|  |  |  |
| --- | --- | --- |
| UNIVERSIDAD INDUSTRIAL DE SANTANDER  Escuela de Ingeniería de Sistemas e Informática | | Fecha límite de recepción  **Domingo 13 Diciembre-2020 - 11 pm.**  [<oop.eisi@gmail.com>](mailto:%3coop.eisi@gmail.com%3e) |
| **DESARROLLO DE SOFTWARE ORIENTADO A OBJETOS** | | |
| **ANÁLISIS Y DISEÑO** | | |
| **NOMBRE CLAVE DEL PROYECTO** | ***Burguerman*** | |
| **RESPONSABLES** | ***Jhon Lopez Ruiz, Jhoann Martienz Oviedo*** | |
| **DESCRIPCIÓN DE LA APLICACIÓN - Actualizada** | | |
| La funcionalidad final del aplicativo “Burguerman” es permitir a sus usuarios la compra de productos de una forma eficiente, intuitiva y muy cómoda, visualizar las descripciones de cada producto y si el cliente u/o usuario lo desea ver los productos por categorías, haciendo así la búsqueda más amena y acorde a sus gustos para así poder tomar un pedido acorde a sus gustos, para el uso de esta aplicación es necesario la creación de un perfil, luego de creado este se le permite al usuario agregar uno o varios productos a la “canasta” desde acá podremos personalizar nuestra orden según nuestras preferencias pero, acorde a la materia prima de la cual se disponga, elegir diferentes métodos de pago, visualizar los pedidos en curso, y el historial de pedidos los cuales abarcan periodos de dos semanas, también se implementara un seguimiento de pedido desde que se hace la compra del producto hasta su respectiva entrega.  Ahora sus funcionalidades administrativas permiten la modificación de productos, precios, ~~gestionar ofertas~~, dar un informe de los productos más vendidos y menos vendidos y hacer modificaciones al estado de los pedidos. | | |
| **ANÁLISIS DE TEXTO** | | |
| La funcionalidad final del aplicativo “Burguerman” es permitir a sus usuarios la compra de productos de una forma eficiente, intuitiva y muy cómoda, visualizar las descripciones de cada producto y si el cliente u/o usuario lo desea ver los productos por categorías, haciendo así la búsqueda más amena y acorde a sus gustos para así poder tomar un pedido acorde a sus gustos, para el uso de esta aplicación es necesario la creación de un perfil, luego de creado este se le permite al usuario agregar uno o varios productos a la “canasta” desde acá podremos personalizar nuestra orden según nuestras preferencias pero, acorde a la materia prima de la cual se disponga, elegir diferentes métodos de pago, visualizar los pedidos en curso, y el historial de pedidos los cuales abarcan periodos de dos semanas, también se implementara un seguimiento de pedido desde que se hace la compra del producto hasta su respectiva entrega.  Ahora sus funcionalidades administrativas permiten la modificación de productos, precios, dar un informe de los productos más vendidos y menos vendidos y hacer modificaciones al estado de los pedidos. | | |
| **TABLA DE SUSTANTIVOS Y VERBOS** | | |
| |  |  | | --- | --- | | **Sustantivos** | ***Verbos*** | | Aplicativo Burgerman | *Permite ver y comprar productos* | | Cliente/Usuario | *Visualizar productos*  *Agregar productos al carrito*  *Realizar compras*  *Realizar pagos*  *Ver pedidos realizados* | | Producto | *Pertenece a una categoría*  *Existencia según la materia prima*  *posee descripción*  *posee precio* | | Categoría | *Cuenta con varios productos de una misma clase* | | Canasta | *Agregar productos*  *Personalizar productos*  *Eliminar productos* | | Pedido | *Posee datos de los productos y del usuario* | | Método de pago | *Tiene diferentes métodos de pago para hacer valido el pedido.* | | ~~orden~~ | *Sinónimo de pedido* | | Estado de pedido | *Hace parte del pedido* | | Perfil | *Necesario para usar la aplicación* | | Precios | *Hace parte del producto* | | Historial de pedido | *Forma parte del perfil* | | Materia prima | *Hace parte del producto* | | | |
| **ESCENARIOS** | | |
| * Un usuario dese hacer un pedido de una hamburguesa “killerqueen” pero este aún no se encuentra registrado en la aplicación, para proceder a hacer el pedido primero debe estar registrado en la aplicación o de lo contrario no podrá. * El ingeniero Rodolfo desea comprar una hamburguesa y la desea con pan francés, pero en los ingredientes no se encuentra el pan francés, él debe elegir entre otros ingredientes los cuales si se encuentran disponibles en la personalización de su pedido. * Martin procede a hacer un pedido, pero luego de un rato el desea desplazarse de dirección y quiere hacer cambio de entrega del pedido, si el pedido aún no ha salido del restaurante podrá hacer el cambio dentro de la aplicación, pero si este ya ha salido a su dirección de entrega el deberá llamar al domiciliario para hacer el debido cambio. * Un cliente realiza un pedido dentro de la aplicación, pero al momento de proceder a pagar su método de pago es invalido debido a que no posee fondos suficientes para realizar el pedido, el tendrá la opción de cancelar el pedido o pagar en efectivo cuando se le entregue. * Un cliente realiza un pedido y está ansioso por recibir su orden, entra a la aplicación para estar pendiente minuto a minuto del estado de su pedido, unos momentos antes de recibir su orden se le notifica que su pedido va en camino a su dirección y al paso de un rato llega exitosamente su orden. | | |
| **TARJETAS CRC** | | |
| |  |  | | --- | --- | | Cliente/Usuario | Colaboradores  Perfil | | *Responsabilidades*  *Registrarse*  *Iniciar sesión*  *Crear un perfil*  *Ver productos*  *Añadir productos a la canasta*  *Realizar compra*  *Ver historial de pedidos* |  |  |  | | --- | --- | | Producto | Colaboradores  Categoría  Materia prima | | *Responsabilidades*  *pertenece a una categoría*  *tiene descripción*  *existe siempre y cuando exista la materia prima*  *tiene precio* |  |  |  | | --- | --- | | categoría | Colaboradores | | *Responsabilidades*  *Etiquetar los productos* |  |  |  | | --- | --- | | Canasta | Colaboradores  Producto  Cliente | | *Responsabilidades*  *Repositor de productos durante el proceso de pedido* |  |  |  | | --- | --- | | Pedido | Colaboradores  Producto  Cliente  Perfil | | *Responsabilidades*  *Almacenar productos seleccionados*  *Almacenar datos de envío del cliente*  *Dar lugar al pago*  *Posee estados dependiendo del avance del pedido* |  |  |  | | --- | --- | | Perfil | Colaboradores | | *Responsabilidades*  *Posee los datos del cliente necesarios para hacer un pedido* |  |  |  | | --- | --- | | Administrador | Colaboradores  Producto  Categoría  Colección  Pedido | | *Responsabilidades*  *Gestionar (productos, categorías y materia prima)*  *Ver productos más vendidos y menos vendidos*  *Ver ventas realizadas*  *Modificar (estado de pedidos)*  *Agregar número de seguimiento del pedido* | | | |
| **DISEÑO DE INTERFACES DE CLASES** | | |
| **/\*\***  **\* Abstract class Usuario**  **\* @author: Sebastian Martinez, Jhon López**  **\*/**  **public abstract class Usuario**  **{**  **String correo;**  **String password;**  **/\*\***  **\* Todo usuario se identifica con correo y contraseña**  **\*/**  **public Usuario(String correo,String password)**  **{**  **this.correo = correo;**  **this.password = password;**  **}**    **public String getCorreo()**  **{**  **return this.correo;**  **}**    **public String getPassword()**  **{**  **return this.password;**  **}**    **public void setPassword(String password)**  **{**  **this.password = password;**  **}**  **}**  **/\*\***  **\* Con esta clase se crean los objetos cliente los cuales podran comprar los productos**  **\* @author: Sebastian Martinez, Jhon López**  **\* @version 1**  **\*/**  **public class Perfil extends Usuario**  **{**  **private String nombre;**  **private int telefono;**  **private String Direccion;**  **/\*\***  **\* El Cliente debe inicializarse con el nombre,correo electrónico, el**  **\* teléfono, direccion y una clave**  **\*/**  **public Perfil(String nombre,int telefono,String correo,String clave,String Direccion)**  **{**  **super(correo,clave);**  **}**  **public String getNombre()**  **{**  **return this.nombre;**  **}**    **public int getTelefono()**  **{**  **return this.telefono;**  **}**  **public int getDireccion()**  **{**  **return this.direccion;**  **}**    **/\*\***  **\* @parametro: Un nuevo número telefónico del usuario**  **\*/**  **public void setTelefono(int telefono)**  **{**  **this.telefono = telefono;**  **}**    **/\*\***  **\* @parametro: Un nuevo nombre del usuario**  **\*/**  **public void setNombre(String nombre)**  **{**  **this.nombre = nombre;**  **}**  **/\*\***  **\* @parametro: nombre un nuevo nombre del usuario**  **\*/**  **public void setDireccion(String direccion)**  **{**  **this.direccion = direccion;**  **}**  **}**  **/\*\***  **\* Representan grupos de productos con caracteristicas en comun definidas por el administrador**  **\***  **\* @author: Sebastian Martinez, Jhon López**  **\* @version 1**  **\*/**  **public class Categoria**  **{**  **private int Id;**  **private String nombre;**  **private String imagen;**    **/\*\***  **\* Las categorías se crean con un nombre, un Id y una imagen**  **\*/**  **public Categoria(String nombre, int Id, String imagen)**  **{**  **}**  **public int getId()**  **{**  **return this.Id;**  **}**    **public String getNombre()**  **{**  **return this.nombre;**  **}**    **public String getImagen()**  **{**  **return this.imagen;**  **}**  **public void setNombre(String nombre)**  **{**  **this.nombre = nombre;**  **}**    **public void setImagen(String imagen)**  **{**  **this.imagen = imagen;**  **}**    **}**  **/\*\***  **\* Encargada de representar cada uno de los productos de la aplicación**  **\* @author: Sebastian Martinez, Jhon López**  **\* @version 1**  **\*/**  **public class Producto**  **{**  **private String referencia;**  **private String nombre;**  **private int costo;**  **private String imagen;**  **private String descripcion;**  **private Ingredientes ingredientes;**  **private Categoria categoria;**    **/\*\***  **\* El producto debe inizializarse con un nombre, una referencia, un costo, un nombre imagen y una descripción**  **\*/**  **public Producto(String nombre,String referencia,int costo,String imagen,**  **String descripcion, Ingredientes ingredientes,Categoria categoria)**  **{**  **}**  **public String getNombre()**  **{**  **return this.nombre;**  **}**    **public String getReferencia()**  **{**  **return this.referencia;**  **}**    **public String getDescripcion()**  **{**  **return this.descripcion;**  **}**    **public int getCosto()**  **{**  **return this.costo;**  **}**    **public String getImagen()**  **{**  **return this.imagen;**  **}**    **public Ingredientes getIngredientes()**  **{**  **return this.ingredientes;**  **}**    **public Categoria getCategoria()**  **{**  **return this.categoria;**  **}**  **public void setNombre(String nombre)**  **{**  **this.nombre = nombre;**  **}**    **public void setDescripcion(String descripcion)**  **{**  **this.descripcion = descripcion;**  **}**    **public void setImagen(String imagen)**  **{**  **this.imagen = imagen;**  **}**    **}**  **public class Ingredientes**  **{**  **private int Id;**  **private String nombre;**  **private int costo;**    **/\*\***  **\* Los ingredientes se crean con un nombre, un identificador, una imagen y un costo**  **\*/**  **public Ingredientes(int Id, String nombre, int Costo)**  **{**  **}**  **public int getId()**  **{**  **return this.Id;**  **}**    **public String getNombre()**  **{**  **return this.nombre;**  **}**    **public int getCosto()**  **{**  **return this.costo;**  **}**    **public void setNombre(String nombre)**  **{**  **this.nombre = nombre;**  **}**    **public void setCosto(int costo)**  **{**  **this.costo = costo;**  **}**  **}**  **import java.util.\*;**  **/\*\***  **\* Esta clase se encargar de crear las canastas de compra para cada usuario**  **\***  **\* @author: Sebastian Martinez, Jhon López**  **\* @version 1**  **\*/**  **public class Canasta**  **{**  **private int Id;**  **private Perfil perfil;**  **private int subtotal;**  **/\*\***  **\* la canasta es unica para cada usuario y su subtotal comienza en 0**  **\*/**  **public Canasta(Perfil perfil)**  **{**  **}**  **/\*\***  **\* Con este metodo se pueden agregar productos a la canasta**  **\* @parametro: producto, producto a agregar**  **\* @parametro: cantidad, cantidad del producto**  **\*/**  **public void aggProducto(Producto producto, int cantidad)**  **{**  **}**  **/\*\***  **\* Con este metodo se pueden Personalizar productos dentro de la canasta**  **\* @parametro: producto, producto a Personalizar**  **\* @parametro: cantidad, cantidad del producto a personalizar**  **\*/**  **public void PersProducto(Producto producto, int cantidad)**  **{**  **}**    **/\*\***  **\* Con este método se pueden eliminar productos de la canasta**  **\* @parametro: producto a eliminar**  **\*/**  **public void supProducto(Producto producto)**  **{**  **}**    **/\*\***  **\* Este metodo muestra los productos agregados y su cantidad**  **\* @return detalles de la canasta**  **\*/**  **public String mostrarContenido()**  **{**  **return "";**  **}**    **/\*\***  **\* Con este metodo se vacia la canasta**  **\*/**  **public void vaciarCanasta()**  **{**  **}**    **/\*\***  **\* Con este método se modifica la cantidad de un producto agregado a la Canasta**  **\* @parametro: cantidad, número de unidades del producto**  **\* @parametro: producto, producto a modificar**  **\*/**  **public void modificarCantidad(int cantidad, Producto producto)**  **{**  **}**    **/\*\***  **\* Con este metodo se calcula el subtotal a pagar por los productos**  **\*/**  **public int calcularSubTotal()**  **{**  **return 0;**  **}**  **}**  **/\*\***  **\* esta clase crea los pedidos detallandolos con la informacion de los productos y la del usuario**  **\* @author: Sebastian Martinez, Jhon López**  **\* @version 1**  **\*/**  **public class Pedido**  **{**  **public int numero\_pedido;**  **public String fecha;**  **public String estado;**  **public Perfil usuario;**  **public int total;**  **public boolean pagado;**    **/\*\***  **\* Se crea el pedido con número de pedido,una fecha, un usuario propietario, una direccion y un estado de pedido**  **\* El estado inicial del pedido es no confirmado**  **\*/**  **public Pedido(int numero,Perfil usuario)**  **{**  **this.total = 0;**  **this.estado = "no confirmado";**  **this.pagado = false;**  **}**  **public int getNumero\_pedido()**  **{**  **return this.numero\_pedido;**  **}**    **public String getUsuario()**  **{**  **return this.usuario.getNombre();**  **}**    **public String getEstado()**  **{**  **return this.estado;**  **}** | | |
| **MODELO DE OBJETOS DEL DOMINIO**  **(DIAGRAMA DE CLASES UML)** | | |
|  | | |
| **OTROS MODELOS UML** | | |
| **Busqueda de productos**    **Añadir a la canasta**    **Eliminar producto de la canasta**    **Agregar categoría**    **Eliminar categoría**    **Agregar producto** | | |
| **OBSERVACIONES** | | |
| Después de registrado el cliente en la aplicación y hecho un pedido, si este se cancela por algún motivo los productos quedarán guardados en el carrito para una futura compra si así lo desea, estos productos podrán ser eliminados por el cliente en cualquier momento y dejar vacía la canasta para nuevas órdenes.  El usuario podrá editar cuantas veces sean necesarias su perfil acorde a sus necesidades. | | |
|  | | |