I.E.S. Punta del Verde

Desarrollo de aplicaciones Web

KnowBuySell

Proyecto Fin de Grado

Autor:
Yolanda Morales Zamora

Junio 2024

1. Introduction

1.1. Introducción del proyecto

Este proyecto se ha elaborado a raíz de la necesidad de control de información por parte de los usuarios de una plataforma de compras, para gestionar esa información para su mejor organización económica, es decir, como usuario "quiero saber en qué gasto más dinero o menos dineroz a su vez con la misma información obtenida generar marketing personalizado para el usuario por parte de la entidad vendedora, es decir, como empresa "quiero conocer que es lo que más compra el usuario según cactegoría para ofrecerle otros productos que no conozca de la misma categoría".

Datos técnicos básicos del proyecto:
Nombre del proyecto: Knowbuysell.
Framework utilizado: Laravel, el framework de los artesanos de la web.
Lenguaje de programación: PHP, Blade, Javascript.
Lenguaje Latex para la realización de este informe.

1.2. Propósito

Desarrollar una aplicación que ayude a mejorar el control económico del usuario cuando compre a través de ella, y a la misma vez, ayude a mejorar el marketing de la entidad vendedora, a través de la información registrada. A través de la información registrada organizarla para beneficiar a la parte del çlientez a la parte del "servidor".

1.3. Objetivos del proyecto

Objetivos del proyecto:

- a) control por parte del usuario de la información que generan sus compras realizadas
- b) el control por parte de la entidad vendedora de la información de compra del cliente para ofrecerle más productos o servicios (mejora de marketing).

  En los apartados a) y B) se reutiliza la misma información sólo que según el enfoque del cliente o servidor variará su gestión o uso.

1.4. Coste del proyecto

1.4.1. Costes de desarrollo

Los costes directos comprenden los costes de personal. Para calcular el coste de personal es necesario saber cuál es el salario bruto del trabajador y los costes sociales. Para llevar a cabo esta planificación se tiene en cuenta un contrato a tiempo completo, en el que la empresa debe abonar una cuota patronal a la seguridad social del 23,60 % actualmente en España.

Estableciendo el coste medio de la hora por parte de un programador en España en 20 euros y siendo 500 las horas totales planificadas para el desarrollo del proyecto completo, se obtiene una remuneración total bruta de 10.000 euros. Tabla 1:

Concepto	Coste
Remuneración bruta	10.000 €
Costes Seguridad Social	2.360 €
TOTAL	12.360 €

Cuadro 1: Costes directos

Para la obtención de los costes indirectos, en este caso, hay que tener en cuenta la amortización del equipo utilizado por el trabajador.
La tabla 1 muestra aquellos dispositivos hardware que están expuesto a un desgaste en el tiempo en el que se elabora el el proyecto:

1.4.2. Costes de implantación

Debemos tener en cuenta que a continuación damos estimaciones de los costes de implantación:

- En cuanto a Hosting y Servidores:

  Hosting Compartido: 5 a 20 €/ mes

  Servidor VPS: 20 a 100 €/ mes

  Servidor Dedicado: 80 a 300 €/ mes

  Servicios en la Nube (AWS, Google Cloud, Azure): 50 a 200 €/mes

- Para un primer mes de operación:

  Costo Inicial de Hosting: 50 a 300 €/ anual

  Registro de Dominio: 10 a 15 €/ anual

- Certificados SSL: 0 a 200 €/anual (Let's Encrypt gratuito o pagos de proveedores como Comodo)

  Subtotal Infraestructura ((Primer Mes): 60 a 515 €

- Licencias y Herramientas

  IDE y Herramientas de Desarrollo: utilizamos herramientias de código abierto y gratuitas como Viscual Studio Code a modo de IDE. Licencias de Software: Depende de necesidades específicas, pero generalmente bajo costo

  Subtotal Licencias y Herramientas = 0 €

- Seguridad y Auditorías

  Firewall y Seguridad en la Nube (opcional): 20 a 100 €/mes

  Auditorías de Seguridad: 500 a 1,500 €/ mes

  Subtotal Seguridad y Auditorías: 520 a 1,600 €/ mes

  Total del importe estimado:

  Coste fijo: 12.360 €

  Coste variable: 580 a 2.115 €/ mes

2. Análisis del sistema

2.1. Introducción

Detallar el proyecto a realizar: análisis, diseño, implementación o codificación y prueba.

2.2. Análisis de requisitos

El proyecto quiere obtener la mejora en la información económica registrada para el usuario de la aplicación así como la mejora a nivel de marketing para la empresa que presta el servicio de venta. El usuario debe poder realizar la compra de productos, descargar la información y crear un documento formato PDF, acceder a su registro, a la información de sus compras, poder comprar, modificar la compra y eliminar la compra. El usuario no podrá acceder a la información de otros usuarios, sólo a la suya propia. El administrador o empleado de la empresa propietaria de la aplicación, podrá acceder a toda la información de las compras realizadas por cada usuario, podrá modificar, eliminar cualquier registro de la base de datos, podrá obtener la información en formato gráfica de las categorías de compras realizadas por el usuario con la finalidad de mejorar el marketing para poder aumentar la compras, y por tanto los beneficios.

2.2.1. Requerimientos funcionales

Los requisitos funcionales de una aplicación web son las funciones y características específicas que la aplicación debe tener para cumplir con los objetivos y necesidades de los usuarios. En nuestro proyecto, aunque algunos no se hayan conseguido como metas serían los siguientes:

- Inicio de sesión (Conseguido)
- Registro de usuario (Conseguido)
- Gestión de perfiles (Conseguido)
- CRUD para realizar compras (Conseguido)
- Integración de terceros como un sistema de pagos (No conseguido)
- Medidas de seguridad: cifrado de claves de usuario (Conseguido)
- doble verificación (No conseguido)

2.2.2. Requerimientos no funcionales

Los requisitos no funcionales de una aplicación web son las características que definen la calidad y el comportamiento del sistema en lugar de las funciones específicas que realiza.

En nuestro caso serían los siguientes:

- Rendimmiento: rapidez que se consigue con Laravel
- Disponibilidad (no conseguido)
- Seguridad: autenticación, CSRF (conseguido)
- Usabilidad: intuitiva y claridad en el diseño (conseguido)
- Mantenibilidad: modularización del código a través de Modelo Vista Controlador, documentación (conseguido)
- Cumplimiento Normativo: GDPR o Legislación de Protección de datos.

2.3. Casos de uso: Diagramas y Narrativas de Caso de Uso

¡¡¡¡VOY POR AQUÍ!!!!! Los diagramas de casos de uso reflejarán las funcionalidades que correspondan a cada uno de los ámbitos concretos y más importantes del sistema. En la narrativa de los casos de uso se especifica las funciones, tareas e interacciones que se dan lugar en el sistema, es decir, se describen detalladamente cada uno de los casos de uso. Explica que son estos diagramas y realiza los pertinentes para tu aplicación. https://ingsotfwarekarlacevallos.wordpress.com/2015/06/04/uml-casos-de-uso/

EJEMPLO DE NARRATIVAS DE CASOS DE USO

DIAGRAMA DE CASO DE USO: GESTION DE DOCUMENTOS

- Caso de uso 7: CARGAR DOCUMENTOS
  Nombre del caso de uso: Cargar documentos.
  Actor Principal: Usuario.
  Condiciones de entrada: El usuario tiene que estar registrado en el sistema.
  Flujo de eventos:
  1. El usuario selecciona el objetivo, dentro del módulo y del ciclo correspondiente, que cumple el documento que va adjuntar.
  2. El sistema muestra una lista con los nombres de los documentos que hasta ahora han sido subidos.
  3. El usuario selecciona “Subir documento”.
  4. El sistema le solicita que examine donde tiene guardado dicho documento.
  5. a. El usuario selecciona el documento y confirma.
  6. a. El sistema lo almacena añadiéndole un numero identificativo y mostrándolo en el conjunto

de documentos que se encuentran en la ruta especificada.
Camino alternativo:
5. b. El usuario selecciona cancelar la acción.
6. b. El sistema regresa al paso 2
Condiciones de salida: El sistema almacena el nuevo documento.

- Condiciones de uso 8 : DESCARGAR DOCUMENTOS (descripción como el caso de uso 7)
- Caso de uso 9: ELIMINAR DOCUMENTOS. (descripción como el caso de uso 7)

3. Diseño del sistema

3.1. Introducción

En el presente capítulo se tratará ver cómo llevar a cabo el diseño teniendo como punto de vista el dominio de la solución a dicho problema. Por tanto con este apartado se busca diseñar con destreza una solución que satisfaga los requisitos. Se desarrollará una solución lógica, donde se llevarán a cabo diferentes fases:

- Diagrama de clases.
- Diseño de la base de datos.
- Diseño de la interfaz.

***NOTA: si usas otro framework en lugar de una base de datos SQL, cambia los apartados necesarios.

3.2. Diagrama de clases

Diagrama que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

3.3. Diseño de la Base de Datos.

3.3.1. Diseño Lógico.

Entidades, atributos, claves primarias, identificadores únicos, relaciones entre las entidades.

3.3.2. Diseño Conceptual.

Tablas, campos, tipos de datos, restricciones, claves primarias, claves foráneas, triggers de validadación si los hubiera.

3.4. Diseño de la Interfaz.

En este apartado se define cual va a ser la apariencia visual de la aplicación: interfaz visual entre el usuario y la aplicación. En este apartado se debe explicar cómo son las plantillas en las que se basa la apariencia de nuestra interfaz así como cada una de las pantallas que tengamos.

4. Implementación

4.1. Introducción

En este momento ya se encuentra definido el problema y la solución, por lo que lo siguiente será transformar el modelo obtenido en las actividades anteriores en código fuente.

4.2. Arquitectura Cliente/Servidor

Explicar en qué consiste

4.3. Lenguajes de Programación.

Explicar un poco cada uno de los lenguajes de programación que has utilizado para la realización de tu proyecto.

4.4. Herramientas de Desarrollo

Explicar que herramientas has utilizado para desarrollar tu aplicación

4.5. Codificación

Todo el código desarrollado se encuentra disponible en el pendrive / moodle que se adjunta con esta memoria (a determinar por el departamento de informática).

5. Pruebas de software

5.1. Introducción

En el desarrollo de cualquier software, una de las actividades asociadas a este proceso es la prueba. En este capítulo se detallarán las pruebas realizadas al software desarrollado. Se intentará valorar la calidad de la aplicación generada, así como detectar y corregir posibles errores. Breve descripción de que son las pruebas del software.

5.2. Técnicas de Prueba

Explicación de las pruebas de caja blanca y caja negra

5.2.1. Pruebas de caja blanca o enfoque estructural

Explicar qué son y en qué consisten este tipo de pruebas. Nota: en caso de proyectos amplios es posible que no se realice documentación sobre ninguna de estas pruebas puesto que la documentación de las mismas sería muy tediosa. Menciona los aspectos que se han tenido en cuenta en caso de estar en esta situación.

5.2.2. Pruebas de caja negra o enfoque funcional

Explicar qué son y en qué consisten este tipo de pruebas. Estas pruebas se realizan tras la fase de codificación y nos van a determinar si el requisito de una aplicación es parcial o completamente satisfactoria. Puedes aplicar pruebas mediante JUnit. Si tu programa es muy grande pon las que te resulten más importantes.

6. Conclusiones

6.1. Resumen de conclusiones

6.2. Propuestas Futuras (opcional)

Ampliaciones y/o mejoras que podrías añadir a tu proyecto en un futuro.

NOTAS INTERNAS PARA TENER EN CUENTA EN EL LATEX NO OLVIDAR BORRAR ANTES DE ENTREGAR

We hope you find Overleaf useful, and do take a look at our help library for more tutorials and user guides! Please also let us know if you have any feedback using the Contact Us link at the bottom of the Overleaf menu — or use the contact form at https://www.overleaf.com/contact.

Referencias