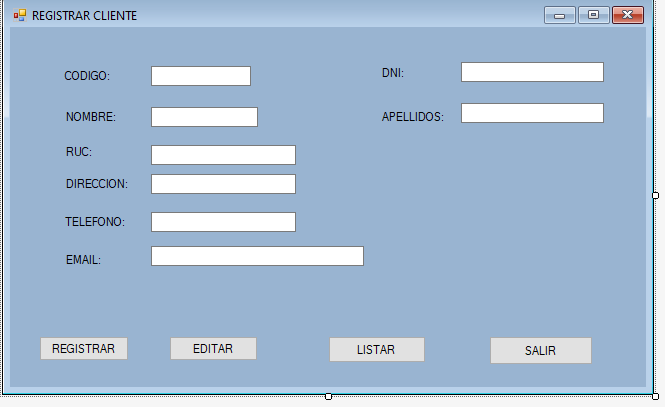
|  |
| --- |
| **Propósito**   * Evaluar los resultados de las pruebas y registrar las solicitudes de cambio. * Calcular y entregar las medidas clave de prueba. * Generar el resumen de la evaluación de la prueba. |
| **Pasos**   * Analice los resultados de las pruebas y envíe solicitudes de cambio * Evaluar la cobertura de pruebas basada en requisitos * Evaluar la cobertura de prueba basada en código * Analizar defectos * Determinar si se han cumplido los criterios de finalización de la prueba y de éxito * Genere el resumen de evaluación de la prueba |
| **Artefactos de entrada:**  Resultados de la prueba (Definido anteriormente) |
| **Artefactos resultantes**:  Informe de evaluación de la prueba. - El Informe de evaluación de la prueba organiza y presenta un análisis resumido de los Resultados de la prueba y las medidas clave de la prueba para su revisión y evaluación, generalmente por parte de las partes interesadas clave en la calidad. Además, el Informe de evaluación de la prueba puede contener una declaración general de calidad relativa y proporcionar recomendaciones para futuros esfuerzos de prueba. |
| **Rol:**  Probador (Definido anteriormente) |

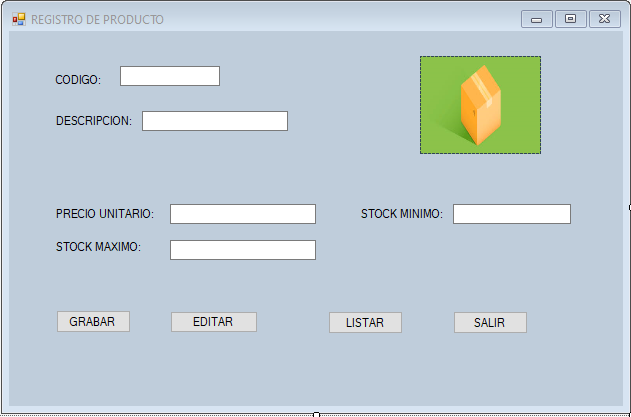
Caso Práctico Nro. 5

La solución está construida en Ms Visual Studio. Net, C#, aplicación de escritorio para un Sistema Integrado de Ventas, Compras y Almacén: Se adjunta Formularios Frm Login, Registrar Cliente, Registro de Productos

* Se utilizó el Entity Framework
* Motor de Base de Datos, Ms SQL SERVER 2017/2019
* Erwin Modeler, para diseñar la base de datos, Modelo Lógico y Modelo Físico.
* Para elaborar el Análisis y Diseño Orientado a Objetos, el IBM Rational Software Architect Designer V9.6
* Se programó utilizando el Patrón MVC (Model – View – Controll)

****



****

Lista De Requerimientos Funcionales

|  |  |
| --- | --- |
| **NUM REQ FUNC** | **DESCRIPCION** |
| RF01 | Validación de Usuario |
| RF02 | Validación de Contraseña |
| RF03 | Validación de Numero de intentos |
| RF04 | Validación de Frm Mant - Cliente.cs |
| RF05 | Validación de Frm Mant - Productos.cs |
| RF06 | Validación del Frm Stock Máximo Permitido |
| RF07 | Validación de la existencia de Clientes |

Lista De Pruebas

|  |  |
| --- | --- |
| **NUM CASO USO** | **DESCRIPCION** |
| PRB01 | Prueba de que el usuario exista y también de los componentes del Frm |
|  |  |
| PRB02 | Prueba de que el password sea de 8 caracteres, que inicie con una letra Mayúscula y el resto números enteros, y además que encripte la contraseña con asteriscos |
| PRB03 | Prueba de que tenga como máximo 3 intentos para ingresar, después el sistema se cierra |
| PRB04 | Valide que todos los campos son obligatorios y que valide cada campo de acuerdo al tipo de datos (char, varchar( ), integer, numeric, decimal, texto, gráfico, etc.), además de cada uno de los botones del formulario cliente.cs |
| PRB05 | Valide que todos los campos son obligatorios y que valide cada campo de acuerdo al tipo de datos (char, varchar( ), integer, numeric, decimal, texto, gráfico, etc.), además de cada uno de los botones del formulario producto.cs |
| PRB06 | Validar de que el cliente no pueda solicitar más productos del Stock Máximo |
| PRB07 | Validar que el cliente exista al momento de realizar un pedido de venta y además validad cada uno de los campos del Frm pedido de venta.cs |

Trazabilidad De Pruebas

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | RF01 | RF02 | RF03 | RF04 | RF05 | RF06 | RF07 |
| PRB01 | **X** |  |  |  |  |  |  |
| PRB02 |  | **X** |  |  |  |  |  |
| PRB03 |  |  | **X** |  |  |  |  |
| PRB04 |  |  |  | **X** |  |  |  |
| PRB05 |  |  |  |  | **X** |  |  |
| PRB06 |  |  |  |  |  | **X** |  |
| PRB07 |  |  |  |  |  |  | **X** |

**Actividad:**

Ingresa a la plataforma virtual, luego desarrolla la siguiente actividad propuesta:

1. **CUESTIONARIO TÉCNICO**
2. Tomando como ejemplo el Caso Nro. 04, de la presente guía Diseño de Prototipos, elabore los diferentes prototipos de pantallas Logín y Menú Principal de su proyecto en grupo a desarrollar. Deberá utilizar Ms Visual Studio .Net, C#, solución de escritorio
3. Elaborar el Diseño de Prototipos de los formularios transaccionales de su proyecto en grupo a desarrollar. Deberá utilizar Ms Visual Studio .Net, C#, solución de escritorio.
4. Tomando como ejemplo el Caso Nro. 05, Pruebas de Software, elabore una lista de requerimientos funcionales, lista de pruebas
5. Elabore la matriz de Trazabilidad de Pruebas
6. Tomando como ejemplo el Caso Nro. 05, Pruebas de Software, realice los diferentes Test de Pruebas de Calidad de Software de la Solución de Escritorio de su proyecto en grupo a desarrollar. Deberá utilizar Ms Visual Studio .Net, C#, solución de escritorio.
7. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_