Informe de Laboratorio Trabajo Final

Tema: Trabajo Final

Nota

Estudiante	Escuela	Asignatura
Andre Renzo Añasco Huamanquispe Jhamil Yeider Turpo Añasco Melsy Melani Huamani Vargas Frank's Javier Vilca Quispe	Escuela Profesional de Ingeniería de Sistemas	Programación Semestre: I Código: 20231001

Laboratorio	Tema	Duración
Trabajo Final	Trabajo Final	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 30 junio 2023	Al 05 agosto 2023

1. Tipos de Datos:

- La página web que diseñamos es útil para la cafetería Caffe.Aqp con una óptima interacción humano-computadora y características centradas en la eficiencia operativa y la satisfacción del cliente tiene el potencial de revolucionar la forma en que las cafeterías gestionan sus operaciones y brindan servicios a sus clientes. La plataforma ofrece una solución integral que beneficia tanto a los administradores como a los clientes, creando una experiencia más conveniente y placentera en todos los aspectos. La página web se ha desarrollado con el propósito de optimizar la gestión de productos, pedidos, reservas de mesas y la generación de boletas.

2. Requisitos del sistema

- El sistema debe satisfacer los siguientes requisitos funcionales y no funcionales:
- RQ01 : El sistema debe estar disponible en Internet a través de una URL.
- RQ02 : El sistema debe permitir el inicio/cierre de sesión.
- RQ03 : El sistema debe permitir gestionar los productos ingresados y los eliminados solo a los administradores.
- RQ04 : El sistema debe permitir poder configurar cada uno de los atributos de los productos solo a los administradores.

- RQ05 : El sistema debe permitir realizar reservaciones para cada uno de los clientes..
- RQ06 : El sistema debe permitir realizar compras en línea.
- RQ07: El sistema debe enviar notificaciones por correo electrónico, cuando se realice una compra.
- RQ08: El sistema debe ser compatible con dispositivos móviles.

3. Modelado de datos

- El modelo de comidas: En esta entidad se guardarán cada uno de los productos ingresados. Los atributos por los que estará compuesto esta entidad son: id, que será su código único, name el cual será el nombre del producto, tipos el cual indicará si es bebida comida o postre, imagen para poder guardar una imagen en el, desc la cual es una breve descripción del producto y por último price que será el precio del producto.

Listing 1: Modelado de comidas

```
class comidas(models.Model):
    id = int
    name = models.CharField(max_length=100)
    tipo = models.CharField(max_length=2, choices=Tipo, default='4')
    img = models.ImageField(upload_to='pics')
    desc = models.TextField(default='')
    price = models.IntegerField()
```

- El modelo de clientes: En esta entidad se guardaran los clientes que ingresamos en nuestra bd. Los atributos que contendrá nuestra entidad son nombre, el cual obtendrá el nombre de este, email, el cual contendrá su email donde se enviará su boleta de compra y por último su teléfono para poder comunicarnos con el.

Listing 2: Modelo de clientes

```
class Cliente(models.Model):
   nombre = models.CharField(max_length=100)
   email = models.EmailField()
   telefono = models.CharField(max_length=20)

def __str__(self):
    return self.nombre

class Meta:
    ordering = ['nombre']

def get_absolute_url(self):
    return reverse('cliente-detail', args=[str(self.id)])
```

- El modelo de Pedido: En esta entidad podremos guardar los valores de los Pedidos realizados, comenzando por el cliente el cual realizó dicho pedido, la fecha en la que se adquirió el pedido y los productos adquiridos, También tendrá un modelo de nombre ESTADOSPEDIDO, para saber si esta pendiente, ya se esta realizando o se completo.

Listing 3: Modelo de pedido

```
class Pedido(models.Model):
       cliente = models.ForeignKey('Cliente', on_delete=models.CASCADE)
       fecha = models.DateTimeField(auto_now_add=True)
       productos = models.ManyToManyField('Producto', through='ItemPedido')
     ESTADOS_PEDIDO = [
        ('P', 'Pendiente'),
        ('E', 'En preparacin'),
        ('C', 'Completado'),
     ]
      estado = models.CharField(max_length=1, choices=ESTADOS_PEDIDO, default='P')

      def __str__(self):
          return f"Pedido {self.cliente} - {self.id}"
```

- El modelo ITEMPEDIDO: Esta entidad viene a ser de relación, entre producto y pedido, donde tendrá el llamado de ambas tablas y se le añadirá el valor de cantidad.

Listing 4: Modelo ITEMPEDIDO

```
class ItemPedido(models.Model):
      pedido = models.ForeignKey('Pedido', on_delete=models.CASCADE)
      producto = models.ForeignKey('Producto', on_delete=models.CASCADE)
      cantidad = models.PositiveIntegerField()

      def subtotal(self):
          return self.cantidad * self.producto.precio

      def __str__(self):
      return f"Pedido {self.producto.nombre}, Cantidad {self.cantidad}, Precio
          unitario {self.producto.precio}"
```

- El modelo carrito: Esta entidad será de referencia para saber qué productos se adquirió.

Listing 5: Modelo de carrio

```
class Carrito(models.Model):
      cliente = models.OneToOneField('Cliente', on_delete=models.CASCADE)
      productos = models.ManyToManyField('Producto', through='ItemCarrito')

      def __str__(self):
          return f"Carrito de {self.cliente.nombre}"
```

- El modelo ReservaMesa: Esta entidad sirve para poder guardar la información de las mesas que serán reservadas, se guardará el cliente, la fecha en la que se hará la reserva, la hora en la que se acercara el cliente, y la cantidad de personas que asistiran.

Listing 6: Modelo de ReservaMesa

```
class ReservaMesa(models.Model):
      cliente = models.ForeignKey('Cliente', on_delete=models.CASCADE)
      fecha = models.DateField()
      hora = models.TimeField()
      cantidad_personas = models.PositiveIntegerField()

      def __str__(self):
```

```
        return f"Reserva para {self.cantidad_personas} personas el
            {self.fecha} a las {self.hora}"
```

4. Diccionario de datos

- **Categoria:**

  Campos: nombre, descripcion Descripción: Este modelo almacena información sobre las categorías de productos disponibles en la cafetería. Cada categoría tiene un nombre y una descripción.

	id [PK] bigint	nombre character varying (50)
1	1	Bebidas
2	2	Postres
3	3	Comidas
4	4	Otros

Figura 1: Tabla de categoria

- **Producto:**

  Campos: categoria, nombre, descripcion, precio, imagen Descripción: El modelo de Producto guarda detalles sobre los productos ofrecidos en la cafetería. Cada producto está asociado a una categoría y tiene un nombre, descripción, precio y una imagen.

	id [PK] bigint	nombre character varying (100)	imagen character varying (100)	descripcion text	precio numeric (10,2)	categoria_id bigint
1	1	Colado	pics/descarga.jpeg	Café colado	10.00	1
2	2	Americano	pics/americano.jpeg	Café americano	15.00	1

Figura 2: Tabla producto

- **Cliente:**

  Campos: nombre, apellido, correo, telefono Descripción: El modelo de Cliente almacena información de los clientes que visitan la cafetería. Incluye el nombre, apellido, correo electrónico y número de teléfono de los clientes.

	id [PK] bigint	nombre character varying (100)	email character varying (254)	telefono character varying (20)
1	1	Yeyder	miemail@gmail.com	123456
2	2	Juan	juan@gmail.com	789456

Figura 3: tabla de cliente

- **Pedido:**

  Campos: cliente, producto, fecha-pedido, estado Descripción: El modelo de Pedido registra los pedidos realizados por los clientes. Cada pedido tiene un cliente asociado, un producto solicitado, la fecha del pedido y su estado (pendiente, entregado, etc.).

	id [PK] bigint	fecha timestamp with time zone	estado character varying (1)	cliente_id bigint
1	1	2023-08-05 08:06:17.643016-05	P	1

Figura 4: Tabla pedido

- **itemPedido:**

  Campos: pedido, cantidad Descripción: El modelo de DetallePedido guarda información sobre los productos y las cantidades asociadas a un pedido específico. Cada detalle de pedido está vinculado a un pedido y registra la cantidad de un producto solicitado.

	id [PK] bigint	cantidad integer	precio_unitario numeric (10,2)	pedido_id bigint	producto_id bigint
1	1	2	15.00	1	2
2	2	2	10.00	1	1

Figura 5: Tabla itemPedido

- **Carrito:**

  Campos: cliente Descripción: El modelo Çarritorepresenta el carrito de compras de un cliente. Está asociado a un cliente específico y actúa como un contenedor temporal para los productos que el cliente planea comprar.

	id [PK] bigint	cliente_id bigint
1	3	1
2	4	2

Figura 6: Tabla carrito

- **ItemCarrito:**

  Campos: carrito, producto, cantidad Descripción: El modelo ÏtemCarritoregistra los productos y las cantidades agregadas al carrito de compras de un cliente. Cada elemento del carrito está vinculado a un carrito, un producto específico y la cantidad deseada.

	id [PK] bigint	cantidad integer	carrito_id bigint	producto_id bigint
1	1	2	3	1
2	2	2	3	2
3	3	2	4	2

Figura 7: Tabla itemCarrito

5. Diagrama Entidad-Relación (ERD)

En esta sección se mostrará los modelos creados con sus respectivos campos en el proyecto y la relación entre cada uno de ellos.

Inicialmente se mostrará cada uno de las tablas y sus relaciones, para una mejor comprensión, las tablas genteradas son: categoria, producto, cliente, carrito, pedido, itemcarrito e itempedido.

5.1. modelo categoria

Consta de un campo "nombrez tiene una relación de uno a muchos con el modelo producto, esto porque una categoría (bebidas, postres y entre otros) pueden tener muchos productos.

5.2. modelo producto

Consta de de campos como nombre, imagen, descripción, precio y tiene una relación de mushos a uno con categoría, como se explicó anteriormente, y también tiene una relación de uno a muchos con itemcarrito e itempedido (ambos).

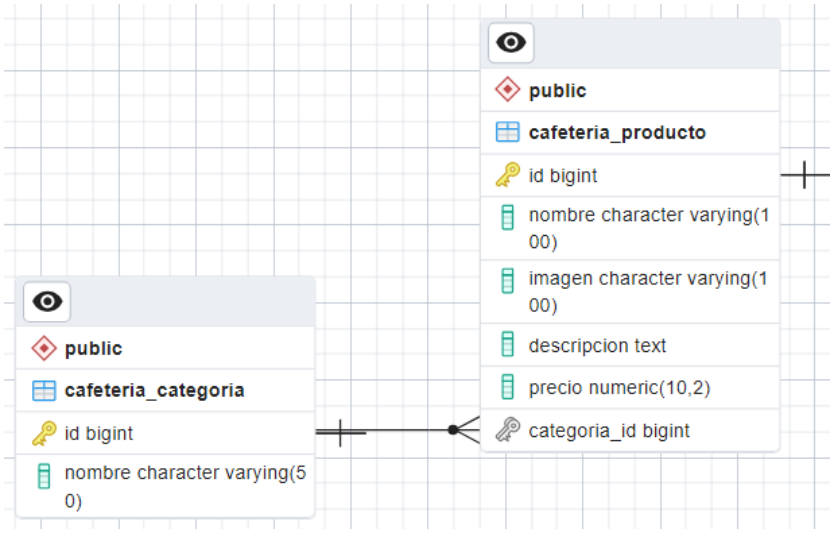

Figura 8: Figura de del modelo categoria y sus relaciones

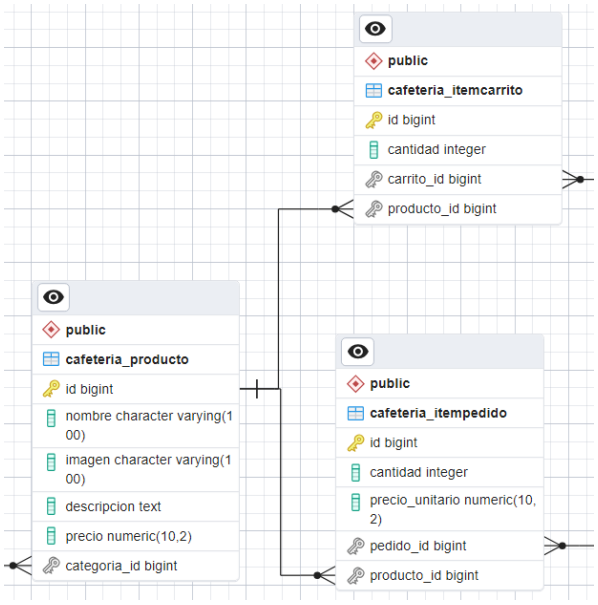

Figura 9: Figura de del modelo producto y sus relaciones

5.3. modelo cliente, carrito, pedido

El modelo cliente consta de campos como nombre, email y telefono, y tiene una relación de uno a muchos con carrito e edido (ambos). A su vez el modelo carrito tiene una relación de uno a muchos con el modelo itemcarrito, puede tene muchos itme carrito que su su vez guardan productos. Por otro lado, el modelo pedido también tiene una relación de uno a muchos con el modelo itempedido, y esta guarda a productos.

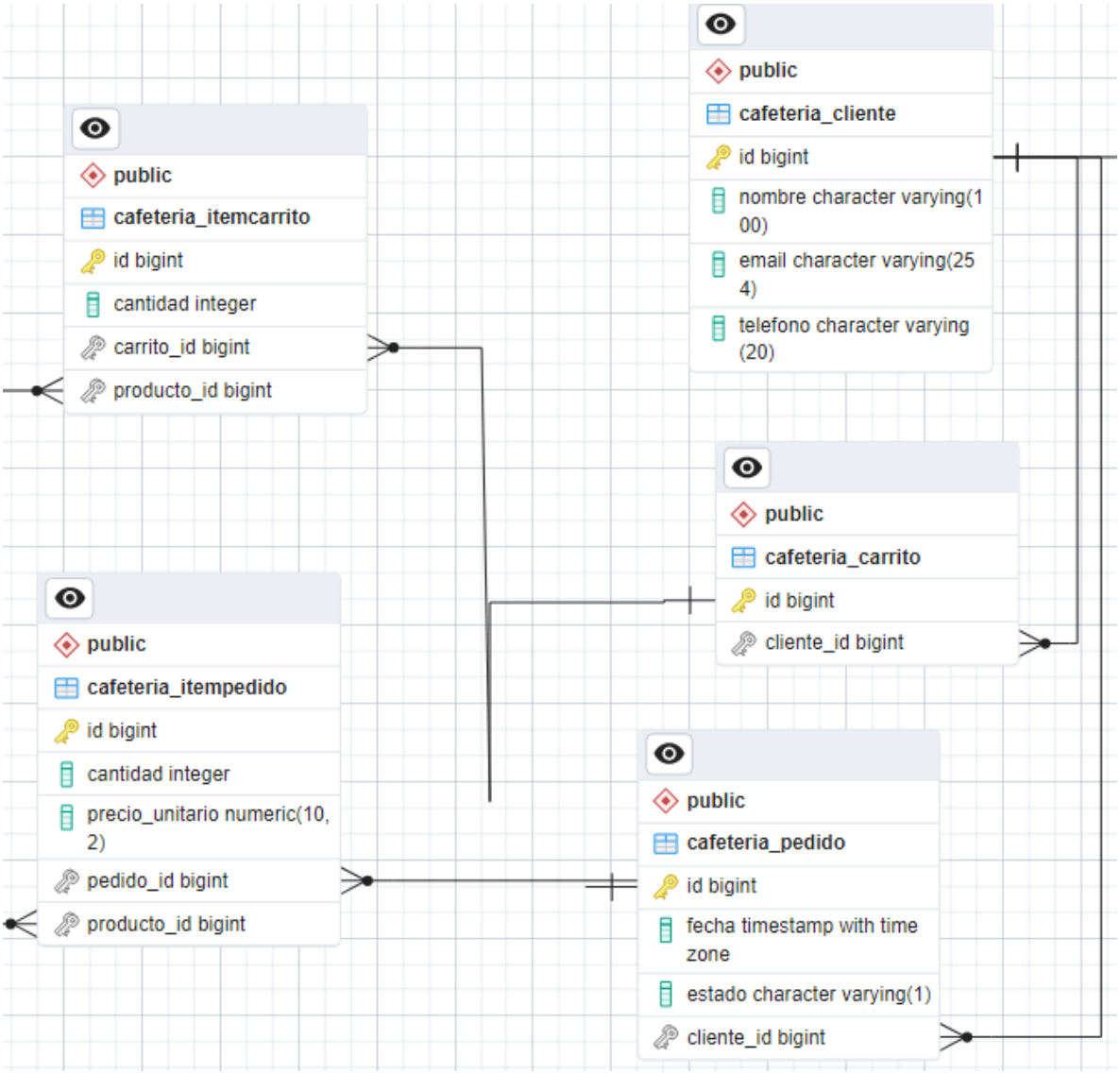

Figura 10: Figura de del modelo cliente, carrito y pedido con sus respactivas relaciones

5.4. modelo cliente, carrito, pedido

Antes de ver estos modelos, debemos tener en cuenta que estos modelos sirven como enlace a producto con carrito y pedido, esta relación inicialmente se consideraba de muchos a muchos, sin embargo, Django genera otra tabla por defecto con relación de uno a muchos. Estas tablas cumplen la misma funcionalidad y tienen algunos campos más.

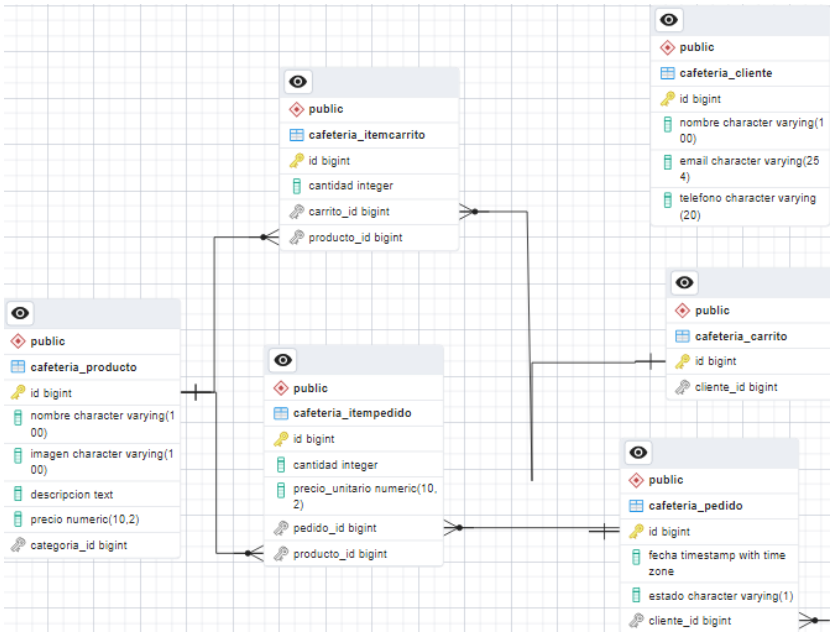

Figura 11: Figura de del modelo itemcarrito e itempedido con sus respactivas relaciones

5.5. modelo ERD

Ahora se mostrará la tabla con los modelos completos y sus respectivas relaciones.

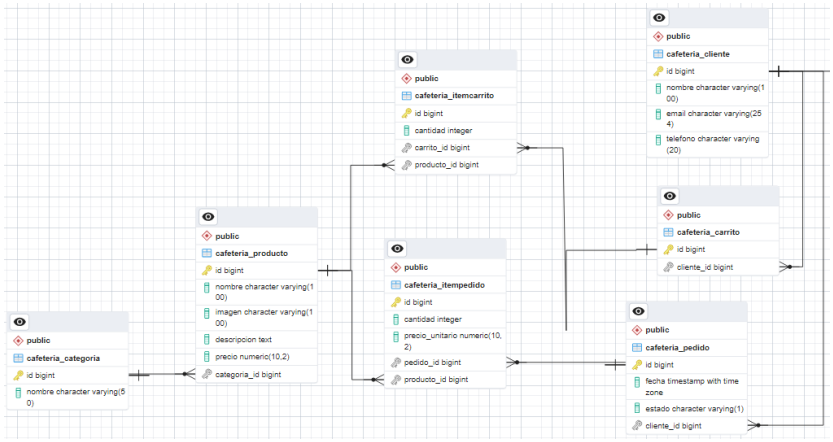

Figura 12: Se puede visualizar el Diagrama Entidad Relación completo, con los 7 modelos creados en código

6. Administración con Django

6.1. Setup

1. Se utilizo django como framework de trabajo
2. Se creo un entorno virtual "n"para no tener conflictos con otros proyectos
3. Se descargaron librerias de python con pip como:
   - pillow
   - psycopg2
   - sqlparse
4. se descargo postgres y pg admin como base de datos.
5. se creo el proyecto con "python startproject tienda"

7. Plantillas Bootstrap

- Se seleccionó la siguiente plantilla para el usuario final (No administrador).
  - Demo online: Yummy
  - URL: Link Aqui

  Se muestran las actividades realizadas para adecuación de plantillas, vistas, formularios en Django.
- Se adecuo a una carpeta static, todos aquellos elementos como su nombre lo indica, estaticos.
- Esto incluye imagenes, hojas de estilos y javascripts.

- Para usar los elementos de tipo dinamico se modifico usando diccionarios de contexto que permitan representar los elementos de la base de datos

```
{% for comids in comids %}
{% if comids.tipo == '1' %}
<!-- Bebidas -->
<div class="row gy-5">
  <div class="col-lg-4 menu-item">
   <img src="{{comids.img.url}}" class="menu-img img-fluid" alt="">
   <h4>{{comids.name}}</h4>
   <p class="ingredients">
     {{comids.desc}}
   </p>
   <p class="price">
     ${{comids.price}}
   </p>
  </div><!-- Menu Item -->
</div>
{% endif %}
{% endfor %}
```

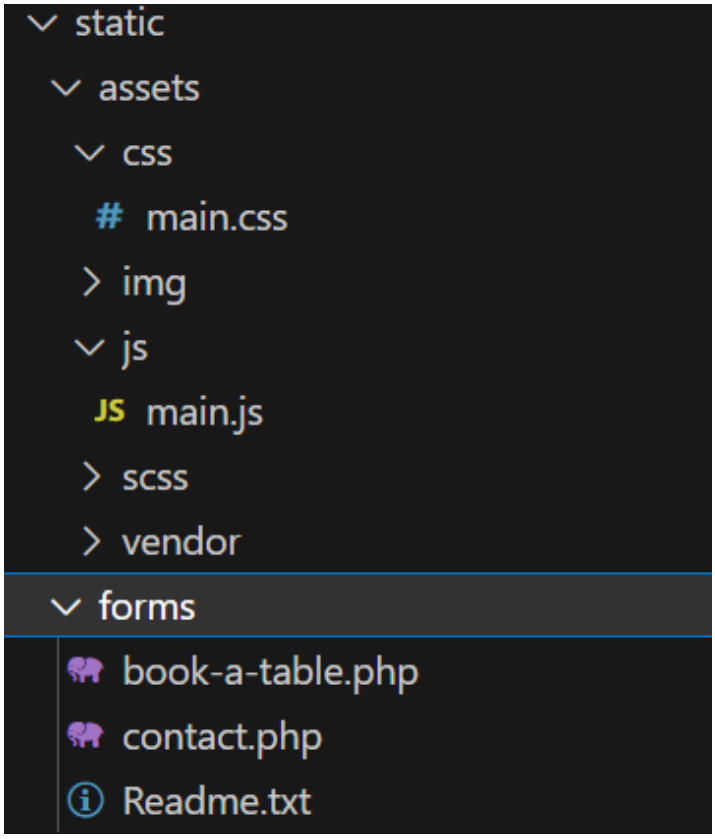

Figura 13: Partes staticas.

8. CRUD - Core Business - Clientes finales

- El núcleo de negocio del sistema de ventas para el restaurante radica en la facial vizualizacion de los contenidos y productos para el usuario, como el control y manejo para los miembros del staff:

  1. El staff inicia sesión.
  2. Si pertenece al grupo selecto es capaz de editar los productos.
  3. Se dirije a la pagina de agregar contenidos usando la interfaz de django.
  4. El miembro del staff vizualiza mediante las tablas, los productos previamente ingresados.
  5. El staff es es capaz de añadir, editar o eliminar los productos.
  6. El staff cierra sesión.

- Todas y cada una de estas pantallas debe funcionar en la plantilla bootstrap. A continuación se muestran las actividades realizadas para su construcción:

8.1. index.html

- El index.html es la primera impresión que los visitantes tienen de tu cafetería en línea. Proporciona la oportunidad de causar una impresión positiva al mostrar un diseño atractivo, imágenes de productos deliciosos y un diseño organizado que refleja la personalidad y el ambiente de tu cafetería.

- Navegación: El archivo index.html sirve como el centro de navegación de tu sitio. Mediante enlaces o secciones, los usuarios pueden acceder fácilmente a información esencial como los productos disponibles, la información de la cafetería, la ubicación y los precios. Esto facilita que los usuarios encuentren rápidamente lo que están buscando.
- Presentación de Productos: A través del index.html, puedes presentar tus productos de manera atractiva. Mediante imágenes de alta calidad y descripciones detalladas, puedes mostrar la variedad y calidad de los productos que ofreces, lo que puede atraer el interés y el apetito de los visitantes.

8.2. register.html

- La página register.html es un componente crucial en el sitio web de la cafetería, ya que permite a los visitantes registrarse y crear cuentas personalizadas
  - Acceso Personalizado: La página de registro proporciona a los usuarios un acceso personalizado al sitio web de la cafetería. Al crear cuentas individuales, los clientes pueden disfrutar de una experiencia más rica y específica que incluye funciones como realizar pedidos en línea y comunicarse con el personal del staff.
  - Recopilación de Información Esencial: Durante el proceso de registro, se recopila información clave, como nombres, direcciones de correo electrónico y nombres de usuario. Esta información permite al personal de la cafetería una forma responder a las peticiones y feedback con los clientes, brindar un servicio más orientado a sus necesidades.
  - Seguridad y Autenticación: La página register.html juega un papel fundamental en la seguridad y autenticación de los usuarios. Después de registrar una cuenta, los usuarios deben iniciar sesión con sus credenciales únicas, lo que asegura que solo las personas autorizadas tengan acceso a las funciones y datos relacionados con la cuenta.
  - Base de Datos de Clientes: La información recopilada durante el registro se almacena en la base de datos de PostgreSQL. Esto no solo permite la autenticación de usuarios, sino que también proporciona a la cafetería una valiosa base de datos de clientes para futuros análisis y estrategias de marketing.

9. Investigación: Email, Upload.

- Email: Se utilizará la funcionalidad del uso de envío de correos electrónicos para la parte de comunicación del usuario con el staff. Para ello se utilizo, un formulario para contactarse con los clientes y den su opinion acerca del negocio

```
<form action="forms/contact.php" method="post" role="form" class="php-email-form p-3
    p-md-4">
  <div class="row">
   <div class="col-xl-6 form-group">
     <input type="text" name="name" class="form-control" id="name" placeholder="Your Name"
         required>
   </div>
   <div class="col-xl-6 form-group">
     <input type="email" class="form-control" name="email" id="email" placeholder="Your
         Email" required>
   </div>
  </div>
  <div class="form-group">
```

```
 <input type="text" class="form-control" name="subject" id="subject"
     placeholder="Subject" required>
</div>
<div class="form-group">
 <textarea class="form-control" name="message" rows="5" placeholder="Message"
     required></textarea>
</div>
<div class="my-3">
 <div class="loading">Loading</div>
 <div class="error-message"></div>
 <div class="sent-message">Your message has been sent. Thank you!</div>
</div>
<div class="text-center"><button type="submit">Send Message</button></div>
```

10. REFERENCIAS

- https://www.w3schools.com/java/default.asp
- https://www.geeksforgeeks.org/insertion-sort/