Informe de Laboratorio 01
Tema: Git y GitHub

Nota

Estudiante	Escuela	Asignatura
Juan Perez Luna jperez@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Semestre: I Código: 20231001

Laboratorio	Tema	Duración
01	Git y GitHub	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 10 Abril 2023	Al 17 Abril 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 preparacion'),
         ('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. 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. ...

5. 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:

5.1. 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.

5.2. 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.

6. 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>
```

7. a

7.1. a.1

- Como es el primer laboratorio se creo el repositorio GitHub.
- Se realizaron los siguientes comandos en la computadora:

Listing 7: Creando directorio de trabajo

```
$ mkdir -p $HOME/rescobedoq/
```

Listing 8: Dirijiéndonos al directorio de trabajo

```
$ cd $HOME/rescobedoq/
```

Listing 9: Creando directorio para repositorio GitHub

```
$ mkdir -p $HOME/rescobedoq/programacion
```

Listing 10: Inicializando directorio para repositorio GitHub

```
$ cd $HOME/rescobedoq/programacion
$ echo "# programacion" >> README.md
$ git init
$ git config --global user.name "Richart Smith Escobedo Quispe"
$ git config --global user.email rescobedoq@gmail.com
$ git add README.md
$ git commit -m "first commit"
$ git branch -M main
$ git remote add origin https://github.com/rescobedoq/programacion.git
$ git push -u origin main
```

7.2. Commits

Listing 11: Primer Commit Creando carpeta/archivo para laboratorio 01

```
$ mkdir lab01
$ touch lab01/Insertion.java
$ git add .
$ git commit -m "Creando carpeta/archivo para laboratorio 01"
$ git push -u origin main
```

- Se creo el archivo **.gitignore** para no considerar los archivos ***.class** que son innecesarios hacer seguimiento.

Listing 12: Creando .gitignore

```
$ vim lab01/.gitignore
```

Listing 13: lab01/.gitignore

```
*.class
```

Listing 14: Commit: Creando .gitignore para archivos *.class

```
$ git add .
$ git commit -m "Creando .gitignore para archivos *.class"
$ git push -u origin main
```

- Para el siguiente commit se implemento el algoritmo de ordenamiento por Inserción, se imprime el arreglo caso definido en el mismo código.
- El pseudocódigo utilizado es el siguiente:

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Listing 15: Creando .gitignore

```
$ vim lab01/Insertion.java
```

Listing 16: Compilando y probando código

```
$ cd lab01
$ javac Insertion.java
$ java Insertion
5 2 4 6 1 3
5 5 4 6 1 3
2 5 5 6 1 3
2 4 5 6 1 3
2 2 4 5 6 3
1 2 4 4 5 6
```

Listing 17: Commit: Probando algoritmo de Inserción con arreglo

```
$ git add .
$ git commit -m "Probando algoritmo de Insercion con arreglo"
$ git push -u origin main
```

7.3. Estructura de laboratorio 01

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab01/
|--- Insertion.java
|--- latex
   |--- img
   |   |--- logo_abet.png
   |   |--- logo_episunsa.png
   |   |--- logo_unsa.jpg
   |   |--- pseudocodigo_insercion.png
   |--- programacion_lab01_rescobedoq_v1.0.pdf
   |--- programacion_lab01_rescobedoq_v1.0.tex
   |--- src
      |--- Insertion01.java
```

8. Pregunta: ¿Cúal es el comportamiento del algoritmo de ordenamiento por inserción?

- El algoritmo muestra un comportamiento cuadrático de $O(n^2)$.
- Se trabajarón los peores casos desde una arreglo de tamaño 1 hasta N.
- Para obtener un grafico ideal se utilizó N=10,000.

9. Rúbricas

9.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

9.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4			
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4			
Total		20		12	

10. Referencias

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