Documentación del Proyecto Semestral para Programación Web 2023-1

Integrantes: Felipe González
Docente: Victor Rosendo Lugo
Fecha: 10 de Julio, 2023 – Sección: 007D

1. Introducción

En este documento se presenta el desarrollo y documentación de un proyecto de tienda online para una tienda ficticia de accesorios de mascotas, llamada “GoWest”.

El tiempo total para el desarrollo del proyecto es de 18 semanas, correspondiente al semestre completo, y se pretende que sea organizado de la siguiente forma:

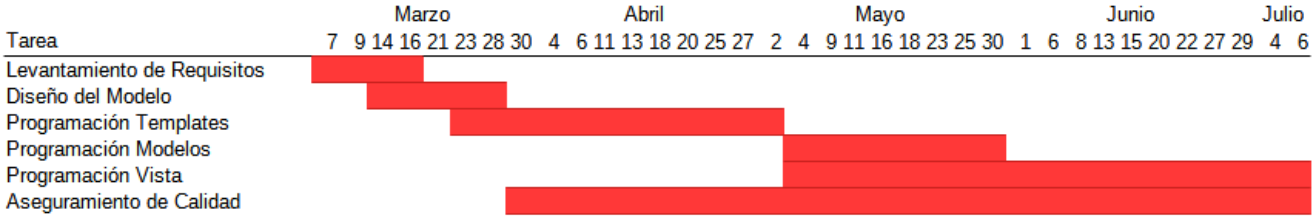

Figura 1: Carta Gantt

El resultado de la tarea “Levantamiento de Requisitos” se muestra en la sección 3, “Requisitos y Casos de Uso”. El resultado de la tarea “Diseño del Modelo” se muestra en la sección 4, “Modelo de Base de Datos”. Las tareas “Programación Templates”, “Programación Modelos”, y “Programación Vista” corresponden al desarrollo del código de cada parte del patrón MTV que se usará en este proyecto. Además, la tarea “Programación Modelos” conlleva la creación de la base de datos que se usará. La tarea “Aseguramiento de Calidad” corresponde al aseguramiento del correcto funcionamiento inter-operativo de las tres tareas anteriores.

En la sección 5, se detalla el uso de una API externa, y el desarrollo de la misma para su uso en este proyecto.

2. Descripción del Sistema

El sistema utiliza el modelo MTV de Django, en que la lógica del sistema y el procesamiento de datos se programa en *vistas*, mientras que el resultado a mostrar al cliente se describe en *templates*. La base de datos se *modela* mediante código y se realizan conexiones con ella mediante las *vistas*. De esto, los *templates* correponden al *front-end*, mientras que las *vistas* y los *templates* correpsonden al *back-end*.

Un usuario (el *cliente* de la arquitectura *cliente-servidor*) interactúa con las páginas, generadas mediante los *templates*, lo que corresponde a la capa de presentación. La solicitud generada se envía al

servidor (de la arquitectura *cliente-servidor*) y se procesa mediante las *vistas*, lo que corresponde a la capa de negocios, y se le retorna una nueva página. Si una *vista* requiere de la base de datos, hace uso de los *modelos* definidos para conectarse a ella y obtener los datos, lo que corresponde a la capa de datos.

Un usuario puede *autenticarse* como "cliente" (de la tienda) o "administrador" (de la página). Esto se hace mediante un correo y una contraseña, y se cuenta con un sistema de recuperación basado en preguntas de seguridad. Ciertas funcionalidades del sistema está restringidos a los usuarios según su rol: Los usuarios no-autenticados, o "visitas", pueden navegar a través del catálogo de productos, pero no pueden añadirlos a un carrito ni realizar compras; Los "clientes" pueden navegar el catálogo, añadir productos a su carrito y realizar la compra del mismo, además de administrar sus datos y direcciones de envío; Los administradores manejan los estados de los productos, las categorías a los que pertenecen, y otros administradores, así como ver los estados de los clientes y las compras que han hecho.

3. Requisitos y Casos de Uso

3.1 Requisitos Funcionales

Se han definido los siguientes requisitos funcionales para el sistema:

3.1.1. Mostrar a los visitantes una galería de productos.
3.1.2. Mostrar los detalles de productos (precio, imagen, y descripción) a visitantes.
3.1.3. Poder registrar clientes nuevos.
3.1.4. Añadir productos a los carritos de compra de los clientes.
3.1.5. Procesar el pago de las compras de los clientes.
3.1.6. Mostrar los datos de la cuenta de Cliente, y permitirle editarlos. Esto incluye añadir, editar, o eliminar direcciones de envío.
3.1.7. Permitir a los clientes suscribirse a la Fundación "Ayuda a un Peludo" (ver Sección 4)
3.1.8. Manejar inventario de productos – Añadir, Actualizar, Activar/Desactivar productos y categorías
3.1.9. Crear y Eliminar administradores de página.
3.1.10. Cambiar los estados de compras (ver sección 2.3.5).

2.2 Requisitos No-Funcionales

Se han definido los siguientes requisitos no-funcionales para el sistema:

3.2.1. El sistema debe levantarse sobre Django.
3.2.2. El sistema debe trabajar con una base de datos Oracle.
3.2.3. El sistema debe manejar APIs internas.

2.3 Reglas de Negocio

Se han definido las siguientes reglas de negocio.

3.3.1. No se puede realizar una venta sobre un carrito vacío.
3.3.2. Cada cliente debe tener al menos una dirección registrada.
3.3.3. Un administrador sólo puede ser creado o eliminado por otro administrador.
3.3.4. Los productos y categorías pueden desactivarse, pero no pueden eliminarse, a fin de mantener registros históricos de ventas realizadas con productos antiguos.
3.3.5. Las compras tienen distintos estados, “Carrito”, “Pagada”, “Despachada”, y “Completada”
  3.3.5.1. Una compra es “Carrito” antes de que el cliente pague por ella. Un cliente siempre tiene una compra de estado “Carrito”, y tiene sólo una.
  3.3.5.2. Una compra es “Pagada” entre el momento en que el cliente la paga y un administrador indica que el envío de la compra ha sido despachada.
  3.3.5.3. Una compra es “Despachada” entre el momento en que el envío de esta se ha realizado y el cliente indica que ha recibido el envío.
  3.3.5.4. Una compra es “Completada” cuando el cliente indica haber recibido una compra “Despachada”.

2.4 Casos de Uso

Se han definido los siguientes casos de uso.

3.4.1. Un visitante puede *Registrarse* como cliente nuevo.
3.4.2. Un cliente puede *Añadir a su carrito* productos del catálogo.
3.4.3. Un cliente puede *Pagar* su compra.
3.4.4. Un cliente puede *Marcar recepción* de una compra “Despachada”.
3.4.5. Un cliente puede *Actualizar* sus datos.
3.4.6. Un cliente puede *Añadir/Editar/Eliminar* direcciones de envío.
3.4.7. Un cliente puede *Suscribirse* a la Fundación “Ayuda a un Peludo” (ver sección 4).
3.4.8. Un cliente puede *Recuperar su cuenta* en caso de olvidar su contraseña.
3.4.9. Un administrador puede *Actualizar* sus datos.
3.4.10. Un administrador puede *Añadir/Editar/Eliminar* productos.
3.4.11. Un administrador puede *Añadir/Editar/Eliminar* categorías.
3.4.12. Un administrador puede *Ver* clientes y sus datos.
3.4.13. Un administrador puede *Ver* ventas y sus detalles.
3.4.14. Un administrador puede *Añadir/Editar/Eliminar* administradores.

3. Modelo de Base de Datos

Dada la descripción del proyecto, los requisitos levantados y los casos de uso, se ha diseñado el siguiente modelo de base de datos:

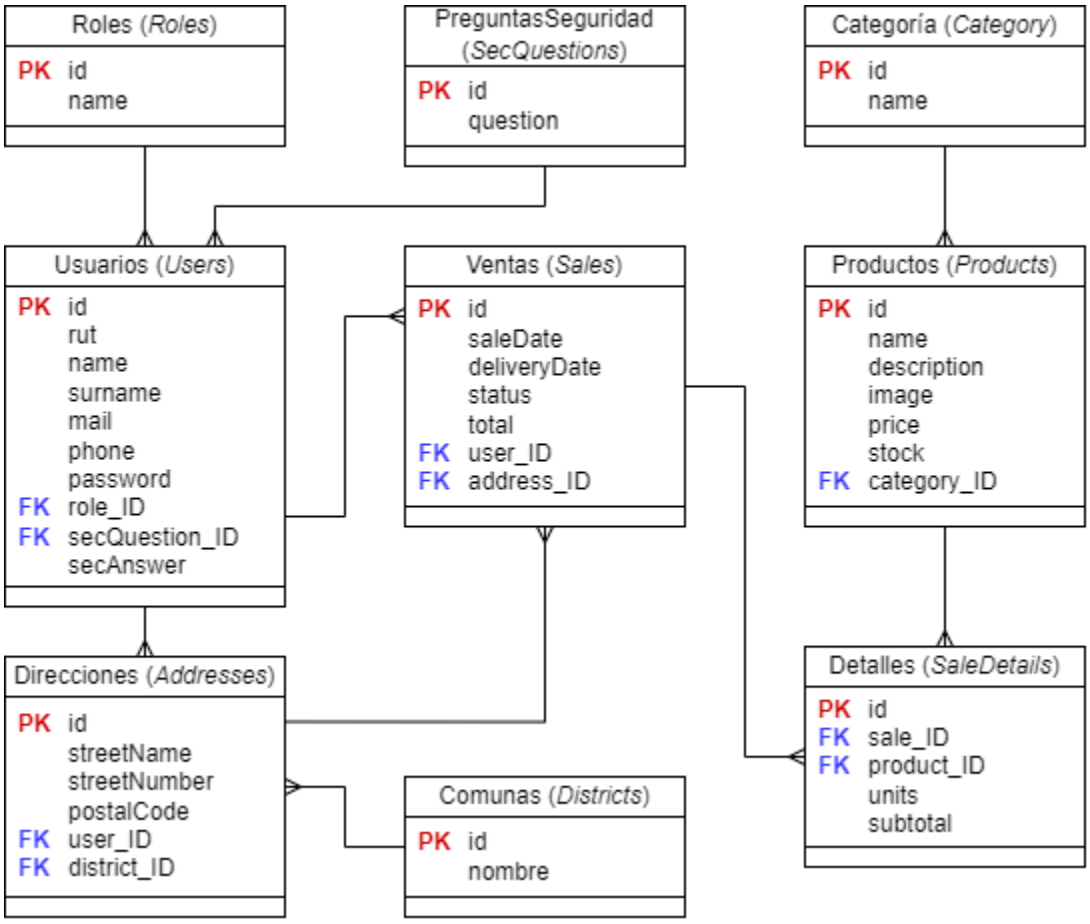

Figura 2: Modelo de Base de Datos

Se hacen las siguientes observaciones respecto al modelo presentado:

- *Roles* indica si un usuario es un "Cliente" o un "Administrador".
- La tabla *PreguntasSeguridad*, su relación con la tabla *Usuarios*, y el campo *secAnswer* de esta última se usan para la recuperación de cuenta.
- *Detalles* corresponde a una tabla que relaciona cada *Venta* con los *Productos* correspondientes.
- *Ventas* está relacionada con *Usuarios* y con *Direcciones*. Aunque una tabla normalizada correctamente no tendría la relación *Ventas-Usuarios*, se ha decidido añadir esta relación para facilitar el desarrollo del sistema.

4. API externa: Fundación "Ayuda a un Peludo"

Para la creación de este proyecto, se solicitó el uso de APIs de servicios externos. Para ello, se desarrolló un servicio API propio en un servidor externo, ubicado en http://dintdt.c1.biz/aup. Conceptualmente, este servicio corresponde a suscripciones a una fundación ficticia, "Ayuda a un

Peludo", que se centra en la adopción y tenencia responsable de mascotas, cuya página fue desarrollada como parte del curso durante las primeras clases, y que se encuentra ubicada en https://feigonzalez.github.io/test20230309/index.html.

El servicio API desarrollado cuenta con los siguientes endpoints:

- `/getSub.php` [GET]: Retorna, como un objeto JSON, los datos de la suscripción de una persona identificada por su RUT, entregado como parámetro "`rut`" a la API. Se usa en la vista `clientFoundation`, para indicar al cliente si está suscrito, y, de estarlo, la fecha hasta la que la suscripción está vigente. Además, se usa en las vistas `cart` y `checkout`, para aplicar un descuento del 10% a las compras de los clientes suscritos.
- `/getAllSubs.php` [GET]: Retorna, como un objeto JSON, los datos de todas las suscripciones registradas. Se usa en la vista `adminClients`, para indicar, por cada cliente, si se encuentra actualmente suscrito a la fundación.
- `/postSub.php` [POST]: Registra una nueva suscripción, asignada a una persona identificada por su RUT, entregado como parámetro "`rut`" a la API. Se usa en la vista `clientFoundation` para que un cliente pueda suscribirse, o renovar su suscripción, a la fundación.