Especificación de requisitos de software

Proyecto: App para restaurante
Revisión 2.0

Abril 2024

Ficha del documento

Fecha	Revisión	Autor	Verificado dep. calidad.
22/02/2024	1.0	Juan Omar Trivellari Ramírez Luis Antonio Peniche López Andrik Yahir Rosado Trejo	
03/04/2024	2.0	Juan Omar Trivellari Ramírez Luis Antonio Peniche López Andrik Yahir Rosado Trejo	

Documento validado por las partes en fecha:

Por el cliente	Por la empresa suministradora

Contenido

Introducción

Propósito

El propósito de este documento es definir de manera clara y detallada los requisitos funcionales y no funcionales del software de bienes raíces, con el fin de guiar el desarrollo y asegurar que el sistema satisfaga las necesidades y expectativas tanto de la empresa como de sus clientes.

Alcance

La propuesta es desarrollar un proyecto de software utilizando PHP, NodeJS y MySQL. El proyecto se centrará en una página web que permita a los clientes de la empresa seleccionar entre varios anuncios de casas para adquirirlas.
Por otro lado, los administradores del sistema podrán agregar, modificar y eliminar los anuncios de propiedades.

Por lo que, para finalizar, se puede decir que este proyecto busca proporcionar una solución integral y eficiente para la promoción y ventas de las propiedades, respaldado por una interfaz de usuario amigable al usuario.

Personal involucrado

Nombre	Juan Omar Trivellari Ramírez
Rol	Líder de proyecto y Programador
Categoría profesional	Ingeniero de Software
Responsabilidades	Gestión de tiempos del proyecto y codificación

Nombre	Luis Antonio Peniche López
Rol	Programador
Categoría profesional	Ingeniero de Software
Responsabilidades	Asegurar la calidad y codificación

Nombre	Andrik Yahir Rosado Trejo
Rol	Diseñador y Programador
Categoría profesional	Ingeniero de Software
Responsabilidades	Diseño y codificación

Resumen

Este documento detalla exhaustivamente el producto de software y se organiza en las siguientes secciones:

a. Descripción: En esta sección se proporcionará una visión completa del producto de software, incluyendo su propósito, contexto de uso y una visión de sus capacidades principales.

b. Requisitos Específicos: Los requisitos específicos se presentarán en detalle en esta sección. Esto incluirá una lista exhaustiva de los requisitos funcionales y no funcionales del software, con descripciones detalladas y ejemplos cuando sea necesario.

c. Apéndices: Los apéndices contendrán información adicional que complemente la información del proyecto que se está desarrollando.

Este documento servirá como una referencia integral para todos los stakeholders involucrados en el proyecto de desarrollo de software, proporcionando una comprensión detallada de los requisitos y las expectativas del producto.

Descripción general

Perspectiva del producto

La página web surge bajo la necesidad de gestionar de manera eficiente y ordenada los anuncios de propiedades. Se busca que el cliente pueda acceder a la página para consultar anuncios de propiedades y adquirirlas.

Funcionalidad del producto

El sistema contara con las siguientes funcionalidades:

Seleccionar propiedades:

- Los clientes podrán visualizar los anuncios que la empresa tiene disponible, para seleccionarlos y añadirlos a su carrito.

Realizar compra:

- Cuando el cliente termine de seleccionar sus propiedades podrá realizar su compra.

CRUD de propiedades:

- Los administradores del sistema podrán añadir, eliminar y modificar las propiedades que tienen disponibles en el momento. Al añadir, modificar o eliminar un producto se manipularán los siguientes datos:
  - Vendedor
  - Titulo
  - Precio
  - Imagen
  - Descripción
  - Número de habitaciones
  - Número de Baños
  - Cajones de estacionamiento

Ver lista de propiedades:

- Los administradores podrán ver la lista de propiedades disponibles.

Características de los usuarios

Tipo de usuario	Cliente
Formación	Ninguna
Habilidades	Leer instrucciones y uso básico de una computadora
Actividades	Seleccionar propiedades y adquirirlas.

Tipo de usuario	Administrador
Formación	Capacitación mínima
Habilidades	Uso básico de una computadora
Actividades	CRUD de propiedades, consultar y cancelar anuncios

Restricciones

- El sistema solo estar disponible en español.
- El sistema funcionara en el cualquier navegador.

Suposiciones y dependencias

- El personal de la empresa debe estar capacitado para utilizar el sistema.

Requisitos comunes de los interfaces

Interfaces de usuario

RI 1: La interfaz de usuario debe ser diseñada de manera que los clientes y administradores puedan acceder y utilizar todas las funcionalidades del sistema de forma sencilla y sin confusión. Debe ser intuitiva, lo que significa que las acciones y opciones disponibles deben ser claramente comprensibles y fáciles de encontrar. La navegación dentro del sistema debe ser fluida, permitiendo a los usuarios moverse entre diferentes secciones y realizar sus tareas de manera eficiente. Se deben evitar elementos de diseño complicados o distracciones innecesarias que puedan dificultar la experiencia del usuario.

Interfaces de software

RI 2: El sistema debe de poseer una conexión con el sistema manejador de base de datos MySQL.

Requisitos funcionales

Requisito funcional 1

RF 1: El sistema debe permitir crear un CRUD para las propiedades, incluyendo datos como:

- Vendedor
- Titulo
- Precio
- Imagen
- Descripción
- Número de habitaciones
- Número de Baños
- Cajones de estacionamiento

Requisito funcional 2

RF 2: El sistema debe almacenar los datos de las compras de los clientes, los cuales son el número de pedido, nombre del cliente, correo electrónico del cliente, detalles del pedido (Vendedor, Titulo, Precio, Imagen, Descripción, Número de habitaciones, Número de Baños, Cajones de estacionamiento) y precio total.

Requisito funcional 3

RF 3: El sistema debe implementar un método de encriptación para proteger los datos (Nombre del cliente, correo electrónico y contraseña) de los clientes.

Requisito funcional 4

RF 4: El sistema debe generar un reporte en formato PDF con un listado de las compras incluyendo datos como (Vendedor, Titulo, Precio, Imagen, Descripción, Número de habitaciones, Número de Baños, Cajones de estacionamiento) y precio total.

Requisito funcional 5

RF 5: El sistema permitirá a los clientes realizar compras de las propiedades disponibles.

Requisito funcional 6

RF 6: El sistema debe generar un listado con las compras pendientes.

Requisito funcional 7

RF 7: El sistema debe considerar reglas de negocio.

Requisitos nofuncionales

Requisitos de rendimiento

RNF 1: El sistema debe de tener un ágil acceso y respuesta para su actualización instantánea reflejada.

Seguridad

RNF 2: El sistema deberá solicitar al cliente una cuenta para poder realizar compras.

Fiabilidad

RNF 3: El sistema debe tener un tiempo medio entre fallos de al menos 100 horas de funcionamiento continuo.

Disponibilidad

RNF 4: El sistema debe de funcionar el 99% del tiempo.

Mantenibilidad

RNF 5: Se debe proporcionar un manual de usuario detallado y documentación técnica para facilitar el mantenimiento del sistema.

Apéndices

Plan de trabajo (22 febrero – 23 mayo)

Entregable	Inicio	Fin	Responsable (s)
Definición formal de requisitos y arquitectura	22 de febrero	29 de febrero	Todos
Diagramas UML (Clases)	1 de marzo	4 de marzo	Luis Peniche
Diagrama entidad-relación	1 de marzo	10 de marzo	Andrik Rosado
Bocetos visuales del UI	1 de marzo	3 de marzo	Juan Trivellari
Desarrollo del UI	4 de marzo	10 de marzo	Juan Trivellari Luis Peniche
Desarrollo de la base de datos	11 de marzo	21 de marzo	Luis Peniche Andrik Rosado
Codificación del CRUD de Propiedades (RF1)	22 de marzo	31 de marzo	Andrik Rosado Juan Trivellari
Codificación del registro de compra (RF2)	1 de abril	10 de abril	Juan Trivellari Andrik Rosado
Implementación de un sistema de encriptación (RF3)	11 de abril	21 de abril	Todos
Implementación del reporte de compras (RF4)	22 de abril	27 de abril	Luis Peniche Andrik Rosado
Seleccion de propiedades (RF5)	28 de abril	10 de mayo	Andrik Rosado
Generar listado compras pendientes (RF6)	3 de mayo	10 de mayo	Juan Trivellari
Implementación de pruebas unitarias	11 de mayo	14 mayo	Todos
Pruebas de integración	15 de mayo	18 de mayo	Luis Peniche Juan Trivellari
Documentación	18 de mayo	23 de mayo	Todos

Estándar de programación

- **Convención de nombres**:
  - Los nombres de clases, interfaces y enumeraciones deben comenzar con una letra mayúscula y seguir la convención pascal case (MiClase, MiInterfaz, MiEnum).
  - Los nombres de métodos y variables deben comenzar con una letra minúscula y seguir la convención camel case (miMetodo, miVariable).

- **Indentación y Espaciado**:
  - Utilizar sangrías consistentes (por ejemplo, 4 espacios por nivel) para mejorar la legibilidad.
  - Usar espacios alrededor de operadores y después de comas en declaraciones y definiciones.

- **Comentarios y Documentación**:
  - Documentar el código con comentarios claros y concisos. Utilizar comentarios de estilo Javadoc para documentar clases, métodos, campos y parámetros.
  - Proporcionar una descripción significativa y documentación sobre el propósito y el funcionamiento de las clases y métodos.

- **Clases y Métodos**:
  - Dividir las clases y métodos en unidades pequeñas y cohesivas con responsabilidades claras.
  - Utilizar modificadores de acceso (public, private, protected) de manera adecuada para controlar la visibilidad y el acceso.
  - Implementar correctamente los métodos equals(), hashCode(), y toString() cuando sea necesario.

- **Gestión de Excepciones**:
  - Utilizar excepciones apropiadas para manejar errores y situaciones excepcionales.
  - No atrapar excepciones genéricas como Exception si no es necesario; capturar solo las excepciones que se puedan manejar de manera efectiva.

- **Gestión de Recursos**:
  - Utilizar bloques try-with-resources para garantizar la liberación adecuada de recursos como archivos o conexiones.
  - Cerrar recursos, como flujos o conexiones, en un bloque finally cuando sea necesario.

- **Constantes**:
  - Nombrar las constantes con letras mayúsculas y guiones bajos (MI_CONSTANTE).

- **Herencia y Polimorfismo**:
  - Aplicar el principio de sustitución de Liskov y evitar la duplicación de código al máximo posible.
  - Usar interfaces y clases abstractas cuando sea apropiado para definir contratos y comportamientos comunes.

- **Manejo de Colecciones**:
  - Utilizar las clases y métodos de la biblioteca estándar de Java (ArrayList, HashMap, etcétera) para el manejo de colecciones en lugar de implementar estructuras de datos personalizadas.

- **Versionamiento de Código**:
  - Utilizar un sistema de control de versiones (Git) para el seguimiento de cambios en el código fuente y para colaborar con otros desarrolladores de manera efectiva.

Prácticas de construcción

- **Métricas aplicadas**
  - Span.
  - Tiempo de vida.

- **Uso de estándares de codificación**

- **Nombramiento de variables**
  - Se nombrará de acuerdo al problema (Problema-Orientación).
  - Longitud óptima del nombre.
  - Se utilizarán calificadores de cálculos.
  - Se utilizarán opuestos comunes.
  - Las variables de estado tendrán un nombre específico.
  - Las variables temporales se les asignará un nombre específico.
  - Las variables booleanas tendrán nombres que impliquen verdadero o falso y sean positivos.
  - Nombres específicos para las constantes.
  - Se identificarán las variables globales (g_).
  - Los nombres tendrán un formato previamente definido (Formato definido en los estándares de programación).
  - Se evitará el uso de abreviaciones.
  - Se evitarán nombres con significados similares.
  - No se utilizarán números en los nombres.

- **Nombramiento de variables**
  - Se evitan lo números mágicos.
  - Se anticiparán las divisiones entre cero.
  - Se evitarán las comparaciones de tipo mixto.
  - Se evitarán sumas y restas en números que tengan magnitudes muy diferentes.
  - Se evitarán cadenas y caracteres mágicos.
  - Se asegurará de que todos los índices del arreglo estén dentro de los límites del arreglo.

- **Técnicas de organización de sentencias**
  - Se el código para que las dependencias sean obvias.
  - Se documentarán las dependencias poco claras con comentarios.
  - Se buscará que el código se lea de arriba abajo, en las declaraciones cuyo orden no importe.
  - Se agruparán sentencias relacionadas.

- **Uso de condicionales**
  - Se escribirá primero la ruta nominal a través del código, luego escribe los casos inusuales.
  - Se asegurará que todos los casos estén cubiertos.
  - Se remplazará las sentencias if por switch en caso de ser necesario y posible.
  - Se evitará inventar variables falsas para poder usar la sentencia case.
  - Se utilizará la cláusula default solo para detectar valores por defecto legítimos.

- **Bucles de control**
  - No se utilizará un ciclo for cuando un ciclo while es más apropiado.
  - Se evitarán bucles vacíos.
  - Se colocarán los contadores que controlan los ciclos del bucle al principio o al final del bucle.
  - Cada bucle tendrá una sola función.

- Se asegurará que el bucle termine.
- Cada bucle tendrá un solo break.
- Se limitará el anidamiento a 3 niveles.

- **Rutinas de calidad**
  - Se evitará el código duplicado.
  - Se buscará que las rutinas sean cohesivas.
  - Los nombres de las rutinas describirán lo que hace la misma.
  - Se evitarán el uso de verbos sin sentido en los nombres.
  - Los nombres serán tan largos como sean necesarios.
  - Las rutinas utilizarán todos los parámetros que posean.

- **Construcción de clases**
  - Se evitan las clases Dios.
  - Se evitan las clases irrelevantes.
  - Se evitan los verbos en los nombres de las clases.