UTPL
La Universidad Católica de Loja

Vicerrectorado de Modalidad Abierta y a Distancia

Desarrollo Basado en Plataformas Web

Guía didáctica

UTPL

Facultad Ingenierías y Arquitectura

Desarrollo Basado en Plataformas Web

Guía didáctica

Carrera	PAO Nivel
Tecnologías de la información	VI

Autor:

René Rolando Elizalde Solano

DSOF_3048

Universidad Técnica Particular de Loja

Desarrollo Basado en Plataformas Web

Guía didáctica

René Rolando Elizalde Solano

Diagramación y diseño digital

Ediloja Cía. Ltda.
Marcelino Champagnat s/n y París
edilojacialtda@ediloja.com.ec
www.ediloja.com.ec

ISBN digital -978-9942-25-930-1

Año de edición: octubre, 2020

Edición: primera edición reestructurada en agosto 2025 (con un cambio del 50%)

Loja-Ecuador

Índice

1. Datos de información

1.1 Presentación de la asignatura

1.2 Competencias genéricas de la UTPL

- Comportamiento ético.

1.3 Competencias específicas de la carrera

- Implementar aplicaciones de baja, mediana y alta complejidad, integrando diferentes herramientas y plataformas para dar solución a requerimientos de la organización.

1.4 Problemática que aborda la asignatura

- Las temáticas sobre desarrollo de aplicaciones web tradicionales desde el ámbito del cliente y el servidor; servicios en la nube; manejo de ORM para acceso a base de datos relacionales y acceso a bases de datos NoSql; brindarán un sustento para entender la importancia del uso de tecnologías de ambiente web en el proyecto que involucra la creación de un marco para describir cómo una comunidad productiva puede organizarse con el apoyo de las Tecnologías de la Información para mejorar su producción y oportunidades de negocio.

2. Metodología de aprendizaje

Para el desarrollo de los contenidos en la asignatura, se usa la metodología de aprendizaje denominada ***Blended Learning***. A través de estos procedimientos, el proceso de aprendizaje se divide en trabajo autónomo y actividades síncronas y asíncronas. Se enfoca en brindar al estudiante los espacios para organizar su tiempo para cumplir con las actividades propuestas en cada una de las unidades de estudio de la asignatura, destacando los aspectos teóricos y prácticos. Además, permite al docente acompañar al estudiante a través de actividades síncronas permanentes para dar solución y aclarar dudas. Finalmente, los profesionales en formación reciben una atención personalizada en la adquisición de los resultados de aprendizaje propuestos.

Para revisar la información relacionada con la metodología descrita, usted puede revisar el siguiente artículo que describe la Metodología *blended learning*.

3. Orientaciones didácticas por resultados de aprendizaje

Primer bimestre

Resultado de aprendizaje 1:

Describe la diferencia entre SaaS y una aplicación Web tradicional.

El resultado de aprendizaje planteado permitirá el estudio de las características y tecnologías en el desarrollo de aplicaciones web tradicionales, junto a los conceptos y ámbitos de aplicación de servicios en la nube que involucran infraestructura, plataforma y software.

Contenidos, recursos y actividades de aprendizaje recomendadas

Recuerde revisar de manera paralela los contenidos con las actividades de aprendizaje recomendadas y actividades de aprendizaje evaluadas.

Semana 1

Unidad 1. Conceptos generales de desarrollo de plataformas *web*

Estimado estudiante, en el estudio de la asignatura de Desarrollo basado en plataformas *web*, usted podrá revisar un conjunto de conceptos, metodologías y tecnologías en el desarrollo de aplicaciones *web* desde el punto de vista del servidor y del cliente.

1.1. Características deseables de un sitio web

Para comprender las cualidades fundamentales que debe poseer un sitio *web*, es crucial analizar las expectativas de los usuarios y las organizaciones.

Las aplicaciones *web* deben incorporar un conjunto de características indispensables para asegurar su idoneidad y eficacia en función de las necesidades para las que fueron concebidas.

Estas propiedades de calidad del *software* son directamente aplicables al desarrollo de sitios *web* y se detallan a continuación:

Corrección y funcionalidad

El concepto de corrección se refiere a la capacidad de un sistema para ejecutar tareas o rutinas sin errores, de acuerdo con las especificaciones establecidas. La funcionalidad, por su parte, abarca el conjunto de posibilidades y operaciones que un sistema ofrece. Antes de iniciar el proceso de desarrollo, es fundamental plantearse interrogantes claves relacionadas con:

- La delimitación precisa del alcance del proyecto.
- La claridad y completitud de los requerimientos.
- Considerar los diversos casos de uso que puedan surgir a partir de los requerimientos.

Por ejemplo, un sistema de reservas de vuelos que permite a los usuarios buscar, seleccionar y pagar boletos sin errores en el cálculo de tarifas o en la confirmación de la reserva, y que además ofrece opciones para gestionar equipaje y asientos.

Robustez

Complementando la corrección, la robustez se define como la capacidad de las aplicaciones para operar y responder adecuadamente, incluso en escenarios inesperados o fuera de lo común. El objetivo es prevenir que la aplicación deje de funcionar debido a situaciones imprevistas. La importancia de la robustez en un sitio *web* es directamente proporcional a su nivel de criticidad; es decir, cuanto más crítica sea la aplicación, mayor será la relevancia de su robustez.

Por ejemplo, una plataforma de banca en línea que sigue funcionando y procesando transacciones de forma segura y precisa, incluso si hay un pico inesperado de usuarios simultáneos o fallas parciales en la red.

Facilidad de uso e imagen atractiva

Esta característica se centra en la simplicidad con la que personas de diversas formaciones y aptitudes pueden interactuar con la aplicación. Además, se debe considerar la complejidad de su instalación y puesta en marcha. Un diseño intuitivo y una interfaz visualmente atractiva son fundamentales para una experiencia de usuario óptima. Se relacionan con los conceptos de accesibilidad *web*.

Por ejemplo, un sitio *web* de comercio electrónico con una navegación clara, botones grandes y visibles, un proceso de compra con pocos pasos y un diseño visual moderno y agradable que invite al usuario a explorar los productos.

Portabilidad y compatibilidad

La portabilidad se refiere a la capacidad de una aplicación para ser instalada y ejecutada en diferentes entornos de *hardware* y *software*. La compatibilidad mide el grado de interoperabilidad con elementos de otros programas o sistemas. Es crucial establecer estas características como objetivos fundamentales desde las etapas iniciales del proyecto de desarrollo.

Por ejemplo, una aplicación *web* educativa que funciona correctamente en navegadores Chrome, Firefox y Safari, y que se adapta visualmente a pantallas de computadoras de escritorio, tabletas y teléfonos móviles sin perder funcionalidad.

Seguridad

La seguridad implica la implementación de procedimientos para prevenir accesos no autorizados, desde los niveles más básicos hasta los más complejos. La integridad de un sistema *web* es un aspecto crítico que debe ser

cuidadosamente considerado para evitar inconvenientes en la implementación y uso de la aplicación. Un ejemplo fundamental de técnicas de seguridad a aplicar es la prevención de inyecciones SQL maliciosas.

Por ejemplo, un portal de gestión de información personal que utiliza cifrado de extremo a extremo, autenticación de dos factores y filtros de entrada para prevenir ataques de inyección SQL, asegurando que solo los usuarios autorizados puedan acceder y modificar sus datos.

Facilidad de mantenimiento desde la visión del usuario

Durante la fase de operación de una aplicación, es común que surja la necesidad de incorporar nuevas funcionalidades. El sistema debe estar diseñado de manera que la implementación de estas mejoras sea lo más transparente posible tanto para los desarrolladores como para los usuarios.

Por ejemplo, un Sistema de gestión de Contenidos (CMS) donde un administrador puede añadir una nueva sección al sitio *web* (un *blog*) sin necesidad de modificar el código fuente, simplemente utilizando las herramientas de la interfaz administrativa.

Oportunidad y economía

La oportunidad se refiere a la capacidad de lanzar el sistema en la fecha establecida en el cronograma o incluso antes. En cuanto a la economía, implica que el desarrollo del sistema pueda completarse dentro del presupuesto asignado, optimizando los recursos disponibles.

Por ejemplo, un equipo de desarrollo que entrega una aplicación de gestión de inventario para una pequeña empresa en el plazo de tres meses acordado y dentro del presupuesto establecido, utilizando herramientas de código abierto y metodologías ágiles para optimizar los recursos (Gutiérrez González & López Goytia, 2016).

1.2. Tecnologías para la creación de sitios web

En la dinámica del entorno tecnológico actual, es fundamental que las organizaciones mantengan una posición competitiva, lo que implica comprender y aplicar las tecnologías más relevantes en el desarrollo *web*.

En el siguiente módulo didáctico, se detallan conceptos esenciales que sustentan la creación y funcionamiento de los sitios *web* modernos.

Conceptos fundamentales de la Arquitectura Web

Como pudo observar, comprender los fundamentos y tecnologías que conforman la arquitectura *web* es esencial para crear sitios *web* eficientes, accesibles y seguros.

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 2

Unidad 1. Conceptos generales de desarrollo de plataformas *web*

1.3. Servicios en la nube (*cloud*)

Estimado estudiante, continuamos con la planificación de la asignatura para abordar el tema relacionado con Servicios en la nube. Se recomienda realizar una lectura del capítulo I del libro DevOps y seguridad en cloud.

El *cloud computing*, o computación en la nube, representa un modelo de entrega de servicios tecnológicos que permite el acceso bajo demanda a un conjunto compartido y configurable de recursos informáticos (como redes, servidores, almacenamiento, aplicaciones y servicios) a través de *Internet*. Estos recursos se pueden aprovisionar y liberar rápidamente con un esfuerzo

mínimo de gestión con el proveedor de servicios. La transparencia en la disponibilidad y el acceso es una característica fundamental de este modelo. Se destacan las siguientes características adicionales:

- **Bajo demanda:** los usuarios pueden aprovisionar recursos informáticos, como memoria del servidor y almacenamiento en red, de forma unilateral y automática, sin requerir interacción humana con cada proveedor de servicios.
- **Acceso a la red:** las capacidades están disponibles a través de la red y se accede a ellas mediante diversos medios (por ejemplo, teléfonos móviles, tabletas, computadoras portátiles y estaciones de trabajo).
- **Agrupación de recursos:** los recursos informáticos del proveedor se agrupan para servir a múltiples consumidores utilizando un modelo de múltiples inquilinos, con diferentes recursos físicos y virtuales asignados y reasignados dinámicamente según la demanda del consumidor.
- **Servicio medido:** los sistemas de la nube controlan y optimizan el uso de los recursos aprovechando una capacidad de medición apropiada para el tipo de servicio (por ejemplo, almacenamiento, procesamiento, ancho de banda y cuentas de usuario activas). El uso de los recursos puede ser monitoreado, controlado y reportado, proporcionando transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

Además, es importante mencionar las principales formas de despliegue de los servicios en la nube, las cuales se exponen en el siguiente módulo didáctico:

Principales formas de despliegue de los servicios en la nube

1.4. Tipos de servicio en la nube

Los servicios de computación en la nube se clasifican por lo general en tres modelos principales, cada uno ofreciendo diferentes niveles de control y gestión por parte del usuario.

1.4.1. Infraestructura como servicio

IaaS proporciona las bases de construcción fundamentales de la computación en la nube. Permite a los usuarios alquilar recursos de infraestructura virtualizados, como máquinas virtuales, almacenamiento, redes y sistemas operativos, sin tener que gestionar el *hardware* físico formalmente. El usuario tiene control sobre el sistema operativo, las aplicaciones y los datos, mientras que el proveedor gestiona la infraestructura.

Entre sus principales características se tiene que: ofrece un alto grado de control y flexibilidad sobre los recursos informáticos; permite escalar recursos de forma rápida y eficiente; modelo de pago por uso, reduciendo la necesidad de grandes inversiones de capital, y, el proveedor es responsable del mantenimiento del *hardware*, la virtualización y la red.

Se detallan algunos ejemplos en plataformas de uso común:

- ***Amazon Web Services* (AWS) EC2:** un desarrollador puede lanzar máquinas virtuales (instancias EC2) para alojar un servidor web, una base de datos o un entorno de desarrollo personalizado. Paga solo por el tiempo en que las instancias están en ejecución y por el almacenamiento utilizado.
- ***Microsoft Azure Virtual Machines:*** una empresa puede migrar sus servidores locales a máquinas virtuales en Azure para reducir costos de *hardware* y mantenimiento, manteniendo el control sobre el *software* y las configuraciones del sistema operativo.
- ***Google Compute Engine:*** un equipo de ciencia de datos puede aprovisionar rápidamente *clusters* de máquinas virtuales de alto rendimiento para ejecutar análisis complejos o modelos de aprendizaje automático, escalando los recursos según la necesidad de cada proyecto.

1.4.2. Plataforma como servicio

PaaS ofrece un entorno completo para el desarrollo, ejecución y gestión de aplicaciones, sin la complejidad de construir y mantener la infraestructura. Incluye sistemas operativos, entornos de ejecución de lenguajes de

programación, bases de datos, servidores web y otras herramientas, permitiendo a los desarrolladores centrarse únicamente en el código y la lógica de sus aplicaciones.

Algunas características son: simplifica el desarrollo y despliegue de aplicaciones, ya que el proveedor gestiona la infraestructura; proporciona herramientas integradas para el ciclo de vida del desarrollo de *software*; permite una colaboración eficiente entre equipos de desarrollo, y, existe mayor abstracción de la infraestructura en comparación con IaaS.

Se detallan algunos ejemplos en plataformas de uso común:

- ***Google App Engine:*** un desarrollador puede desplegar una aplicación web Python o Node.js sin preocuparse por la gestión de servidores o bases de datos. App Engine escala automáticamente la aplicación para manejar el tráfico.
- **Heroku:** un equipo de desarrollo puede desplegar rápidamente su aplicación Ruby on Rails o Node.js en Heroku, aprovechando sus herramientas integradas para bases de datos (PostgreSQL), monitoreo y despliegue continuo, sin necesidad de configurar servidores.
- ***AWS Elastic Beanstalk:*** una empresa puede desplegar y escalar aplicaciones web desarrolladas en Java, .NET, PHP, Node.js, Python, Ruby, Go y Docker. Elastic Beanstalk maneja el aprovisionamiento de la capacidad, el balanceo de carga, el escalado automático y el monitoreo de la salud de la aplicación.
- ***Microsoft Azure App Service:*** un desarrollador puede construir, desplegar y escalar aplicaciones web y móviles en una plataforma completamente gestionada, integrándose fácilmente con otros servicios de Azure como bases de datos y funciones sin servidor.

1.4.3. Software como servicio

SaaS es un modelo de entrega de *software* donde las aplicaciones son alojadas por un proveedor de servicios y están disponibles para los usuarios a través de *Internet*, generalmente mediante un navegador *web*. Los usuarios no necesitan instalar, mantener y tampoco gestionar el software; simplemente acceden como un servicio normal.

Algunas características son: acceso basado en la *web*/móvil, disponible desde cualquier dispositivo con conexión a *Internet*; gestión y mantenimiento del *software* a cargo del proveedor; modelo de suscripción o pago por uso, eliminando la necesidad de licencias de *software* tradicionales; actualizaciones y parches automáticos.

En relación SaaS, algunos ejemplos de plataformas:

- **Gmail (*Google Workspace*):** un usuario accede a su correo electrónico y herramientas de productividad (Documentos, hojas de cálculo) a través de un navegador *web*, sin instalar ningún *software*. Google se encarga de los servidores, el mantenimiento y las actualizaciones.
- **Salesforce:** una empresa utiliza Salesforce como su sistema de gestión de relaciones con clientes (CRM) basado en la nube. Sus equipos de ventas y *marketing* acceden a la plataforma a través de un navegador para gestionar contactos, oportunidades y campañas, sin preocuparse por la infraestructura subyacente.
- **Microsoft 365:** los usuarios acceden a aplicaciones como Word, Excel y *PowerPoint*, así como a servicios de correo electrónico y colaboración (*SharePoint Online*), a través de la nube, pagando una suscripción mensual o anual.
- ***Zoom:*** una organización utiliza *Zoom* para videoconferencias y reuniones en línea. Los usuarios se conectan a través de la aplicación o el navegador, y *Zoom* gestiona toda la infraestructura de comunicación.

Para enriquecer su conocimiento, realice las actividades que se presentan a continuación:

Actividades de aprendizaje recomendadas

1. Realizar el siguiente crucigrama interactivo, donde se presentan conceptos clave de *cloud computing* y desarrollo *web*. Esta actividad le permitirá identificar el nivel de comprensión alcanzado sobre estas temáticas y los conceptos que debe volver a revisar para lograr un dominio completo. Además, con esa actividad, usted podrá reafirmar lo analizado y estudiado. Luego de intentar resolverlo, puede comprobar sus respuestas y plantearse preguntas similares.

   Juego de completar

2. Realizar la autoevaluación 1, donde se presentan un conjunto de preguntas en función de las temáticas descritas en la unidad 1: *Conceptos generales de desarrollo de plataformas web*. La autoevaluación le permitirá identificar el nivel de aprendizaje alcanzado y los ítems que se deben volver a revisar para lograr los resultados de aprendizaje propuestos. Además, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Autoevaluación 1

Instrucción: lea detenidamente cada uno de los enunciados de las preguntas y seleccione la o las respuestas correctas.

1. **Identifique una propiedad de calidad de software en el desarrollo de sitios web, de las siguientes expresiones propuestas:**

   a. *Framework*.
   b. Corrección y funcionalidad.
   c. HTML.

2. **Prevenir la inyección de SQL maliciosas, ¿a qué propiedad de *software* en el desarrollo de sitios *web* se refiere?**

a. Facilidad de mantenimiento.
b. Portabilidad.
c. Seguridad.

3. **¿A qué literal se refiere la siguiente idea? Una comunidad internacional que trabaja en la generación de estándares para la web a nivel global.**

a. HTML.
b. XHTML.
c. W3C.

4. **Identifique cuál de los siguientes protocolos es usado por los navegadores:**

a. SSH.
b. HTTPS.
c. UDP.

5. **¿Cuáles son las partes de una URL?**

a. Protocolo, recurso, ruta.
b. Protocolo, dominio, ruta.
c. Protocolo, dominio, HTTP.

6. **¿Cuál de las siguientes ideas es válida para describir los servicios en la nube?**

a. Busca que los servicios estén a disposición de usuarios solo en procesos críticos.
b. Modelo que permite acceder a entornos locales como: almacenamiento en servidores y acceso a redes de información.
c. Modelo que permite acceder a diversos servicios como: almacenamiento en servidores y acceso a redes de información.

7. **Identifique el concepto correcto para *HyperText Markup Language*.**

   a. Es un lenguaje de programación funcional que permite la generación de páginas.
   b. Es un lenguaje de orientación a objetos que permite la generación de páginas.
   c. Es un lenguaje de etiquetado de documentos que permite la generación de páginas.

8. **Identifique dos complementos del lenguaje XML.**

   a. XSL, XPath.
   b. HTML5, Ajax.
   c. JavaScript, DHTML.

9. **Alojar una página *web* en un servidor. ¿Con qué nombre se conoce?**

   a. Dirección IP.
   b. Hosting.
   c. Dominio.

10. **¿A qué concepto hace referencia lo siguiente? Un equipo con características particulares en lo relacionado con el almacenamiento y procesamiento de información.**

    a. Cliente.
    b. *Hosting*.
    c. Servidor.

Ir al solucionario

¡Excelente! Usted ha finalizado el estudio de la primera unidad. Es importante recordarle que, si necesita afianzar los conceptos que pudieran resultar más difíciles, puede revisar los apartados correspondientes. Avancemos a la siguiente unidad.

Resultado de aprendizaje 2:

Discute el impacto de la implementación de estándares y/o atributos de calidad en aplicaciones web

Las temáticas abordadas a través del recurso de aprendizaje descrito son relacionadas al entendimiento conceptual y práctico de programación de lado del cliente usando HTML, CSS y JavaScript y el estudio de conceptos de ORM para la interacción con base de datos relacionales.

Contenidos, recursos y actividades de aprendizaje recomendadas

Recuerde revisar de manera paralela los contenidos con las actividades de aprendizaje recomendadas y actividades de aprendizaje evaluadas.

Semana 3

Unidad 2. Programación del lado del cliente

Las herramientas como HTML, CSS y JavaScript facilitan la programación del lado del cliente, es decir, sus procesos se ejecutan en el navegador del usuario. De la aplicación *backend* se hablará en las próximas unidades.

En la unidad número 2 de la presente guía se estudiarán Conceptos y ejemplos sobre la programación del lado del cliente

2.1. Introducción

En el desarrollo de aplicaciones web, es fundamental dominar las tecnologías del lado del cliente como HTML, CSS y JavaScript. HTML proporciona la estructura semántica y el contenido fundamental de la página, actuando como el esqueleto sobre el cual se construye toda la experiencia. CSS es indispensable para definir la presentación, el estilo visual y el diseño responsivo de la interfaz, asegurando que la aplicación sea estéticamente atractiva y se adapte a diferentes dispositivos y tamaños de pantalla.

Finalmente, JavaScript añade interactividad, dinamismo y lógica de negocio en el navegador, permitiendo la manipulación del *Document Object Model* (DOM), el manejo de eventos de usuario, y la realización de peticiones asíncronas (AJAX) para crear experiencias de usuario ricas, fluidas y altamente responsivas.

La combinación de estas tres tecnologías es la base ineludible para construir cualquier Interfaz de Usuario moderna y eficiente, independientemente del modelo de servicio en la nube o la arquitectura de *backend* utilizada.

En el transcurso de los próximos años, los estándares evolucionarán. La sintaxis puede variar, pero la responsabilidad como profesionales de las Tecnologías de la Información es saber utilizar las herramientas en diferentes contextos o problemáticas. Se resalta la importancia de dejar de lado el diseño de plano de pantallas y siempre asociar el diseño a las funcionalidades que se desea construir; en el marco de desarrollo de aplicaciones *web*. En cuanto al funcionamiento obvio y adaptable, se indica la necesidad obligatoria de crear aplicaciones *web* intuitivas, evitando distracciones innecesarias por parte de los usuarios al momento de usarlas; sobresalen las siguientes recomendaciones:

- Generar pruebas con usuarios reales.
- Los sitios deben ser muy fáciles de usar.
- Estructuras de las páginas fáciles de identificar.
- Bajar el ruido visual.

Además, para la generación de elementos HTML en el ámbito de diseño *web* adaptativo, se sugieren dos características:

- Los elementos HTML se deben expresar en porcentajes, no píxeles.
- Cuando sea indispensable, se deben generar dimensiones mínimas en los elementos (Gutiérrez González & López Goytia, 2016).

2.2. Ejemplos de HTML y CSS

Para entender este punto de la guía didáctica se solicita a usted, estimado estudiante, realizar una lectura de la unidad 2: Estructura de una página *web*, del libro HTML y CSS como nunca antes se lo habían contado.

a. **Ejemplo 1**

Para el primer ejemplo básico de HTML se usará el siguiente código:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
2   <html>
3   <head>
4   <meta http-equiv="content-type" content="text/html;
charset=utf-8" />
5   <title>El País Ecuador</title>
6   </head>
7   <body>
8   <h1>Ecuador</h1>
9   <h2>
10  Oficialmente República del Ecuador, es un país soberano ubicado
en la región noroccidental de América del Sur, compuesto por
veinticuatro provincias. Limita al norte con Colombia, al sur y al
este con Perú y al oeste con el océano Pacífico, el cual lo separa
de las islas Galápagos por mil kilómetros entre la península de
Santa Elena y la isla San Cristóbal.
11  </h2>
12
13  <h3>Tomado de <a href="https://es.wikipedia.org/wiki/
Ecuador">wikipedia</ a></h3>
14  </body>
15  </html>
```

Usted debe realizar los siguientes pasos para visualizar el HTML propuesto en un navegador en su computador:

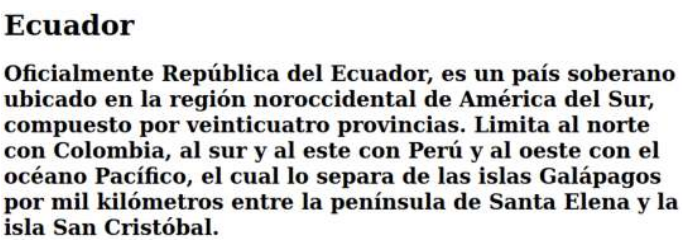
- Copiar el código en un editor de texto: *bloc* de notas, Sublime Text, etc.
- Guardar el archivo con la siguiente estructura:
  - [nombre del archivo].html

- Para visualizar el contenido:
  - **Opción 1:** hacer doble *clic* en el archivo HTML y se abrirá el navegador que tenga configurado por defecto.
  - **Opción 2:** abrir el navegador que usted desee (Chrome, Mozilla Firefox), ir al menú *archivo, opción, abrir archivo* y buscar el archivo en el directorio donde está el archivo HTML.

Luego de realizar los pasos descritos, se visualiza la figura 1.

Figura 1

Ejemplo de HTML básico



Nota. Elizalde, R., 2025.

Explicación del ejemplo:

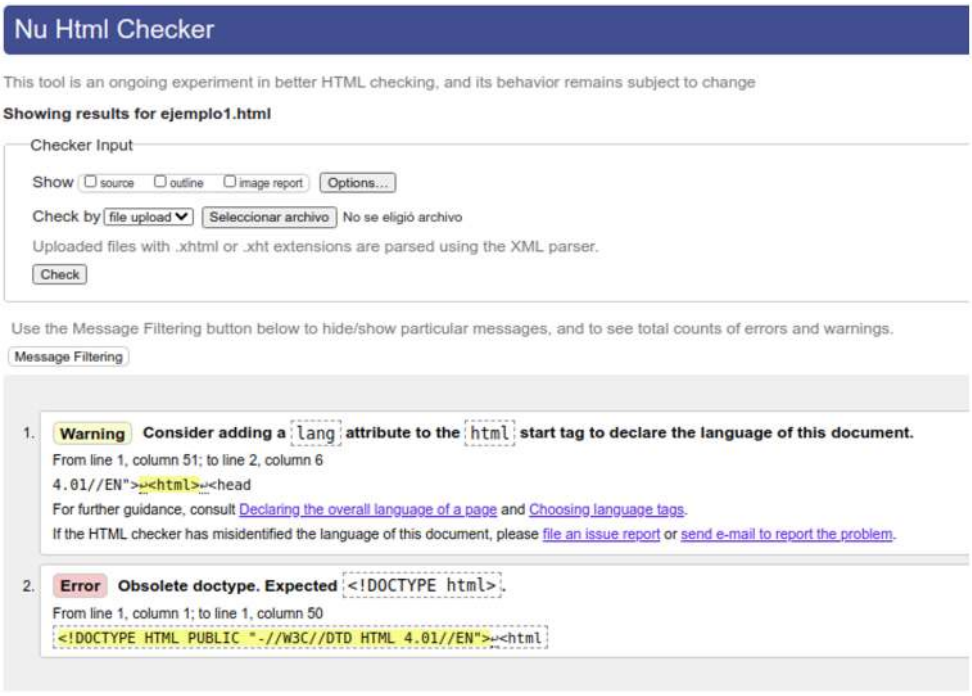
- La línea 1 indica en qué versión de HTML serán interpretadas las líneas expuestas en el archivo.
- Línea 2 da inicio al documento a través de la etiqueta **html**.
- Línea 3 se inicia el encabezado de la página.
- La línea 4 permite especificar el estándar UTF-8 para representar caracteres en cualquier navegador.
- Línea 7. desde esta etiqueta se ubican las etiquetas que deseamos incorporar a la página web.
- Tomar en consideración que cada etiqueta usada en la página ha sido cerrada.

Se recomienda revisar que los contenidos generados en la aplicación web estén bajo las recomendaciones de la W3C; para ello se debe usar la herramienta Markup Validation Service. Para el ejemplo expuesto se tiene la siguiente salida luego de realizar la comprobación expuesta en la Figura 2. Según los resultados, existen algunas consideraciones:

- La primera salida indica una advertencia; se sugiere según el estándar de la W3C, agregar el atributo lang a la etiqueta html.
- La segunda salida es un error. La versión de HTML está obsoleta; es correcto pues el estándar sugiere el uso de <!DOCTYPE html>. Se manifiesta que a pesar de que el estándar del ejemplo está en desuso es muy importante que los estudiantes en formación se familiaricen con encabezados que se siguen usando en algunas aplicaciones en el mercado.

Figura 2

Uso de validador de contenido HTML bajo las consideraciones de la W3C



Nota. Elizalde, R., 2025.

En los siguientes ejemplos se hace uso del estándar HTML5 y CSS3. Las principales diferencias en el uso del estándar HTML5 se listan a continuación:

- El encabezado tiene la siguiente estructura: <!DOCTYPE html> .
- Se usa como recomendación el grupo de caracteres UTF-8.
- Los estilos se los administra desde las hojas de estilo CSS.

Resaltar la importancia de las hojas de estilo y su relevancia en el desarrollo de aplicaciones web; la modularidad y mantenimiento de código, los cambios a futuro deben ser fáciles de realizar por los desarrolladores. Las hojas de estilo permiten tener una uniformidad en cuanto a formato y presentación en una aplicación web.

Puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 2.2.1

b. **Ejemplo 2**

Para el siguiente ejemplo se usan dos archivos:

- El archivo HTML, que tiene el siguiente código:

```
<!DOCTYPE html>
<html>
<head>
<meta charset=”utf-8” />
<meta name=”viewport” content=”width=device-width” />
<link rel=”stylesheet” href=”estilos.css” type=”text/css” />
<title>Ejemplo HTML5 y CSS3</title>
</head>
<body>
<h2>Provincias del Ecuador</h2>
<table>
<tr>
<td class=”titulodecampo”>Nombre</td>
<td class=”titulodecampo”>Capital</td>
</tr>
<tr>
```

```
17  <td>Loja</td>
18  <td>Loja</td>
19  </tr>
20  <tr>
21  <td>Pichincha</td>
22  <td>Quito</td>
23  </tr>
24
25  </table>
26  </body>
27  </html>
```

- El archivo CSS, con el siguiente contenido:

```
1 /* manejo de las características de la etiqueta body*/
2
3   body {
4   background: #FEFEFD;
5   padding: 20px;
6
7 }
8
9   /* manejo de las características de la etiqueta table*/
10  th, td {
11  padding: 15px;
12  border-bottom: 1px solid black;
13  }
14
15 tr:hover {
16 background-color: #258EAD;
17 }
18
19 /* manejo de las características de las clases*/
20
21  titulodecampo {
22  background: #9999FF;
23  color: #000000;
24  font-family: Arial, Helvetica, sans-serif;
25  font-size: 14px;
```

```
26   font-weight: bold;
27
28 }
```

Si usted ejecuta en su computador personal los archivos anteriores, se tiene la salida de la Figura 3.

Figura 3

Visualización del ejemplo 2, usando el estándar HTML5 y CCS3

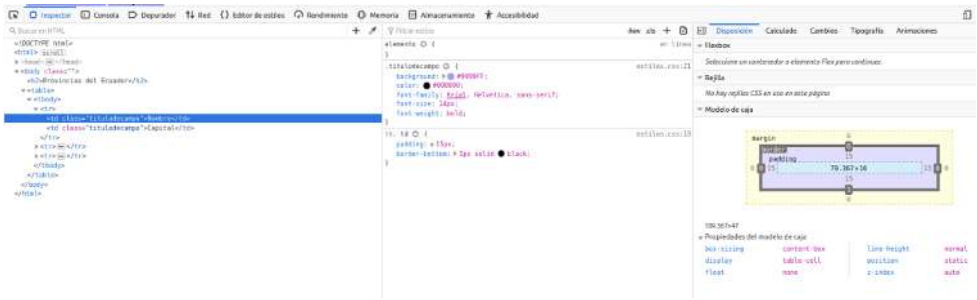
Nota. Elizalde, R., 2025.

Los ejemplos anteriores permiten recordar conceptos ya estudiados en asignaturas anteriores. En el desarrollo de las temáticas se muestran algunas recomendaciones en lo relacionado a librerías, extensiones y páginas en general que apoyan el desarrollo de aplicaciones web.

- En los navegadores Mozilla, Firefox y Chrome se puede utilizar dentro de las herramientas de Desarrollo Web las opciones de inspector, consola, editor de estilos, etc con el objetivo de poder realizar acciones como: realizar cambios en nuestra página sin necesidad de acudir a los archivos fuentes, los cambios no serán persistentes. Revise la Figura 4:

Figura 4

Herramientas de Desarrollo Web de Mozilla Firefox



Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 2.2.2

Ahora, continuemos con el estudio del siguiente tema que trata sobre la creación de interfaces dinámicas y la interacción con el usuario mediante JavaScript y jQuery, herramientas esenciales para desarrollar páginas web modernas y funcionales.

JavaScript y jQuery Interactivo

2.3. Manejo de JavaScript

A continuación, se presentan ejemplos con el lenguaje de programación interpretado JavaScript. Recuerde que es el lenguaje que brinda dinamismo e interactividad a las páginas web. Comprender sus bases (variables, condicionales, ciclos repetitivos, entrada de datos) es esencial para manipular

el contenido de la página, responder a las acciones del usuario y crear experiencias web dinámicas. Se busca tener una experiencia en la manipulación del DOM, manejo de eventos y peticiones asíncronas (AJAX).

a. **Ejemplo 1**

- Las líneas de código propuestas permiten ingresar por teclado dos valores; se procede a realizar el cálculo de la suma y diferencia de los mismos; finalmente se presenta en pantalla los valores resultantes.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width" />
<title>Demo JavaScript</title>
</head>
<body>
<h1>Suma de dos números</h1>
<script type="text/javascript" charset="utf-8">
var entrada1 = window.prompt("Escriba el primer valor: ", "0");
var entrada2 = window.prompt("Escriba el segundo valor: ",
"0");
....
var a = parseFloat(entrada1);
var b = parseFloat(entrada2);
var c = a + b;
var d = a - b;
alert("La suma de los valores es: " + c);
alert("La diferencia de los valores es: " + d);
</script>
</body>
</html>
```

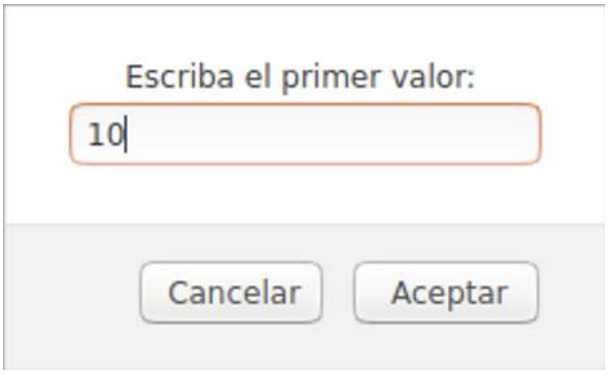
Luego de ejecutar en un navegador, usted puede confirmar que el ejemplo cumple con el sencillo objetivo propuesto.

Explicación:

- En la línea 10 la etiqueta **script** permite indicar que se va a insertar un script ejecutable dentro de la página HTML.
- La etiqueta script tiene una propiedad **type**, cuyo valor determina el lenguaje usado. Para el ejemplo se denota *text/* javascript, que determina el lenguaje JavaScript.
- En la línea 11 y 12 se crean variables que van a almacenar el valor que se ingrese por pantalla, a través del diálogo indicado por el método **window.prompt**; dicha propiedad acepta una cadena que será el mensaje que se presenta al usuario, además se determina el valor por defecto, para el ejemplo es cero (0). Ver Figura 5:

Figura 5

Diálogo mostrado por el método window.prompt

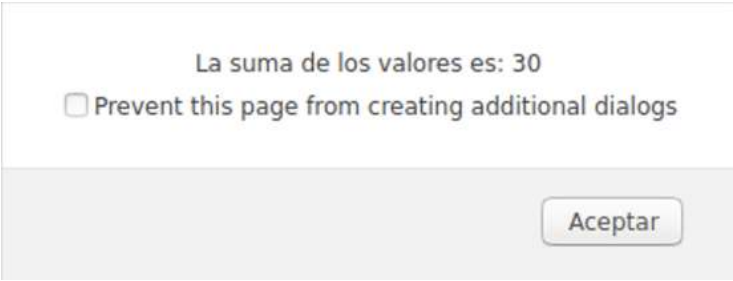


Nota. Elizalde, R., 2025.

- En las líneas 14 y 15 se transforma el valor ingresado por teclado; es una cadena, en un valor numérico de tipo float, a través de la palabra reservada **parseFloat**.
- En las líneas 16 y 17 se realizan las operaciones indicadas en la problemática.
- En las sentencias de las líneas 18 y 19 se usa el método **alert** para mostrar un diálogo para presentar un texto y los valores de las operaciones del punto anterior, ver Figura 6.

Figura 6

Usa el método alert para mostrar un diálogo para presentar un texto



Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: [Desarrollo del ejemplo 2.3_1]

b. **Ejemplo 2**

- El siguiente ejemplo, sigue la misma dinámica del ejemplo anterior; la diferencia radica en la forma de presentar los valores resultantes de las operaciones en la misma página. Se deja de usar el método **alert**.

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width" />
6   <title>Demo JavaScript</title>
7   </head>
8   <body>
9   <h1>Suma de dos números</h1>
10  <p id="resultadoSuma"></p>
11  <p id="resultadoResta"></p>
12
13  <script type="text/javascript" charset="utf-8">
14  var entrada1 = window.prompt("Escriba el primer valor: ", "0");
15  var entrada2 = window.prompt("Escriba el segundo valor: ",
"0");
16
```

```
17  var a = parseFloat(entrada1);
18  var b = parseFloat(entrada2);
19  var c = a + b;
20  var d = a - b;
21  // alert("La suma de los valores es: " + c);
22  // alert("La diferencia de los valores es: " + d);
23  document.getElementById("resultadoSuma").innerHTML = "<b>La
suma de valores\
24  es: " + c + "</b>";
25  document.getElementById("resultadoResta").innerText = "<b>La
diferencia de valores\
26  es: " + d + "</b>";
27  </script>
28  </body>
29  </html>
```

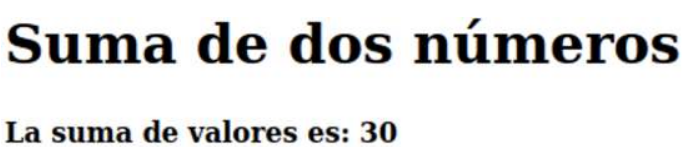
Explicación:

- En las líneas 23 y 25 se expresan las diferencias en relación al ejemplo anterior. La palabra reservada **document** permite acceder a cualquier componente de la página web; y getElementById retorna el elemento o etiqueta que tenga el id pasado como parámetro, para el ejemplo es resultadoSuma y resultadoResta, respectivamente.
- Con el elemento obtenido en la línea 23 se usa la propiedad innerHTML para agregar un HTML al elemento; para el ejemplo se establece una etiqueta <b> con formato de letra negrita o bold.
- Con el elemento obtenido en la línea 25 se usa la propiedad innerText para agregar un texto al elemento; a pesar que se está agregando una expresión que se podría visualizar como un HTML, solo será visible como una cadena, en base a la propiedad innerText.

El resultado final del ejemplo anterior se puede observar en la Figura 7.

Figura 7

Resultado final del ejemplo 2 de lenguaje JavaScript



Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: [Desarrollo del ejemplo 2.3_2]

2.3.1. Librería jQuery

En el presente apartado se estudiará la librería de JavaScript denominada jQuery; que posee características descritas por Pardo et al., (2011) como:

- Mejorar el manejo del DOM
- Manejo de eventos
- Manejo de animación
- Mejora y simplicidad en el uso de la técnica de desarrollo de ambiente web Asynchronous JavaScript And XML (AJAX) para crear aplicaciones interactivas.

Se explicará su funcionamiento a través de algunos ejemplos:

a. **Ejemplo 1**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width" />
<title>Demo Jquery</title>
<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
</head>
```

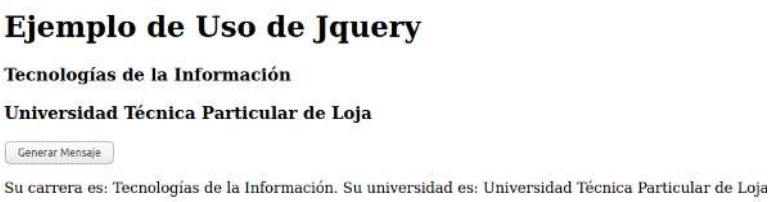
```
9   <body>
10  <h1>Ejemplo de Uso de Jquery</h1>
11  <h3 id=”nombreCarrera”>Tecnologías de la Información</h3>
12  <h3 id=”nombreUniversidad”>Universidad Técnica Particular de
Loja</h3>
13  <button type=”submit”>Generar Mensaje</button>
14  <p id=”mensajeFinal”>
15  </p>
16  </body>
17  <script>
18  $(document).ready(function() {
19  $( “button” ).click(function() {
20  var texto1 = $(‘#nombreCarrera’).text();
21  var texto2 = $(‘#nombreUniversidad’).text();
22  $(‘#mensajeFinal’).text(“Su carrera es: “+ texto1 + “. Su
universidad es: “
23  + texto2);
24  console.log(texto1);
25  });
26  });
27  </script>
28  </html>
```

Explicación:

- En el ejemplo se toma el texto de las etiquetas nombreCarrera y nombreUniversidad y se lo asigna a la etiqueta mensajeFinal, teniendo como salida lo expuesto en la Figura 8.

Figura 8

Salida del ejemplo de uso de jQuery.



Nota. Elizalde, R., 2025.

- Para poder usar las características de jQuery en la página web se debe agregar su referencia. Existen dos formas:
  - En la dirección web "Jquery/Download" se busca el archivo js de la librería y se guarda en la carpeta en la que se está trabajando el proyecto web.
  - Usar la versión de las librerías en línea, a la que se accede a través de la Red de Distribución de Contenido (Content Delivery Network – CDN). En el ejemplo se usa esta opción. En la línea 7 se especifica lo indicado.

- Para acceder a las propiedades de la librería jQuery se usa el carácter **$**; además los procedimientos deben estar entre las etiquetas <script>. En la línea 19 se indica que cuando se haga un clic en cualquier etiqueta **button** de la página web, se realizará una acción. En la línea 20 se está asignando un valor en la variable texto2, el texto que tenga la etiqueta que tiene como atributo id el valor nombreCarrera, a través de la propiedad text().
- En la línea 22 se accede a la etiqueta que tiene el **id** mensajeFinal y a través del método text se le asigna una cadena de texto.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 2.3.1_1

La librería jQuery tiene algunos proyectos relacionados:

- jQuery Mobile
- jQuery User Interface
- QUnit - JavaScript Unit Testing framework

En relación a jQuery User Interface permite agregar plugins, interacciones y widgets de manera sencilla, permitiendo generar páginas interactivas en un alto nivel (jQuery UI, 2020)

b. **Ejemplo 2**

- Se realiza el mismo proceso del Ejemplo 1, pero se hace uso de uno de los componentes de jQueryUI denominado Accordion, que permite ver el contenido en paneles en función de las necesidades que el usuario lo requiera.

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width" />
6   <title>Demo Jquery y JqueryUI</title>
7   <link rel="stylesheet" href="https://code.jquery.com/ui/1.12.1/
themes/base/ jquery-ui.css">
8   <script src="https://code.jquery.com/jquery-3.5.0.js"></script>
9   <script  src="https://code.jquery.com/ui/1.12.1/jquery-
ui.js"></script>
10  </head>
11  <body>
12  <h2>Ejemplo de Uso de jQuery y jQueryUI</h2>
13  <div id="accordion">
14  <h3>Sección 1</h3>
15  <div>
16  <p id="nombreCarrera">Tecnologías de la Información</p>
17  <p id="nombreUniversidad">Universidad Técnica Particular de
Loja</p>
18  <button type="submit">Generar Mensaje</button>
19  </div>
20  <h3>Sección 2</h3>
```

```
21  <div>
22  <p id="mensajeFinal"></p>
23  </div>
24  </div>
25  </body>
26  <script>
27  $(document).ready(function() {
28  $("#accordion").accordion();
29  $( "button" ).click(function() {
30  var texto1 = $('#nombre Carrera').text();
31  var texto2 = $('#nombreUniversidad').text();
32  $('#mensajeFinal').text("Su carrera es: "+ texto1 + ". Su
universidad es "
33  + texto2);
34  console.log(texto1);
35  });
36  });
37 </script>
38 </html>
```

Este código HTML muestra cómo implementar el componente Accordion de jQueryUI, permitiendo organizar el contenido en secciones plegables. Esto se observa en la Figura 9.

Figura 9

Ejemplo de uso de jQuery y jQueryUI

Nota. Elizalde, R., 2025.

Explicación:

- Además de hacer uso de las librerías jQuery a través de su versión en línea vía CDN, se debe agregar dos referencias más para usar las características descritas para jQueryUI. La línea 7 hace referencia a las hojas de estilo y la línea 9 permite referenciar al archivo de JavaScript de jQueryUI.
- Existe una estructura sugerida desde la línea 13 hasta la línea 24.
- En la línea 13 se da inicio a una etiqueta **div** con atributo **id** accordion.
- La estructura tiene la siguiente secuencia:
  - Una etiqueta **h3** y una etiquea **div**
  - La secuencia se repite dos veces; en el ejemplo hay dos paneles.
  - Importante, dentro de cada div en la secuencia se puede ubicar las etiquetas que se requieran sin importar el orden.
- Para poder visualizar los paneles, bajo las características de jQuery, en la etiqueta **script** se debe acceder al componte de la línea 13 y agregarle la funcionalidad accordion(); lo mencionado se realiza en la línea 28.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 2.3.1_2

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 4

Unidad 2. Programación del lado del cliente

2.4. Base de datos NoSql

Casas, Nin y Julbe (2019), explican que el concepto NoSQL abarca diversas soluciones para el almacenamiento de distintos tipos de datos, desde tablas y grafos hasta documentos e imágenes, o cualquier otro formato. Por su propia

definición, toda base de datos NoSQL es distribuida y escalable. En el mercado, existen numerosos productos, muchos de ellos de código abierto, como Cassandra, que opera almacenando la información en columnas en lugar de filas y generando índices asociativos para una recuperación rápida de grandes bloques de datos.

Asimismo, Casas, Nin y Julbe (2019), aclaran que las bases de datos NoSQL no buscan reemplazar a las relacionales, sino que ofrecen soluciones alternativas que mejoran el rendimiento de los sistemas de gestión de bases de datos para problemas y aplicaciones específicas. Por esta razón, el término NoSQL se asocia con la idea de "*not only* SQL" (no solo SQL), lo que significa que no prohíbe el uso del lenguaje estructurado de consultas, sino que lo complementa. En la obra de Herrera et al., (2016), los autores manifiestan ideas sobre el surgimiento de NoSql (*not only* SQL), concepto que nace a raíz de los inconvenientes que tiene con la cantidad de información y el no acoplamiento de la misma a esquemas estrictos relacionales.

De acuerdo con Robles et al., (2015), NoSql es una arquitectura en el ámbito de base de datos que no necesita esquemas a nivel de entidades o tablas fijas. Algunas características de este tipo de base de datos son: escalado horizontal, baja de costos de operación y mantenimiento. Algunos modelos de datos NoSql más comunes se listan a continuación:

- **Modelo llave-valor**, donde a cada llave se le asigna un dato, genera procesamiento de información de manera ágil.
- **Modelo orientado a columnas**, se usa el concepto de tabla, pero no se usa relaciones; cada dato es almacenado en columnas.
- **Modelo de documentos**, se caracteriza por usar formatos como: XML y JSON para el almacenamiento de la información.
- **Modelo orientado a grafos**, realiza una administración de datos provenientes de redes sociales principalmente. La información es dividida en fracciones más pequeñas o básicas y guardando relación entre los datos (Herrera et al., 2016).

En la siguiente tabla, según la clasificación propuesta por Robles et al., (2015), se presentan los principales modelos de bases de datos NoSQL junto con algunos de sus sistemas representativos.

Tabla 1
Clasificación de sistemas NoSql

Modelo NoSql	Ejemplos de sistemas NoSql
Llave – valor	• Redis • Azure Table Storage • Couchbase • Amazon Dynamo
Orientado a columnas	• Hadoop • Hbase • Cassandra
Basado en documentos	• CouchDB • MongoDB
Basado en grafos	• OrientDB • Neo4J

Nota. Adaptado de *¿Qué características tienen los esquemas NoSQL?*, por Robles, D., Sánchez, M., Serrano, R., Adárraga, B., y Heredia Vizca, D., 2015, Revista Investigación y desarrollo en TIC (IDENTIC).

Luego de revisar la Tabla 1 usted puede identificar los principales modelos de sistemas NoSql y ejemplificaciones de cada uno de ellos.

2.4.1. Manipulación de datos con CouchDB

Para la ejemplificación en el uso de bases de datos NoSQL se usará una base de datos de modelo basado en documentos llamada CouchDB. Una de las principales ventajas para proponer su uso es la interfaz gráfica a nivel web que posee para la administración de información.

En el documento de Robles et al. (2015) se indican algunas consideraciones de CouchDB:

- Para el almacenamiento de datos se usa el formato JSON.
- Se pueden realizar consultas a través del uso de Map/Reduce y el API de la base de datos mediante el protocolo HTTP.

Las ventajas de usar CouchDB radican en la adaptabilidad en aplicaciones comunes; un motor de búsqueda muy robusto; y la escalabilidad.

a. **Instalación de CouchDB**

La base de datos CouchDB puede ser instalada en sistemas operativos Windows, GNU/Linux y MacOS.

El proceso de instalación para sistema operativo Windows se lo puede realizar a través del siguiente enlace web [instalación de couchDB en Windows]; en el mismo se destacan algunos puntos:

- Se requiere tener instalado .NET Framework v3.5
- Muy importante, la ruta de instalación de CouchDB debe ser sin espacios. Ejemplo: C:\CouchDB
- En los ejercicios de la presente guía se usará la instalación en modo “single node”.

b. **Interfaz Fauxton**

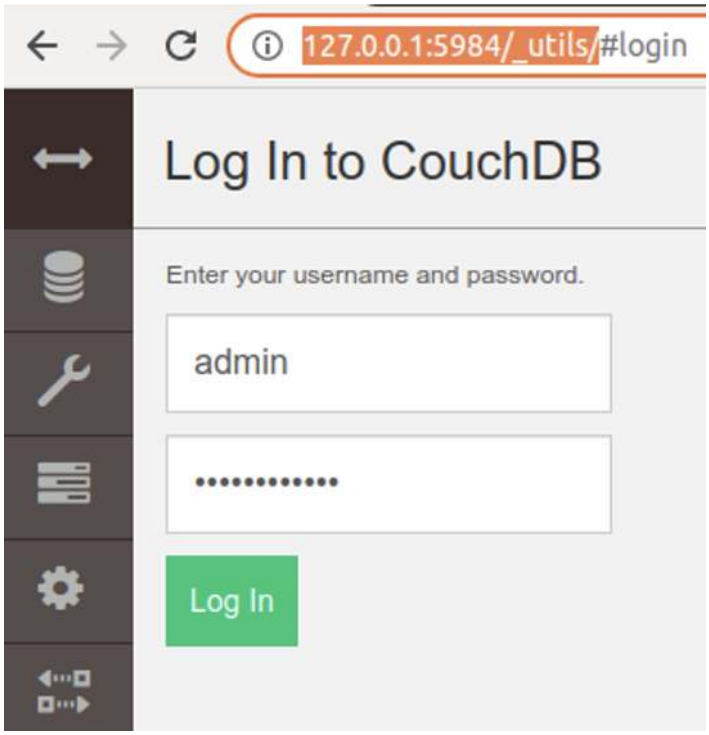
Luego de realizar los pasos de instalación, se puede acceder a la interfaz web de la base de datos; a través de la aplicación Fauxton, desde donde se puede crear, actualizar, borrar documentos.;

La dirección de acceso para usar dicha interfaz es: http://127.0.0.1:5984/_utils/

Se pedirá el ingreso del usuario de administrador y la clave (Figura 10).;

Figura 10

Ingreso de credenciales de acceso de adminstrado de CouchDB



Nota. Elizalde, R., 2025.

Luego de ingresar las credenciales de acceso, la interfaz tendrá la siguiente apariencia (Figura 11).

Figura 11

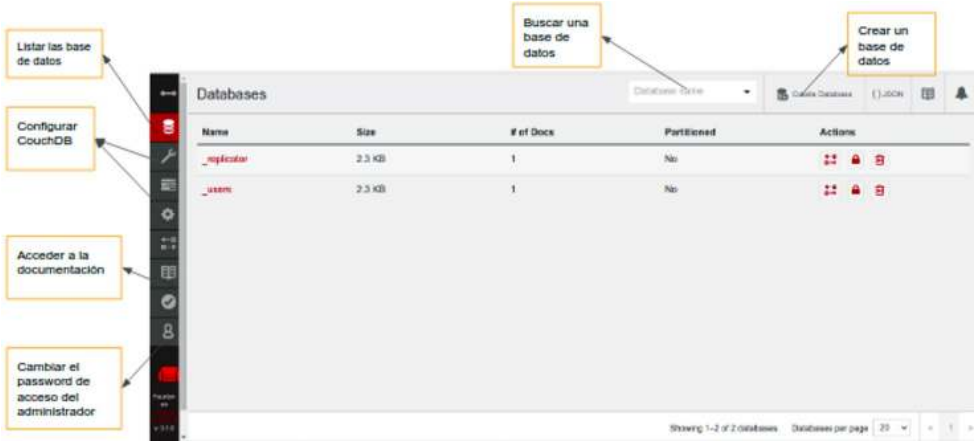
Interfaz principal para la administración de CouchDB

Nota. Elizalde, R., 2025.

En la interfaz se pueden realizar gran parte de la administración en CouchDB, como se observa en laFigura 12.

Figura 12

Opciones de administración de CouchDB a través de Fauxton.



Nota. Elizalde, R., 2025.

En la interfaz de Fauxton se presentan varias opciones que permiten administrar la base de datos de manera completa y eficiente. A continuación, se describen las principales acciones que se pueden realizar:

- Listar las bases de datos.
- Configurar CouchDB.
- Acceder a la documentación.
- Cambiar el password de acceso del administrador.
- Crear una base de datos.
- Buscar una base de datos.

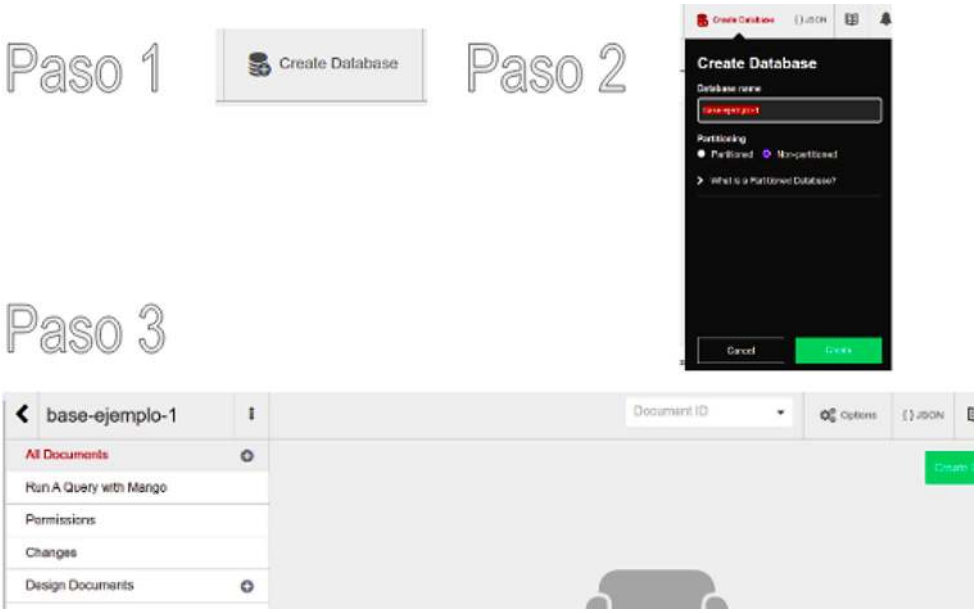
c. **Creación de una base de datos**

Para la creación de una base de datos hacemos clic en la opción “Create Database” de la interfaz Fauxton.

Luego, en la pantalla que se despliega se digita el nombre de la base de datos; en el particionamiento se selecciona **Non- partitioned**; finalmente se pulsa el botón **Create** (Figura 13).

Figura 13

Creación de una base de datos en CouchDB a través de interfaz Fauxton



Nota. Elizalde, R., 2025.

Se despliega la pantalla de acceso a la base de datos donde se puede:

- Visualizar el nombre de la base de datos
- Crear documentos en la base datos
- Permisos de acceso a la base de datos
- Diseño de documentos

d. **<u>Creación de documentos en la base de datos</u>**

Se recuerda que en CouchDB se guardan documentos, bajo el formato JSON. El proceso para crear un documento es:

1. Se pulsa el botón **Create Document**
2. Se presenta una pantalla donde se podrá ingresar los documentos en formato JSON. CouchDB genera el JSON de inicio con una llave denominada "_id" y asignado un valor tipo cadena que contiene número y letras. Este valor "_id" es importante; sirve de referencia

para buscar un documento y debe ser un valor único por cada documento.

3. Se ingresa los valores al documento

   • A partir de JSON creado de manera automática.

```
{
   "_id": "aac322d3ae479d4c3434ae6587001944",
   "ciudad": "Catamayo",
   "Provincia": "Loja"
}
```

   • Se agrega los datos, bajo el considerando; llave: valor.

```
   {
   "_id": "aac322d3ae479d4c3434ae6587001944", "ciudad":
"Catamayo",
   "Provincia": "Loja"
}
```

   En el JSON se agregó dos llaves con dos valores: la llave ciudad se le asignó el valor de Catamayo; y la llave Provincia se le agregó la llave Loja.

4. Se pulsa el botón **Create Document**

   El proceso se puede repetir de acuerdo a las necesidades de la problemática; además es importante manifestar que en cada documento puede existir un esquema (conjunto de llaves) diferente; es una de las potencialidades de CouchDB (Ver Figura 14).

Figura 14

Crear un documento en CouchDB

Nota. Elizalde, R., 2025.

e. **Consulta de datos**

Cuando se agrega un documento, en la interfaz, dentro de la base de datos se muestra un listado de los documentos; se puede visualizar en los siguientes formatos:

- **Tabla** (Figura 15)

Figura 15

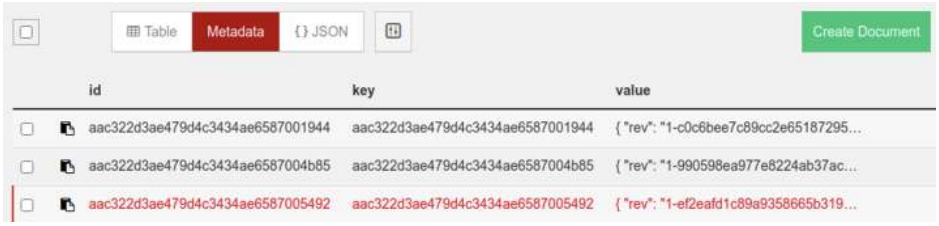
Visualización de datos en formato tabla de un documento en CouchDB.

Nota. Elizalde, R., 2025.

- **Metadata** (Figura 16)

Figura 16

Visualización de datos en formato metadata de un documento en CouchDB

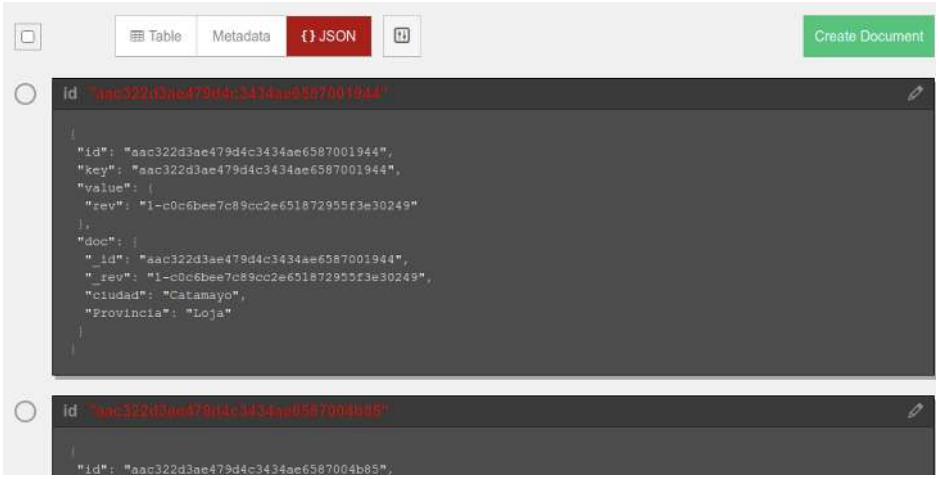


Nota. Elizalde, R., 2025.

- **JSON** (Figura 17)

Figura 17

Visualización de datos en formato JSON de un documento en CouchDB



Nota. Elizalde, R., 2025.

f. **Modificar documentos de base de datos**

En el listado de documentos, dentro de la base de datos se puede pulsar sobre un documento y se procede a cambiar los datos que se necesiten.

g. **Eliminación de documentos de base de datos**

En el listado de documentos, dentro de la base de datos se puede seleccionar los documentos que se requieran borrar y se pulsa el botón borrar (Ver Figura 18).

Figura 18

Eliminación de documentos de base de datos CouchDB

Nota. Elizalde, R., 2025.

h. **Generación de vistas en la base de datos**

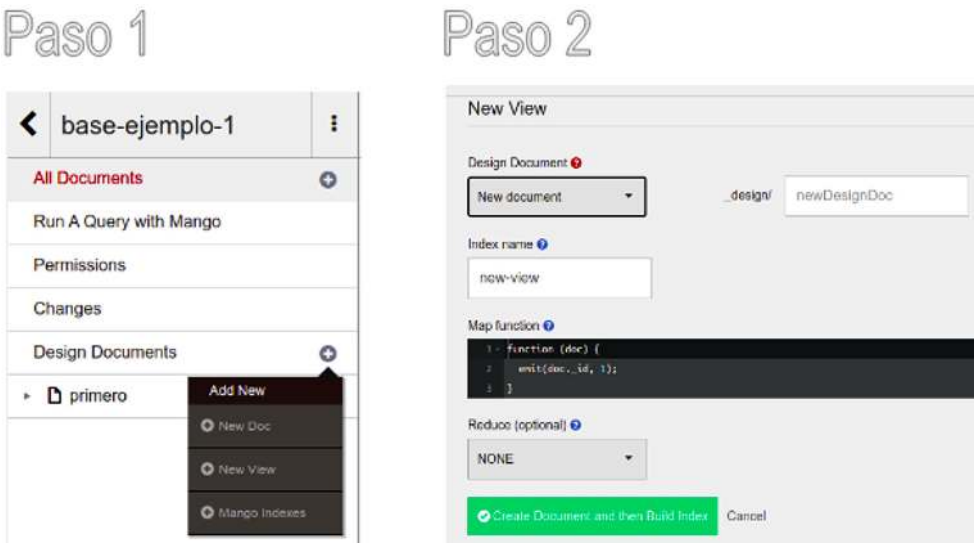
Para realizar consultas a los datos de las bases desarrolladas en CouchDB se hace uso del concepto de vistas. En este punto se debe mencionar que las consultas difieren de cómo se las realiza en modelos tipo relacional.

Una vista es un proceso que se ejecutará por cada documento. Para la creación de vistas en CouchDB a través de Fauxton se realizan los siguientes pasos (ver Figura 19):

- En la base de datos que se requiera, pulsamos la opción**Desing Documents** y luego la subsección **New View**.
- En la siguiente pantalla se escribe el nombre del documento; para el ejemplo se usa primero.
    - El nombre de vista en el campo **index name; para el ejemplo se usa new-view**.
    - Se modifica el campo **Map function**.
    - Se pulsa el botón **Create Document and the Build Index**.

Figura 19

Creación de vistas en CouchDB a través de Fauxton



Nota. Elizalde, R., 2025.

Se debe indicar algunas consideraciones en cuanto al funcionamiento de las vistas. En los puntos anteriores se nombró la expresión **creación de documento**; eso quiere decir que para crear una vista se debe crear un documento, que será tomado como un documento dentro de la propia base de datos.

El punto principal para la creación de la vista es el relacionado con Map function:

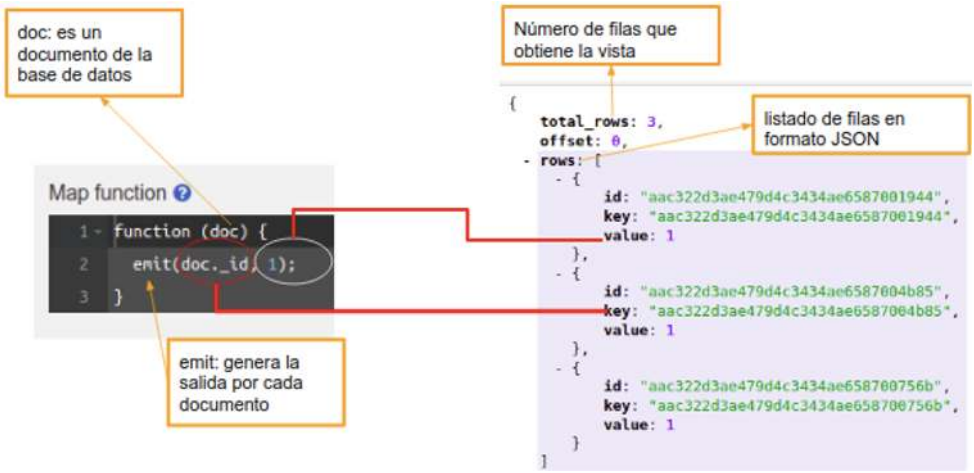
- En este campo se debe agregar código en lenguaje JavaScript.
- La estructura por defecto del JavaScript al momento de generar la vista es la siguiente:

```
function (doc) {
emit(doc._id, 1);
}
```

Donde **doc** hace referencia a un documento de la base de datos;se debe recordar qué vista será ejecutada para cada uno de los documentos que forman la base; **emit** es la respuesta. Es una lista clave-valor de los documentos que coincidan con la lógica propuesta. En el ejemplo la llave será los _id de cada documento y el valor será 1. La salida al momento de ejecutar la vista es la expuesta en la Figura 20.

Figura 20

Salida generada por una vista en CouchDB



Nota. Elizalde, R., 2025.

En líneas anteriores se manifestó que se puede consumir los datos de una base de datos de CouchDB a través del API REST que posee; se puede consumir desde un terminal de cualquier sistema operativo a través del comando **curl** por ejemplo, desde una aplicación web, etc. Veamos un ejemplo:

- Si se quiere obtener el total de documentos de la base de datos se lo hace a través de la siguiente dirección: http://127.0.0.1:5984/base-ejemplo-1/_all_docs

donde:

- http://127.0.0.1:5984 es la dirección y puerto de nuestro servidor de CouchDB.
- base-ejemplo-1 es el nombre de la base de datos.
- _all_docs es la función propia de CouchDB que retorna los documentos de la base de datos; la función mencionada está disponible para todas las bases de datos del servidor de CouchDB (Figura 21).

Figura 21

Uso de la función _all_docs para una base de datos en CouchDB

Nota. Elizalde, R., 2025.

Para comprender mejor cómo se aplica esta funcionalidad en la práctica, a continuación, se presenta un ejemplo que le permitirá observar directamente cómo los conceptos previamente explicados se llevan a la práctica.

Ejemplos de CouchDB con jQuery

Ejemplo 1

Generar una base de datos en couchDB que almacene la siguiente información:

```
{
"ciudad": "Catamayo",
"provincia": "Loja"
}
{
"ciudad": "Machala",
"provincia": "El Oro"
}
{
"ciudad": "Loja",
"provincia": "Loja"
}
{
"ciudad": "Cuenca",
}
```

Generar una vista que devuelva todos los documentos que tenga una llave provincia, tal como se observa en la figura 22.

Figura 22

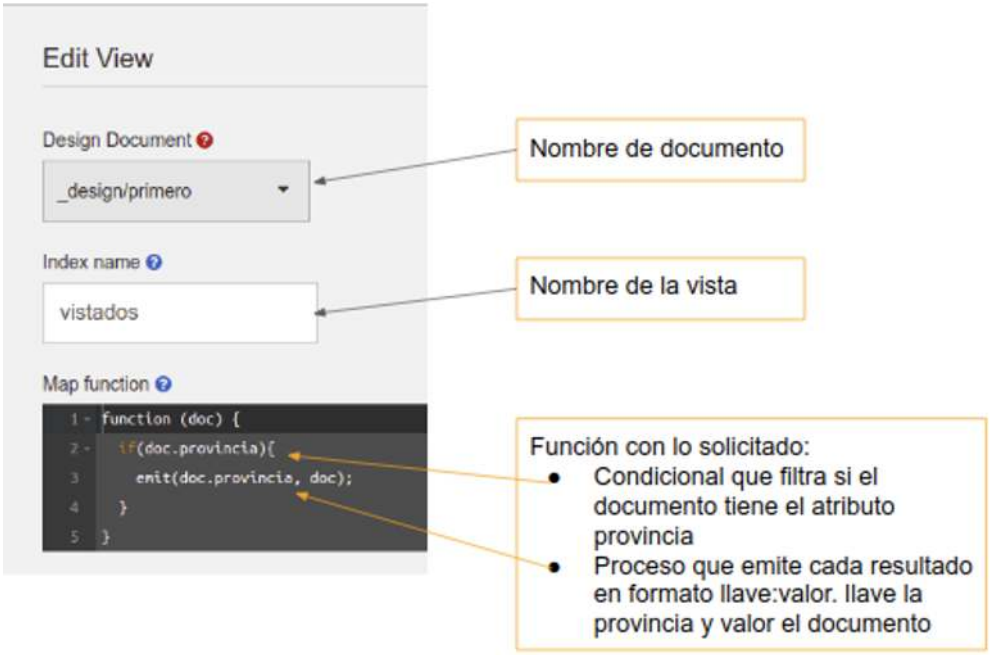
Datos ingresados a la base de datos CouchDB

Nota. Elizalde, R., 2025.

- Se crea la base de datos y se agrega los elementos dados en la problemática (Figura 22).
- Se genera el documento y la vista solicitada (Figura 23).

Figura 23

Vista generada en base a la problemática del Ejemplo 1



Nota. Elizalde, R., 2025.

- Salida, a través del uso del comando curl, desde un terminal. (ver Figura 24).
  - En terminal de su sistema operativo digitamos lo siguiente:

    ```
    curl http://127.0.0.1:5984/base-ejemplo-1/_design/ primero/_view/ vistados
    ```

    donde:

    - http://127.0.0.1:5984 es la dirección y puerto de nuestro servidor de CouchDB.
    - base-ejemplo-1 es el nombre de la base de datos.
    - _design, expresión que va en la URL que se quiere consumir una vista.
    - primero, nombre del documento generado
    - _view, expresión que va en la URL que se quiere consumir una vista.

- Vistados, nombre de la vista.

Figura 24

Salida generada en base a la vista para solucionar la problemática

Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git y agregar la información a su base de datos NoSql: Desarrollo del ejemplo 2.4.1_1

2.4.2. Consumo de datos de CouchDB desde JavaScript

Para el presente ítem de estudio se plantea usar la librería jQuery. Se presenta el siguiente ejemplo que demuestra la potencialidad de consumo que posee la base de datos CouchDB, a través de su servicio REST-API

a. **Ejemplo 1**

- Dada una base de datos en CouchDB que tiene los siguientes datos:

```
{
"ciudad": "Catamayo",
"provincia": "Loja"
}
{
"ciudad": "Machala",
"provincia": "El Oro"
}
{
"ciudad": "Loja",w
"provincia": "Loja"
```

```
}
{
"ciudad": "Cuenca",
}
```

- Crear una página web que haga uso de la librería jQuery y la técnica de programación AJAX para acceder a los documentos de la base de datos que tengan la llave provincia y los despliega en la página creada.

  Para solucionar lo anterior en primer lugar se genera la base de datos en couchDB, luego se agrega los datos y se crea la vista correspondiente. Se usará la base de datos y vista generados en el Ejemplo 1 del punto 2.4.2 de la presente guía.

  El diseño de la página web se realiza a través de las siguientes líneas de código.

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width" />
6   <title>Demo jQuery - jQueryUI - CouchDB</title>
7   <link rel="stylesheet" href="https://code.jquery.com/ui/
1.12.1/themes/base/ jquery-ui.css">
8   <script src="https://code.jquery.com/jquery-3.5.0.js"></
script>
9   <script src="https://code.jquery.com/ui/1.12.1/jquery-
ui.js"></script>
10  </head>
11  <body>
12  ····
13  <h3>Ejemplo de Uso de Jquery + CouchDB</h3>
14  <div id="pestanias">
15  <ul>
16  <li><a href="t1">Principal</a></li>
17  </ul>
18  <div id="t1">
19  <p id="mensajeFinal"></p>
20  <button type="submit">Obtener información de CouchDB</
```

```
button>
21  </div>
22  </div>
23  </body>
24  <script>
25  $(document).ready(function() {
26  $("#pestanias").tabs();
27  $( "button" ).click(function() {
28  $.ajax({
29  dataType: 'json',
30  url: "http://127.0.0.1:5984/base-ejemplo-1/_design/primero/
_view/ vistados"
31  }).done(function(data) {
32  for(var i=0; i<data.rows.length; i++){
33  var c = data.rows[i].value.ciudad;
34  var prov = data.rows[i].value.provincia;
35  $('#mensajeFinal').append("<p> <b>Ciudad:</b> "+ c + "
<b>Provincia:</b> " +
36  prov + "</p>");
37  }
38  }).fail(function(msg, texto, textoerror) {
39  console.log( "error" );
40  console.log( msg );
41  console.log( textoerror );
42  });
43  });
44  });
45  </script>
46  </html>
```

Para la ejecución del código se solicita alojar el archivo en un servidor web; además su servidor de base de datos CouchDB debe estar en el mismo computador.

En el código se hace uso de la técnica de desarrollo de ambiente web AJAX para acceder al REST API de CouchDB, particularmente a la vista generada. Se explica a continuación el proceso realizado desde la línea 24 del código presentado.

- En la línea 26 se hace uso del contenedor de paneles tabs de jQueryUI.

- La línea 27 indica que cuando se haga un clic en el componente button, se generan algunos procesos.
- En la línea 28 empieza una petición AJAX a través de la librería jQuery.
- Con dataType = 'json' en la línea 29 se especifica el tipo de dato esperado del servidor.
- El atributo **url** permite indicar la URL a la cual se hace la petición; en este caso se hace uso de la URL que hace la petición servicio REST API que devuelve el resultado de la vista generada en CouchDB.
- Cuando la respuesta del servicio generado es correcta se hace uso del controlador **done** línea 31; y se realiza todo lo expresado en la 32 y 37. Manifestar que la información recibida desde el servidor se la tiene a través de la variable **data**.

    - Se recuerda la información que devuelve el REST API de CouchDB.

      REST API de CouchDB.


```
{"total_rows":3,"offset":0,
"rows":[
{"id":"aac322d3ae479d4c3434ae6587004b85","key":"El
Oro","value":{"_
id":"aac322d3ae479d4c3434ae6587004b85","_rev":"2-6c1df38d25d3
159894274fa b5f63e7bc","ciudad":"Machala","provincia":"El
Oro"}},
{"id":"aac322d3ae479d4c3434ae6587001944","key":"Loja","value"
:{"_
id":"aac322d3ae479d4c3434ae6587001944","_rev":"2-115c7b725cf3
479a717670
7ecf5764e4","ciudad":"Catamayo","provincia":"Loja"}},
{"id":"aac322d3ae479d4c3434ae658700756b","key":"Loja","value"
:{"_
id":"aac322d3ae479d4c3434ae658700756b","_rev":"2-360519e7940f
cae778c3d86 4fb1b0f87","ciudad":"Loja","provincia":"Loja"}}
]
}
```


    - La variable data está en formato JSON y posee las siguientes llaves:

        - "total_rows", "offset","rows"

- De las llaves, la información que se necesita está en la llave “rows”; la misma es una lista,. Por lo tanto, se debe hacer un proceso repetitivo para acceder a cada elemento de la lista. Los elementos tienen las siguientes llaves:
  - “id”, “key”, “value”
- Interesa la llave “value” de cada elemento, para acceder a los valores requeridos. Cada “value” tiene las siguientes llaves
  - “_id”, “_rev”, “ciudad”, “provincia”
- De las llaves anteriores se usa “ciudad” y “provincia” y sus valores correspondientes en cada iteración para agregar una etiqueta tipo párrafo <p> al elemento con **id= “mensajeFinal”**.

• Si existe algún inconveniente con la petición:

- No está bien formada la URL
- No se tiene los permisos de acceso
- Los datos devueltos están mal formados

Se usa el controlador **fail** para manejar el proceso, línea 38 del ejemplo.

El resultado final de la aplicación se muestra en las siguientes figuras:

• Cuando se lanza la aplicación:

Figura 25

Salida inicial de la aplicación web del Ejemplo 1

Nota. Elizalde, R., 2025.

- Cuando se pulsa el botón "**Obtener información de CouchDB**".

Figura 26

Salida final del Ejemplo 1 de la aplicación web luego de ejecutar la acción del botón "Obtener información de CouchDB"

Nota. Elizalde, R., 2025.

Ahora, para reforzar lo aprendido, realice las siguientes actividades:

Actividades de aprendizaje recomendadas

1. Esta actividad de aprendizaje se centra en la exploración de un repositorio ejemplo de CouchDB, diseñado para demostrar diversas formas de utilizar un archivo de datos en formato JSON. Los estudiantes exploran el proceso de migración de estos datos hacia una base de datos CouchDB, empleando tanto comandos CURL para interacciones directas como scripts en Python para automatización. Adicionalmente, se ejemplificará el consumo y la visualización de datos extraídos de CouchDB en una página HTML sencilla, ilustrando el flujo completo desde la gestión de la base de datos hasta la presentación en la Interfaz de Usuario.
2. Es momento de poner en práctica los conceptos de la unidad 2: *Programación del lado del cliente*. Por ello, se solicita revisar la unidad 2 de la presente guía y recrear los ejemplos en sus máquinas personales. Luego, realice la autoevaluación 2, compuesta por preguntas de única respuesta, basadas en los temas estudiados y en los recursos educativos abiertos listados en la parte introductoria de la actividad.

   La autoevaluación le permitirá identificar el nivel de aprendizaje alcanzado y los ítems que se deben volver a revisar para lograr los resultados de aprendizaje propuestos. Además, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Autoevaluación 2

Instrucción: lea detenidamente cada uno de los enunciados de las preguntas y seleccione la o las respuestas correctas.

1. **Para la creación de aplicaciones web intuitivas existen algunas recomendaciones; identifique una de ellas del siguiente listado:**

   a. Generar pruebas con usuarios desarrolladores.

b. Generar pruebas con usuarios reales.
c. Generar pruebas con el gerente de la empresa solicitante.

2. **En un archivo HTML existe la siguiente línea de código:**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
```

Identifique lo que significa dicha línea:

a. La versión de HTML que será interpretada.
b. La versión de JavaScript que será usada.
c. La versión de CSS que será asociada.

3. **¿Cuál es la razón principal de usar la herramienta *Markup Validation Service*?**

a. Verificar que la aplicación *web* esté desarrollada bajo los estándares de Mozilla Firefox, Chrome.
b. Verificar que la aplicación *web* esté desarrollada bajo los estándares expuestos por los clientes.
c. Verificar que la aplicación *web* esté desarrollada bajo los estándares de la *World Wide Web Consortium*.

4. **Dado el siguiente código HTML, identifique la sentencia que hace falta agregar en la línea 4 para establecer como conjunto de caracteres por interpretar: UTF-8.**

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4
5   <meta name="viewport" content="width=device-width" />
6   <title></title>
7   </head>
```

```
8    <body>
9    <p> Nombre</p>
10   <p>Apellido</p>
11   <p>Cédula</p>
12   </body>
13   </html>
```

a. `<script charset="utf-8" />`
b. `<link charset="utf-8" />`
c. `<meta charset="utf-8" />`

5. **Dados los siguientes archivos, identifique la opción correcta para cambiar la línea 12, con el objetivo de visualizar en la etiqueta <p>: un fondo azul y letras de color blanco.**

```
index.html
1    <!DOCTYPE html>
2    <html>
3    <head>
4    <meta charset="utf-8" />
5    <meta name="viewport" content="width=device-width" />
6    <link rel="stylesheet" href="estilos.css" type="text/css" /
>
7    <title>Pregunta Repaso</title>
8    </head>
9    <body>
10   <p>Nombre</p>
11   <p>Apellido</p>
12   <p>Cédula</p>
```

```
13  </body>
14  </html>
```

```
estilos.css
1
2   .tres{
3   background: blue;
4   color: white;5  }
```

a. `<p id="tres">Cédula</p>`
b. `<p class="tres">Cédula</p>`
c. `<p class[id]="tres">Cédula</p>`

6. **Dado el siguiente código, ¿cuál es el resultado que se visualizará en el navegador, si se ingresa el valor de 10 para la variable entrada1?**

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width" />
6   <title>Pregunta Repaso</title>
7   </head>
8   <body>
9   <p id="uno"></p>
10  <p id="dos"></p>
11
12  <script type="text/javascript" charset="utf-8">
13  var entrada1 = window.prompt("Escriba el primer valor: ",
"0");
```

```
14  var a = parseFloat(entrada1);
15  for (var i = 0; i < a; i++) {
16  document.getElementById("uno").innerText = i
17  }
18  document.getElementById("dos").innerText = "<b>" + i + "</
b>";
19  </script>
20  </body>
21  </html>
```

Respuestas

a. `9`

   `10`

b. `9`

   `<b>10</b>`

c. `<b>9</b>`

   `<b>10</b>`

7. **Para acceder a la librería jQuery a través de la Red de Distribución de Contenido y usar sus propiedades en una página web. ¿Cuál de las siguientes líneas de código se debe agregar?**

   a. `<script src="https://code.jquery.com/jquery-3.5.0.js"></script>`

   b. `<a href="https://code.jquery.com/jquery-3.5.0.js"></script>`

   c. `<script href="https://code.jquery.com/jquery-3.5.0.js"></script>`

8. **¿Cuál es el modelo NoSql usado en CouchDB y Neo4J?**

a. CouchDb: basado en documentos; Neo4J: basado en grafos.
b. CouchDb: orientado a columnas; Neo4J: basado en grafos.
c. CouchDb: basado en documentos; Neo4J: Llave-valor.

9. **¿Cómo se llama la interfaz administrativa de CouchDB y cuál es la dirección de acceso local por defecto?**

   a. Interfaz: Fauxton; dirección de acceso local: http//127.0.0.1 : 5984/
   b. Interfaz: CouchDBWeb; dirección de acceso local: http//127.0.0.1 :5984/_utils/
   c. Interfaz: Fauxton; dirección de acceso local: http//127.0.0.1 : 5984/_utils/

10. **¿Cuál de los siguientes documentos son válidos para guardarlos en una base de datos CouchDB?**

    a.
    ```
    {
    "_id": aac322d3ae479d4c3434ae6587001944, "ciudad": Quito,
    "Provincia": Pichincha
    }
    ```
    b.
    ```
    {
    _id: "aac322d3ae479d4c3434ae6587001944", ciudad: "Quito",
    Provincia: "Pichincha"
    }
    ```
    c.
    ```
    {
    "_id": "aac322d3ae479d4c3434ae6587001944", "ciudad":
    "Quito",
    "Provincia": "Pichincha"
    ```

Ir al solucionario

Hemos finalizado con éxito nuestra segunda unidad de estudios. ¡Avancemos con ánimo!

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 5

Unidad 3. Acceso a base de datos relacionales mediante *Object Relational Mapper* (ORM)

Estimado estudiante, durante su formación académica ha tenido la oportunidad de interactuar con bases de datos relacionales y bases de datos no relacionales. En este apartado se retoman conceptos de base de datos relacionales y se aborda una nueva temática que tiene relación con el manejo de datos a través de *Object Relational Mapper* o conocido por sus iniciales ORM.

3.1. Introducción

En referencia a *Object Relational Mapper* (ORM), Cabedo et al., (2010) indica que es una técnica que permite a desarrolladores pasar los datos de lenguajes de programación bajo el paradigma de Orientación a Objetos a datos persistentes para su posterior almacenamiento en bases de datos relacionales.

Pérez Martín (s.f.), en su investigación, presenta algunas ventajas y limitaciones del uso de ORM en el desarrollo de una solución.

Ventajas del uso de ORM

- Proceso de desarrollo más rápido

- Separación de las bases de datos; cuando se utiliza un ORM se puede cambiar el motor de base de datos en cualquier momento. Dado que los ORM transforman automáticamente las consultas, también deben tener la capacidad de adaptarse a diversos gestores de base datos como: MySql, Postegres, Oracle, Sqlite, etc.
- Seguridad, en los ORM existen procedimientos que impiden ataques como SQL injections.

Limitaciones del uso de ORM

- Curva de aprendizaje: es difícil explotar en forma completa un ORM por su extensión; por tal razón, se debe tomar un tiempo considerable para dominar y aplicarlo en una solución.
- Aplicaciones ralentizadas; la aplicación de ORM genera que los procesos sean más lentos en un porcentaje bajo; se debe considerar y sopesar con la velocidad del desarrollo.

Librerías ORM según los lenguajes de programación

Para facilitar la comparación y comprensión de las principales librerías *Object-Relational Mapping* (ORM) según el lenguaje de programación, se presenta la siguiente tabla que resume las opciones más utilizadas en cada caso:

Tabla 2
Listado de librerías ORM

Lenguaje de programación	Librerías ORM
PHP	Doctrine Eloquent Active Record Class (Codeigniter) Zend-db
Java	Hibernate Ebean
Ruby	Active Record Ruby Object Mapper Sequel
Python	SqlAlchemy Django-ORM Pony-ORM Peewee

Nota. Elizalde, R., 2025.

3.2. Manejo de datos con ORM SqlAlchemy

En el presente apartado se va a ejemplificar el uso de la librería ORM denominada SqlAlchemy que está desarrollada en lenguaje Python. Se recomienda revisar la documentación de SQLAlchemy.

Estimado estudiante en su computador necesita crear el ambiente de desarrollo óptimo para realizar los ejemplos propuestos.

Se necesita instalar:

- Lenguaje de programación Python (puede revisar el enlace que permite la descarga e instalación en diversos sistemas operativos)
- Instalación de la librería SqlAlchemy (revisar proceso de instalación); se recomienda usar el proceso vía el gestor de librerías de Python denominado

pip SqlAlchemy está dividido en dos grandes funcionalidades y áreas como lo indica el documento Learning sqlAlchemy; la relacionada con SqlAlchemy Core y qlAlchemy ORM. Se revisará la segunda área en la presente guía.

SqlAlchemy ORM permite manejar un patrón que permite pasar las clases desarrolladas en Python a una base de datos de forma simple y elegante. Manifestar que los ejemplos siguientes pueden ser desarrollados o probados de dos formas:

- Agregando el código a través de un editor de texto en un archivo con extensión py y luego ejecutar desde un terminal o consola con el patrón: python [nombre_archivo].py
- Usando la consola por defecto de python o la librería ipython para hacer uso de la librería existen algunas consideraciones:
- Conectarnos a la base de datos a través del siguiente código

```
1 from sqlalchemy import create_engine
2
3   # se genera enlace al gestor de base de
4   # datos
5   # para el ejemplo se usa la base de datos
6   # sqlite
7   engine = create_engine('sqlite:///demobase.db')
```

En el código anterior se importa el módulo create_engine (línea 1) que permite el enlace hacia un motor de base de datos; en la línea 7 se crea un variable llamada engine que hace uso de create_engine, a la cual se envía como parámetro una cadena de texto que indica el motor a usar. Para el ejemplo se usa un enlace para una base de datos SQLite, el nombre asignado para la base de datos es **demobase.db**.

Estimado estudiante, se solicita instalar en su computador la base de datos y el visor de base de datos SQLite.

En la siguiente tabla se muestran ejemplos de conexiones para distintos gestores de bases de datos, con el fin de ilustrar la sintaxis y parámetros comúnmente utilizados en cada caso.

Tabla 3
Ejemplos de conexiones para algunos gestores de base de datos

Gestor de base datos	Conexión
Postgres	# por defecto **create_engine**('postgresql://usuario:clave@localhost:5432/basedatos') # usando la librería psycopg **create_engine**('postgresql+psycopg2://usuario:clave@localhost:5432/basedatos')
Mysql	# por defecto **create_engine** ('mysql://usuario:clave@localhost/base-de-datos') # usando la librería de Python mysqlclient **create_engine**('mysql+mysql://usuario:clave@localhost/base-de- datos')
Oracle	# por defecto **create_engine**('oracle://usuario:clave@127.0.0.1:1521/esquema') # usando la librería de python **create_engine**('oracle+cx_oracle://usuario:clave@esquema')

Nota. Elizalde, R., 2025.

- Declaración de las clases que en lo posterior se transforman en tablas de la base de datos.

```
8
9   from sqlalchemy.ext.declarative import declarative_base
10  Base = declarative_base()
11
```

Para realizar el proceso se necesita importar la función **declarative_ base** como se observa en la línea 9.

En la línea 10 se crea una clase llamada Base para definir las clases en lenguaje Python.

La clase Base se usa como superclase para todas las clases que se necesitan inicializar.

```
13  from sqlalchemy import Column, Integer, String
14
15  class Saludo(Base):
16  tablename = 'saludo'
17  id = Column(Integer, primary_key=True)
18  mensaje = Column(String)
19
20  Base.metadata.create_all(engine)
```

En la línea 13 se importa las clases Column, Integer y String, donde Column representa una columna en la base de datos. Integer y String son clases que permiten asignar un tipo de dato a la columna.

En la línea 15 se crea una clase llamada Saludo que hereda de la clase Base. A través del atributo de la clase tablename se identifica el nombre de la tabla en la base de datos.

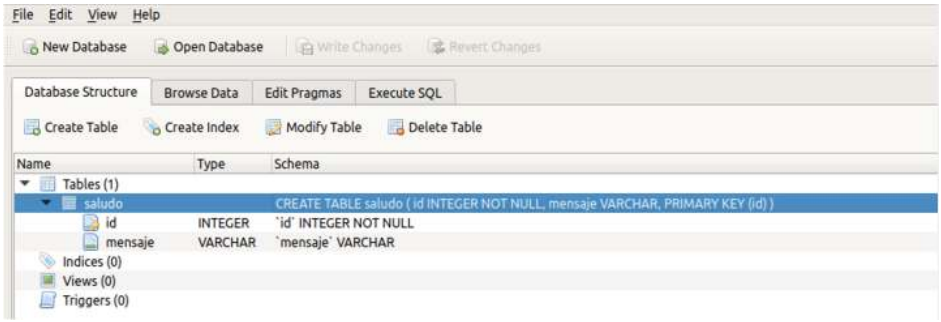
En la línea 17 se crea un atributo de clase llamado id que será una columna que tiene como características que acepta datos de tipo entero y es la clave primaria de la entidad.

En la línea 18 se crea un atributo llamado mensaje que será una columna de tipo cadena.

Con la sentencia de la línea 20 se crean las tablas que no estén creadas aún en la base de datos, ver Figura 27.

Figura 27

Tabla generada desde SqlAlchemy, visualizada desde DB



Nota. Elizalde, R., 2025.

Ahora, se revisarán algunos puntos para guardar información en la base de datos y entidad recién creada.

```
22 from sqlalchemy.orm import sessionmaker
23
24 Session = sessionmaker(bind=engine)
25 session = Session()
```

Algunas explicaciones; sessionmaker es una clase generadora de clases de tipo sesión; se usa como configuración los parámetros enviados a través del constructor. Para el ejemplo, se envía el enlace creado para la base de datos, **engine**.

En la línea 21 se crea una clase de Python llamada Session, desde el generador sessionmaker.

En la línea 22 se crea un objeto session de tipo Session, el que va a permitir guardar, eliminar, actualizar, generar consultas en la base de datos respecto a las entidades creadas.

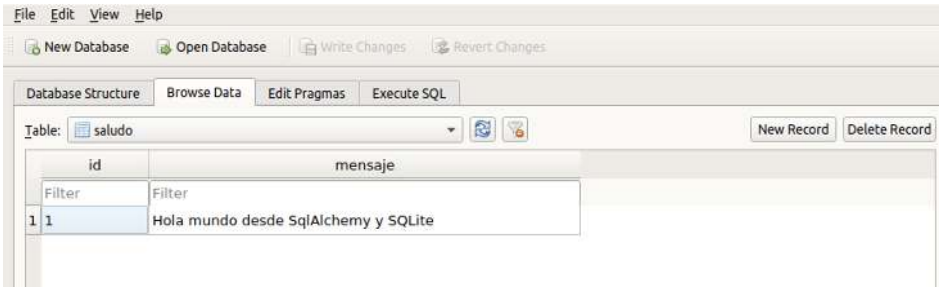
Ahora, se deja unas líneas de código que permiten crear un objeto de tipo Saludo y guardar dicho objeto como registro en la base de datos.

```
# se crea un objeto de tipo
# Saludo

miSaludo = Saludo()
miSaludo.mensaje = "Hola mundo desde SqlAlchemy y SQLite"

# se agrega el objeto miSaludo
# a la entidad Saludo a la sesión
# a la espera de un commit
# para agregar un registro a la base de·
# datos demobase.db
session.add(miSaludo)

# se confirma las transacciones
session.commit()
```

Como se puede visualizar en Figura 28 la base de datos demobase. db ya tiene registros.

Figura 28

Tabla generada desde SqlAlchemy con registros, visualizada desde DB browser for SQLite



Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 3.2_1

3.3. Consulta de datos con ORM SqlAlchemy

Para realizar la consulta de información a una base de datos a través del ORM de SqlAlchemy se deben considerar algunos puntos:

- Usar la variable **session**
- Se usa el método **query()**; se accede desde *session*
- Al método query se le puede enviar como argumentos clases o características de una clase
- Al método query se le agrega algunas opciones como: *all, order_by, filter, filter_by*

En el Anexo 1 podrá observar un ejemplo de esta consulta de datos, el cual le permitirá comprender a cabalidad este tema. Además, estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 3.3_1

Se pueden generar muchos ejemplos, revisar la referencia de manejo de Querying en la página oficial de SqlAlchemy.

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 6

Unidad 3. Acceso a base de datos relacionales mediante *Object Relational Mapper* (ORM)

3.4. Eliminación de datos con ORM SqlAlchemy

La eliminación de registros a través del ORM del SqlAlchemy se lo realiza mediante el uso de la expresión **delete** asociada a la sesión creada en el proceso.

Ejemplo:

- Crear una estructura que permita manipular a los docentes con las características como: nombre, apellido, ciudad.

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, relationship
from sqlalchemy import Column, Integer, String, ForeignKey

# se importa información del archivo configuración
from configuracion import cadena_base_datos

# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)

Base = declarative_base()

class Docente(Base):
  tablename  = 'docentes'
 id = Column(Integer, primary_key=True)
 nombre = Column(String)
 apellido = Column(String)
 ciudad = Column(String, nullable=False)
 ....
 def   repr  (self):
 return "Docente: nombre=%s apellido=%s ciudad:%s" % (
 self.nombre,·
 self.apellido,·
 self.ciudad)

Base.metadata.create_all(engine)
```

- Ingresar información.

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

# se importa la clase(s) del·
# archivo genera_tablas
from genera_tablas import Docente

# se importa información del archivo configuracion
```

```
from configuracion import cadena_base_datos
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)

Session = sessionmaker(bind=engine)
session = Session()

# se crea un objetos de tipo Docente·
docente1 = Docente(nombre="Tony", apellido="García", \
ciudad="Loja")

docente2 = Docente(nombre="Luis", apellido="Borrero", \
ciudad="Loja")

docente3 = Docente(nombre="Ana", apellido="Salcedo", \
ciudad="Zamora")

docente4 = Docente(nombre="Monica", apellido="Valenzuela", \
ciudad="Zamora")

# se agrega los objetos
# a la sesión
# a la espera de un commit
# para agregar un registro a la base de·
# datos
session.add(docente1)
session.add(docente2)
session.add(docente3)
session.add(docente4)

# se confirma las transacciones
session.commit()
```

Respuesta:

```
Presentación de Docentes
Docente: nombre=Tony apellido=García ciudad:Loja
---------
```

```
Docente: nombre=Luis apellido=Borrero ciudad:Loja
---------
Docente: nombre=Ana apellido=Salcedo ciudad:Zamora
---------
Docente: nombre=Monica apellido=Valenzuela ciudad:Zamora
---------
```

- Eliminar los docentes que tengan el apellido igual a “Valenzuela”.

```
1   from sqlalchemy import create_engine
2   from sqlalchemy.orm import sessionmaker
3   from sqlalchemy import and_ # se importa el operador and
4
5   # se importa la clase(s) del·
6   # archivo genera_tablas
7   from genera_tablas import Docente·
8
9   # se importa información del archivo configuracion
10  from configuracion import cadena_base_datos
11  # se genera enlace al gestor de base de
12  # datos
13  # para el ejemplo se usa la base de datos
14  # sqlite
15  engine = create_engine(cadena_base_datos)
16
17
18  Session = sessionmaker(bind=engine)
19  session = Session()
20
21  # Obtener todos los registros de·
22  # la entidad docentes·
23  docentes = session.query(Docente).all()
24
25  # Se recorre la lista a través de un ciclo
26  # repetitivo for en python
27  print(“Presentación de Docentes”)
28  for s in docentes:
29  print(“%s” % (s))
30  print(“ ”)
31
32 # Eliminar datos o registros de los docentes
```

```
33  # que tiene como apellido “Valenzuela”
34  docentes_eliminar = session.query(Docente).filter(Docente.
apellido==”Valenzuela”)\
35  .all()
36
37  for d in docentes_eliminar:
38  # se agrega a la sesión los elementos que
39  # se quiere eliminar, a través de
40  # session.delete()
41  session.delete(d)·
42
43  # se confirma las transacciones
44  session.commit()
45
46  # Obtener todos los registros de·
47  # la entidad docentes·
48  docentes = session.query(Docente).all()
49
50  # Se recorre la lista a través de un ciclo
51  # repetitivo for en python
52  print(“Presentación de Docentes luego de eliminar”)
53  for s in docentes:
54  print(“%s” % (s))
55  print(“ ”)
56
```

Respuesta:

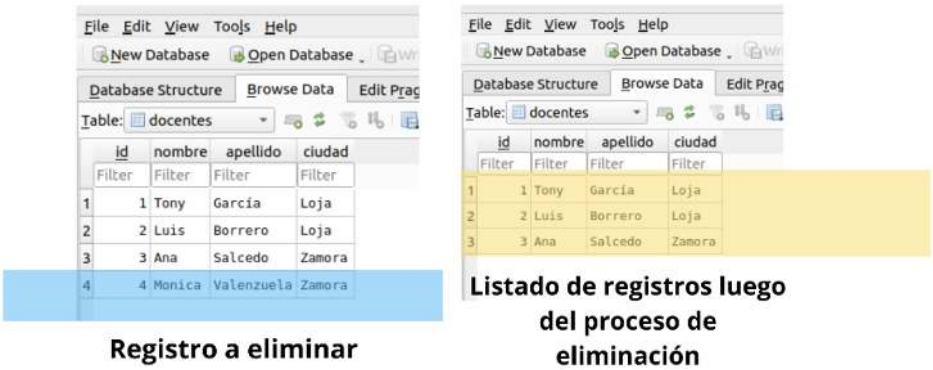
```
Presentación de Docentes
Docente: nombre=Tony apellido=García ciudad:Loja

_______
Docente: nombre=Luis apellido=Borrero ciudad:Loja

_______
Docente: nombre=Ana apellido=Salcedo ciudad:Zamora

_______
Docente: nombre=Monica apellido=Valenzuela ciudad:Zamora

_______
Presentación de Docentes luego de eliminar
Docente: nombre=Tony apellido=García ciudad:Loja

_______
Docente: nombre=Luis apellido=Borrero ciudad:Loja
```

```
_______
Docente: nombre=Ana apellido=Salcedo ciudad:Zamora
_______
```

En la figura 29, se pueden visualizar los cambios en la base de datos.

Figura 29

Visualización de registros de la tabla docentes



Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: [Desarrollo del ejemplo 3.4_1]

Usted puede revisar información del proceso de eliminación a través de la expresión delete en SqlAlchemy en Query.delete

Ahora, continuemos con el estudio del siguiente tema que trata sobre las relaciones de entidades con ORM SqlAlchemy, donde exploraremos cómo vincular clases mediante relaciones uno a uno, uno a muchos y muchos a muchos, y cómo estas relaciones facilitan la manipulación y consulta de datos de manera eficiente en Python.

ORM SqlAlchemy: Relaciones entre Clases

3.5. Relaciones de entidades con ORM SqlAlchemy

A través del ORM de SqlAlchemy se puede crear las relaciones entre clases, de la misma forma que se hace cuando se diseña e implementa una base de datos con lenguaje SQL:

- Uno a uno.
- Uno a muchos.
- Muchos a muchos.

Ejemplo

Dadas las entidades Estudiante y Número Telefónico, que poseen las siguientes características:

- Estudiante:
  - nombre.
  - apellido.
  - cedula.
- NumeroTelefónico.
  - numero_telefónico.
  - tipo.
  - estudiante.

Entre las dos entidades existe una **relación de uno a muchos**; un estudiante puede tener muchos números telefónicos; para el ejemplo propuesto se necesita agregar una **llave foránea** en la entidad **NúmeroTelefónico**.

La generación de las entidades en Python usando el ORM de SqlAlchemy es de la siguiente forma:

```
1   from sqlalchemy import create_engine
2   from sqlalchemy.ext.declarative import declarative_base
3   from sqlalchemy.orm import sessionmaker, relationship
4   from sqlalchemy import Column, Integer, String, ForeignKey
5
6   # se importa información del archivo configuracion
```

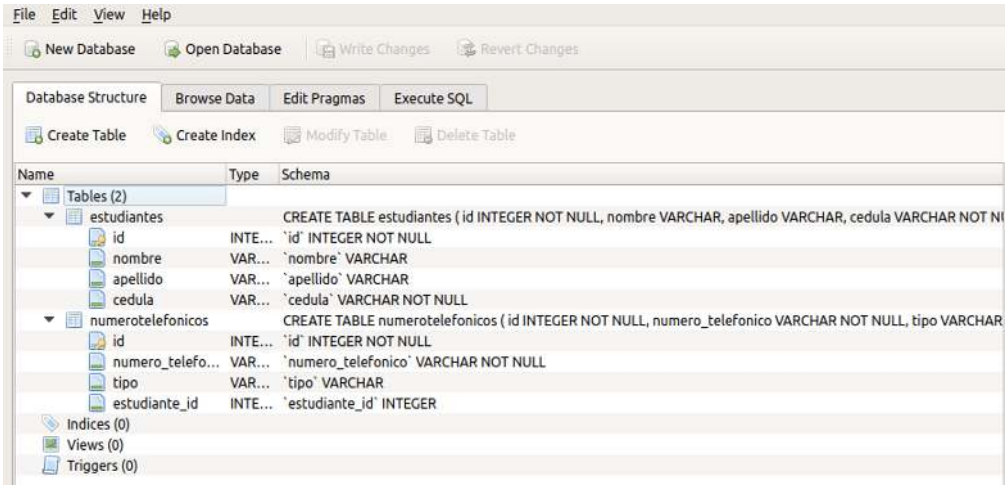
```
7   from configuracion import cadena_base_datos
8
9   # se genera enlace al gestor de base de
10  # datos
11  # para el ejemplo se usa la base de datos
12  # sqlite
13  engine = create_engine(cadena_base_datos)
14
15 Base = declarative_base()
16
17  class Estudiante(Base):
18    tablename    = 'estudiantes'
19  id = Column(Integer, primary_key=True)
20  nombre = Column(String)
21  apellido = Column(String)
22  cedula = Column(String, nullable=False)
23
24  def   repr  (self):
25  return "Estudiante: nombre=%s apellido=%s cedula=%s" % (
26  self.nombre,·
27  self.apellido,·
28  self.cedula)
29
30 class NumeroTelefonico(Base):
31    tablename    = 'numerotelefonicos'
32  id = Column(Integer, primary_key=True)
33  numero_telefonico = Column(String, nullable=False)
34  tipo = Column(String)
35  estudiante_id = Column(Integer, ForeignKey('estudiantes.id'))
36  estudiante = relationship("Estudiante",
back_populates="numerostelefonicos")
37  ····
38  def   repr  (self):
39  return "Número Telefónico %s" % (self.numero_telefonico)
40
41 Estudiante.numerostelefonicos = relationship("NumeroTelefonico", \
42  back_populates="estudiante")
43
44 Base.metadata.create_all(engine)
```

Se explica las líneas más importantes:

- En la línea 35 se crea un atributo **estudiante_id**, que será una columna de tipo entero; se le agrega la directiva **ForeignKey** que indica que los valores en dicha columna deben estar presentes en la columna de la entidad que se pasa como parámetro, para el ejemplo **'estudiantes.id' (entidad estudiantes, columna id)**. Se crea una relación entre numerotelefonicos. **estudiante_id** y estudiantes.**id**
- Se usa la directiva **relationship** en la línea 36 para indicarle al ORM que la clase **NumeroTelefonico** está relacionada con **Estudiante** a través de **NumeroTelefonico.estudiante**.
- En la línea 41 se agrega una directiva **relationship** que permita representar la relación desde la clase Estudiantes con sus números telefónicos.
- Indicar la importancia de la propiedad **back_populates** que permite referenciar los nombres de los atributos complementarios, indicando la relación entre las clases.
    - NumeroTelefonico. estudiante.
    - Estudiante.numerostelefonicos.
- Con la línea 44 se crean las tablas en la base de datos; recuerde que para el ejemplo la conexión a la base de datos Sqlite está representada en la variable cadena_base_datos que se la importa del módulo configuración.py (Ver Figura 30).

Figura 30

Tablas generadas a través de la relación expuesta entre las clases



Nota. Elizalde, R., 2025.

El ingreso de información o registros a las tablas se lo describe en las siguientes líneas:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

# se importa la clase(s) del·
# archivo genera_tablas
from genera_tablas import Estudiante, NumeroTelefonico

# se importa información del archivo configuracion
from configuracion import cadena_base_datos
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)

Session = sessionmaker(bind=engine)
session = Session()

# se crea un objetos de tipo Estudiante·
```

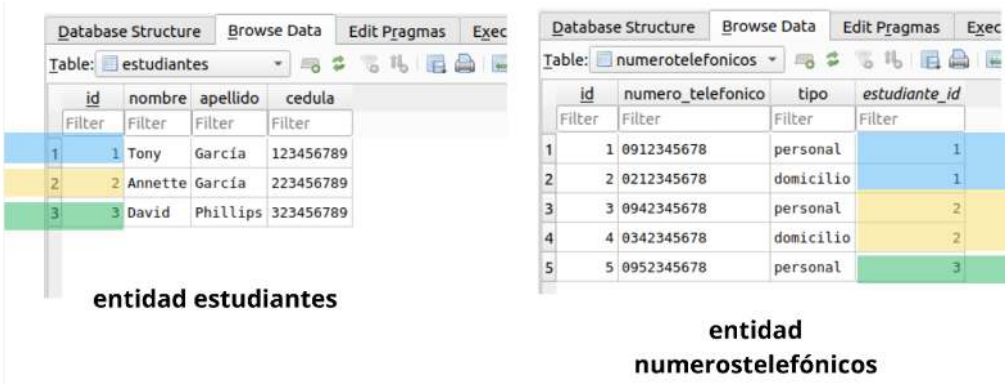
```
estudiante1 = Estudiante(nombre="Tony", apellido="García", \
 cedula="123456789")
estudiante2 = Estudiante(nombre="Annette", apellido="García", \
 cedula="223456789")
estudiante3 = Estudiante(nombre="David", apellido="Phillips", \
 cedula="323456789")

 # Se crean objeto de tipo NumeroTelefonico
 # con sus propiedades
 # numero_telefonico
 # tipo
 # adicional a cada objeto se le agrega
 # un objeto de tipo Estudiante para el·
 # atributo estudiante
 # que representa la relación
 #
 telefono1 = NumeroTelefonico(numero_telefonico="0912345678", \
 tipo="personal", \
 estudiante=estudiante1)
 telefono2 = NumeroTelefonico(numero_telefonico="0212345678", \
 tipo="domicilio",·
 estudiante=estudiante1)
 telefono3 = NumeroTelefonico(numero_telefonico="0942345678", \
 tipo="personal",·
 estudiante=estudiante2)
 telefono4 = NumeroTelefonico(numero_telefonico="0342345678", \
 tipo="domicilio",·
 estudiante=estudiante2)
 telefono5 = NumeroTelefonico(numero_telefonico="0952345678", \
 tipo="personal",·
 estudiante=estudiante3)



 # se agrega los objetos
 # a la sesión
 # a la espera de un commit
 # para agregar un registro a la base de·
 # datos
 session.add(estudiante1)
 session.add(estudiante2)
```

```
session.add(estudiante3)
session.add(telefono1)
session.add(telefono2)

# se confirma las transacciones
session.commit()
```

En la base de datos ya se puede verificar la información ingresada y debidamente relacionada (Figura 31).

Figura 31

Registros en la base de datos, luego de ejecutar los procesos de inserción respectivos para la tabla estudiantes y numerostelefonicos



Nota. Elizalde, R., 2025.

La consulta de información se describe a continuación:

- Obtener un listado de todos los registros de las tablas estudiantes y numerotelefonicos. Comentario: no está muy claro si se ha usado el infinitivo “obtener” como parte de una instrucción. Por esta razón, se ha decidido mantenerlo y no cambiarlo a la segunda persona del singular.

  Registros de las tablas estudiantes y números telefónicos

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import and_ # se importa el operador and

```

```
5   # se importa la clase(s) del·
6   # archivo genera_tablas
7   from genera_tablas import Estudiante, NumeroTelefonico
8
9   # se importa información del archivo configuracion
10  from configuracion import cadena_base_datos
11  # se genera enlace al gestor de base de
12  # datos
13  # para el ejemplo se usa la base de datos
14  # sqlite
15  engine = create_engine(cadena_base_datos)
16
17
18 Session = sessionmaker(bind=engine)
19 session = Session()
20
21 # Obtener todos los registros de·
22 # la entidad estudiantes (clase Estudiante)·
23 estudiantes = session.query(Estudiante).all()
24
25 # Se recorre la lista a través de un ciclo
26 # repetitivo for en python
27 print("Presentación de Estudiantes")
28 for s in estudiantes:
29  print("%s" % (s))
30  print(" ")
31
32 # Obtener todos los registros de·
33 # la entidad numerotelefonicos (clase NumeroTelefonico)·
34 numeros_telefonicos = session.query(NumeroTelefonico).all()
35
36 # Se recorre la lista a través de un ciclo
37 # repetitivo for en python
38
39 print("Presentación de Números Telefónicos")
40 for s in numeros_telefonicos:
41  print("%s" % (s))
42  print(" ")
43
```

Respuesta:

```
Presentación de Estudiantes
Estudiante: nombre=Tony apellido=García cedula=123456789

__________
Estudiante: nombre=Annette apellido=García cedula=223456789

__________
Estudiante: nombre=David apellido=Phillips cedula=323456789

__________
Presentación de Números Telefónicos
Número Telefónico 0912345678

__________
Número Telefónico 0212345678

__________
Número Telefónico 0942345678

__________
Número Telefónico 0342345678

__________
Número Telefónico 0952345678
```

- Obtener un listado de todos los registros de la tabla estudiantes; por cada registro (objeto tipo Estudiante), presentar sus números telefónicos.

```
1   from sqlalchemy import create_engine
2   from sqlalchemy.orm import sessionmaker
3   from sqlalchemy import and_ # se importa el operador and
4
5   # se importa la clase(s) del·
6   # archivo genera_tablas
7   from genera_tablas import Estudiante, NumeroTelefonico
8
9   # se importa información del archivo configuracion
10  from configuracion import cadena_base_datos
11  # se genera enlace al gestor de base de
12  # datos
13  # para el ejemplo se usa la base de datos
14  # sqlite
15  engine = create_engine(cadena_base_datos)
16
17
18  Session = sessionmaker(bind=engine)
19  session = Session()
20
```

```
# Obtener todos los registros de·
# la entidad estudiantes (clase Estudiante)·
estudiantes = session.query(Estudiante).all()

# Se recorre la lista a través de un ciclo
# repetitivo for en python
print("Presentación de Estudiantes")
for s in estudiantes:
print("%s" % (s))
# desde cada objeto de la lista
# estudiantes
# se puede acceder a sus número telefónicos
# haciendo uso de la relación
# Importante
# s.numerostelefonicos es un lista de
# número telefónicos
# Se hace uso de un ciclo repetitivo para
# la presentación
for t in s.numerostelefonicos:
print("\t%s" % (t) )
print(" ")
```

Respuesta:

```
Presentación de Estudiantes
Estudiante: nombre=Tony apellido=García cedula=123456789
Número Telefónico 0912345678
Número Telefónico 0212345678

__________
Estudiante: nombre=Annette apellido=García cedula=223456789
Número Telefónico 0942345678
Número Telefónico 0342345678

__________
Estudiante: nombre=David apellido=Phillips cedula=323456789
Número Telefónico 0952345678
```

- Generar:
  - Obtener un listado de todos los registros de la tabla estudiantes, que tengan al menos un número telefónico con la cadena "091";

- Obtener un listado de todos los registros de la tabla numerostelefonicos, que tengan en su atributo numero_ telefonico la cadena “091” y
- Obtener un listado de todos los registros de la tabla estudiantes, que tengan al menos un número telefónico con la cadena “091” (opción 2)

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import and_ # se importa el operador and

# se importa la clase(s) del·
# archivo genera_tablas
from genera_tablas import Estudiante, NumeroTelefonico

# se importa información del archivo configuracion
from configuracion import cadena_base_datos
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)


Session = sessionmaker(bind=engine)
session = Session()

# Obtener un listado de todos los registros·
# de la tabla estudiantes, que tengan al menos·
# un número telefónico con la cadena “091”

# para la solución se hace uso del método·
# join aplicado a query

estudiantes = session.query(Estudiante).join(NumeroTelefonico).
filter(NumeroTelefonico.numero_telefonico.like(“091%”)).all()

print(“Consulta 1 “)
for e in estudiantes:·
print(e)·
for t in e.numerostelefonicos:
print(t)·

```

```
36  # obtener un listado de todos los registros·
37  # de la tabla numerostelefonicos, que tengan·
38  # en su atributo numero_telefonico la·
39  # cadena "091".
40
41 numeros_telefonicos = session.query(NumeroTelefonico).
filter(NumeroTelefonico.numero_telefonico.like("091%")).all()·
42
43 print("Consulta 2 ")
44 for t in numeros_telefonicos:
45  print("%s - %s" % (t, t.estudiante))
46
47 # Obtener un listado de todos los registros·
48 # de la tabla estudiantes y numerostelefonicos, que tengan al menos·
49 # un número telefónico con la cadena "091"
50
51 # para la solución se hace uso del método·
52 # join aplicado a query
53 # en el query se ubican las dos entidades involucradas
54 #·
55
56 estudiantes = session.query(Estudiante, NumeroTelefonico).
join(NumeroTelefonico).filter(NumeroTelefonico.numero_telefonico.like("
091%")). all()
57
58 print("Consulta 3 ")
59
60  for registro in estudiantes:·
61  # el registro continen·
62  # dos valores en un tupla
63  # posición 0 el estudiante
64  # posición 1 el número telefónico
65  # que cumplen con la condición
66  print(registro[0])
67  print(registro[1])
```

Consultas:

```
Consulta 1
Estudiante: nombre=Tony apellido=García cedula=123456789
Número Telefónico 0912345678
```

```
Número Telefónico 0212345678
Consulta 2
Número Telefónico 0912345678 - Estudiante: nombre=Tony apellido=García
cedula=123456789
Consulta 3
Estudiante: nombre=Tony apellido=García cedula=123456789
Número Telefónico 0912345678
```

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 3.5_1

Finalmente, usted puede revisar los siguientes enlaces para analizar ejemplos del manejo de información a través de:

- QueryAPI de SqlAlchemy
- Querying with Joins en SqlAlchemy.

Para enriquecer su conocimiento, realice las actividades que se presentan a continuación:

Actividades de aprendizaje recomendadas

1. El repositorio streamlit-sqlalchemy presenta un ejemplo académico que integra la biblioteca SQLAlchemy para la definición de modelos relacionales en Python con la herramienta Streamlit, permitiendo generar interfaces gráficas para listar información. El propósito del proyecto es brindar a los estudiantes un ejemplo práctico para comprender la estructura de bases de datos relacionales, la definición de relaciones entre entidades mediante claves foráneas, y la interacción con datos a través de una interfaz *web* amigable.
2. Realice la autoevaluación 3, donde se presentan un conjunto de preguntas en función de las temáticas descritas en la unidad 3. *Acceso a base de datos relacionales mediante Object Relational Mapper (ORM)*. Para ello, se solicita revisar:

    - La unidad 3 de la presente guía.
    - Documento: Learning sqlAlchemy.

La autoevaluación le permitirá identificar el nivel de aprendizaje alcanzado y los ítems que se deben volver a revisar para lograr los resultados de aprendizaje propuestos. Además, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Autoevaluación 3

Instrucción: lea detenidamente cada uno de los enunciados de las preguntas y seleccione la o las respuestas correctas.

1. **De acuerdo al concepto de un ORM, ¿cuál de las siguientes afirmaciones son correctas?**

   a. Pasar objetos de clases realizadas en lenguajes de programación a registros en base de datos relacionales.
   b. Pasar objetos de clases realizadas en lenguajes de programación a registros en archivos planos como TXT y CSV.
   c. Pasar objetos de clases abstractas realizadas en lenguajes de programación a registros en base de datos relacionales.

2. **¿Cuál de las siguientes afirmaciones es verdadera, respecto a los ORM?**

   a. Procesos de desarrollo lentos.
   b. Con el uso de ORM se puede cambiar de motor de base de datos en cualquier momento.
   c. ORM no impide la ejecución de inyecciones SQL.

3. **Identifique un ORM usado bajo el lenguaje de programación PHP.**

   a. Ebean.
   b. Doctrine.
   c. SqlAlchemy.

4. **¿Cuál de las siguientes opciones es correcta para la instalación de la librería SqlAlchemy en un entorno local?**

   a. Pip install SqlAlchemy.
   b. Pip SqlAlchemy.
   c. Python install SqlAlchemy.

5. **¿Cuál de las siguientes sentencias permite conectarse a una base de datos SQLite haciendo uso de la librería SqlAlchemy?**

   a. Engine = create_engine('sqlite:///demobase.db').
   b. Engine = createEngine('sqlite:///demobase.db').
   c. Engine = create('sqlite:///demobase.db').

6. **Identifique la sentencia que permite crear las tablas en la base de datos en función de las clases de Python previamente creadas, haciendo uso de la librería SqlAlchemy.**

   a. from sqlalchemy.ext.declarative import declarative_base engine = create_engine(<<cadena_base_datos>>)

      Base = declarative_base() Base.metadata.create.all(engine)
   b. from sqlalchemy.ext.declarative import declarative_base engine = create_engine(<<cadena_base_datos>>)

      Base = declarative_base() Base.metadata.create_all(engine)
   c. from sqlalchemy.ext.declarative import declarative_base engine = create_engine(<<cadena_base_datos>>)

      Base = declarative_base() Base.metadata.create_all()

7. **Revise las siguientes líneas de código, identifique la salida que se generaría al ejecutar el archivo consulta_datos.py**.

```
# genera_tablas.py

from sqlalchemy import create_engine

from sqlalchemy.ext.declarative import declarative_base
```

```
from sqlalchemy.orm import sessionmaker, relationship

from sqlalchemy import Column, Integer, String,

ForeignKey # se genera enlace al gestor de base de

# datos

# para el ejemplo se usa la base de datos

# sqlite

engine = create_engine('sqlite:///demobase3.db')

Base = declarative_base()

class Hospital(Base):

  _tablename_= 'estudiantes'

  id = Column(Integer,

  primary_key=True) nombre =

  Column(String) numero_camas =

  Column(Integer) numero_pisos =

  Column(Integer)

  def_repr_(self):

     return "Hospital: %s - camas: %d - pisos: %d" % (

        self.nombre,

        self.numero_camas,

        self.numero_pisos)

Base.metadata.create_all(engine)
```

```
# genera_datos.py

from sqlalchemy import create_engine

from sqlalchemy.orm import sessionmaker

# se importa la clase(s) del
```

```
# archivo genera_tablas

from genera_tablas import Hospital

# se genera enlace al gestor de base de

# datos

# para el ejemplo se usa la base de datos

# sqlite

engine = create_engine('sqlite:///demobase3.db')

Session = sessionmaker(bind=engine)

session = Session()

h1 = Hospital(nombre="Isidro Ayora", numero_camas=100, \
numero_pisos=10)

h2 = Hospital(nombre="Andrade Marín", numero_camas=200, \

numero_pisos=20)

h3 = Hospital(nombre="Machala", numero_camas=80, \
numero_pisos=8)

h4 = Hospital(nombre="Luis Vernaza", numero_camas=80, \
numero_pisos=10)

# se agrega los objetos # a la sesión

# a la espera de un commit

# para agregar un registro a la base de # datos

session.add(h1)

session.add(h2)

session.add(h3)

session.add(h4)

# se confirma las transacciones

session.commit()
```

```
consulta_datos.py

from sqlalchemy import create_engine

from sqlalchemy.orm import sessionmaker

from sqlalchemy import and_ # se importa el operador and

# se importa la clase(s) del

# archivo genera_tablas

from genera_tablas import Hospital

# se genera enlace al gestor de base de

# datos

# para el ejemplo se usa la base de datos

# sqlite

engine = create_engine('sqlite:///demobase3.db')

Session = sessionmaker(bind=engine)

session = Session()

# Obtener todos los registros de

# la entidad estudiantes (clase Estudiante)

datos =
session.query(Hospital).filter(Hospital.numero_camas==80).all()

# Se recorre la lista a través de un ciclo

# repetitivo for en python

print("Presentación de Hospitales")

for s in datos:

   print("%s" % (s))

   print(" ")
```

Respuestas

a. Presentación de Hospitales

Hospital: Machala - camas: 80 - pisos: 8

Hospital: Luis Vernaza - camas: 80 - pisos: 8.

b. Presentación de Hospitales

Hospital: Isidro Ayora - camas: 80 - pisos: 8

Hospital: Luis Vernaza - camas: 80 - pisos: 10.

c. Presentación de Hospitales

Hospital: Machala - camas: 80 - pisos: 8

Hospital: Luis Vernaza - camas: 80 - pisos: 10.

8. **Revise las siguientes líneas de código, identifique la salida que se generaría al ejecutar el archivo consulta_datos.py**

```
# genera_tablas.py

from sqlalchemy import create_engine

from sqlalchemy.ext.declarative import declarative_base

from sqlalchemy.orm import sessionmaker, relationship

from sqlalchemy import Column, Integer, String,

ForeignKey # se genera enlace al gestor de base de

# datos

# para el ejemplo se usa la base de datos

# sqlite

engine = create_engine('sqlite:///demobase3.db')

Base = declarative_base()

class Hospital(Base):
```

```
    _tablename_= 'estudiantes'
    id = Column(Integer, primary_key=True)
    nombre = Column(String)
    numero_camas = Column(Integer)
    numero_pisos = Column(Integer)
def   repr  (self):
return "Hospital: %s - camas: %d - pisos: %d" % (
    self.nombre,
    self.numero_camas,
    self.numero_pisos)
Base.metadata.create_all(engine)
```

```
# genera_datos.py
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
# se importa la clase(s) del
# archivo genera_tablas
from genera_tablas import Hospital
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine('sqlite:///demobase3.db')
Session = sessionmaker(bind=engine)
session = Session()
```

```
h1 = Hospital(nombre="Isidro Ayora", numero_camas=100, \
numero_pisos=10)

h2 = Hospital(nombre="Andrade Marín", numero_camas=200, \
numero_pisos=20)

h3 = Hospital(nombre="Machala", numero_camas=80, \
numero_pisos=8)

h4 = Hospital(nombre="Luis Vernaza", numero_camas=80, \
numero_pisos=10)

# se agrega los objetos # a la sesión

# a la espera de un commit

# para agregar un registro a la base de # datos

session.add(h1)

session.add(h2)

session.add(h3)

session.add(h4)

# se confirma las transacciones

session.commit()
```

```
consulta_datos.py

from sqlalchemy import create_engine

from sqlalchemy.orm import sessionmaker

from sqlalchemy import and_ # se importa el operador and

# se importa la clase(s) del

# archivo genera_tablas

from genera_tablas import Hospital

# se genera enlace al gestor de base de # datos

# para el ejemplo se usa la base de datos # sqlite

engine = create_engine('sqlite:///demobase3.db')
```

```
Session = sessionmaker(bind=engine)

session = Session()

# Obtener todos los registros de

datos =
session.query(Hospital).order_by(Hospital.numero_pisos).all()

# Se recorre la lista a través de un ciclo

# repetitivo for en python

print("Presentación de Hospitales")

for s in datos:

   print("%s" % (s))

   print(" ")
```

Respuestas

a. Presentación de Hospitales

Hospital: Andrade Marín - camas: 200 - pisos: 20

Hospital: Isidro Ayora - camas: 100 - pisos: 10

Hospital: Luis Vernaza - camas: 80 - pisos: 10

Hospital: Machala - camas: 80 - pisos: 8.

b. Presentación de Hospitales

Hospital: Isidro Ayora - camas: 100 - pisos: 10

Hospital: Andrade Marín - camas: 200 - pisos: 20

Hospital: Machala - camas: 80 - pisos: 8

Hospital: Luis Vernaza - camas: 80 - pisos: 10.

c. Presentación de Hospitales

Hospital: Machala - camas: 80 - pisos: 8

Hospital: Isidro Ayora - camas: 100 - pisos: 10

Hospital: Luis Vernaza - camas: 80 - pisos: 10

Hospital: Andrade Marín - camas: 200 - pisos: 20.

9. **Revise las siguientes líneas de código; identifique la salida que se generaría al ejecutar el archivo consulta_datos.py**

```
# genera_tablas.py

from sqlalchemy import create_engine

from sqlalchemy.ext.declarative import declarative_base

from sqlalchemy.orm import sessionmaker, relationship

from sqlalchemy import Column, Integer, String,

ForeignKey # se genera enlace al gestor de base de

# datos

# para el ejemplo se usa la base de datos

# sqlite

engine = create_engine('sqlite:///demobase3.db')

Base = declarative_base()

class Hospital(Base):

   _tablename_= 'estudiantes'

   id = Column(Integer, primary_key=True)

   nombre = Column(String)

   numero_camas = Column(Integer)

   numero_pisos = Column(Integer)

def   repr  (self):

return "Hospital: %s - camas: %d - pisos: %d" % (
```

```
   self.nombre,
   self.numero_camas,
   self.numero_pisos)
Base.metadata.create_all(engine)
```

```
# genera_datos.py
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
# se importa la clase(s) del
# archivo genera_tablas
from genera_tablas import Hospital
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine('sqlite:///demobase3.db')
Session = sessionmaker(bind=engine)
session = Session()
h1 = Hospital(nombre="Isidro Ayora", numero_camas=100, \
numero_pisos=10)
h2 = Hospital(nombre="Andrade Marín", numero_camas=200, \
numero_pisos=20)
h3 = Hospital(nombre="Machala", numero_camas=80, \
numero_pisos=8)
h4 = Hospital(nombre="Pedro Ayora", numero_camas=80, \
numero_pisos=10)
# se agrega los objetos # a la sesión
```

```
# a la espera de un commit
# para agregar un registro a la base de
# datos
session.add(h1)
session.add(h2)
session.add(h3)
session.add(h4)
# se confirma las transacciones
session.commit()
```

```
consulta_datos.py
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import and_ # se importa el operador and
# se importa la clase(s) del
# archivo genera_tablas
from genera_tablas import Hospital
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine('sqlite:///demobase3.db')
Session = sessionmaker(bind=engine)
session = Session()
# Obtener todos los registros de
```

```
datos =
session.query(Hospital).filter(Hospital.nombre.like("%Ay%")).al
l()

# Se recorre la lista a través de un ciclo

 # repetitivo for en python

print("Presentación de Hospitales")

for s in datos:

   print("%s" % (s))

   print(" ")
```

Respuestas

a. Presentación de Hospitales

Hospital: Andrade Marín - camas: 200 - pisos: 20

Hospital: Isidro Ayora - camas: 100 - pisos: 10.

b. Presentación de Hospitales

Hospital: Isidro Ayora - camas: 100 - pisos: 10

Hospital: Pedro Ayora - camas: 80 - pisos: 10.

c. Presentación de Hospitales

Hospital: Machala - camas: 80 - pisos: 8

Hospital: Isidro Ayora - camas: 100 - pisos: 10.

10. **¿Cuál de las sentencias siguientes permitiría identificar a un atributo de una clase como clave foránea?**

a. persona_id = Column(Integer, ForeignKey('persona.id'))
b. persona_id = ForeignKe(Integer, Column('persona.id'))
c. persona_id = Column(Integer, ForeignKey())

Ir al solucionario

Resultado de aprendizaje 1 y 2:

- Describe la diferencia entre SaaS y una aplicación Web tradicional.
- Discute el impacto de la implementación de estándares y/o atributos de calidad en aplicaciones web

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 7

Actividades finales del bimestre

En la semana 7 de estudio se recomienda repasar los contenidos del primer bimestre, prestando especial atención a los temas que hayan generado dudas. Asimismo, se sugiere volver a realizar los ejercicios propuestos y responder nuevamente las autoevaluaciones, con el fin de consolidar el aprendizaje y fortalecer las competencias desarrolladas. Para ello, complete las siguientes actividades recomendadas.

Actividades de aprendizaje recomendadas

1. Esta actividad de aprendizaje se centra en el análisis y uso de un repositorio Vite + JQuery diseñado para entender conceptos clave del desarrollo *web* moderno. Se usa AJAX para realizar solicitudes asíncronas al servidor, permitiendo la actualización de contenido sin recargar la página. Se consume información de conjuntos de datos en formato JSON, que simula la interacción con una API, y se integra con la librería Datatable de JQuery, la cual facilita la visualización, paginación y filtrado.
2. El repositorio denominado Ejemplo de uso de la SqlAlchemy; es un proyecto académico en Python cuyo propósito es ejemplificar la creación de entidades, el ingreso y la consulta de información utilizando SQLAlchemy. Para su uso, se requiere Python y la instalación de SQLAlchemy (pip install SQLAlchemy). El proyecto se

puede obtener haciendo un fork y luego clonando el repositorio en un entorno local. Incluye archivos para crear las entidades, agregar información y múltiples *scripts* como para hacer consultas que demuestran ejemplos para la obtención de todos los registros, filtrado por atributos relacionados y navegación de relaciones entre entidades.

3. Realice nuevamente las autoevaluaciones 1, 2 y 3, donde se explica y ejemplifica la generación de ejemplos con HTML, CSS, JavaScript/ JQuery; uso de base datos no relacional a través de CouchDB y uso de bases de datos relacionales a través de ORM con SqlAlchemy. Además, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 8

Actividades finales del bimestre

La semana 8 es un período de tiempo importante destinado para que usted, estimado estudiante, pueda reafirmar temáticas del primer bimestre de estudio que aún están pendientes de comprender. Esto se realiza con el objetivo de prepararse para rendir la evaluación presencial del bimestre.

Por ello se sugiere el repaso de las siguientes unidades mediante la generación de resúmenes, cuadros sinópticos, mapas conceptuales, entre otros.

- Unidad 1. Conceptos generales de desarrollo de plataformas *web*.
- Unidad 2. Programación del lado del cliente.
- Unidad 3: Acceso a base de datos relacionales mediante *Object Relational Mapper* (ORM).

Además, es importante realizar las siguientes acciones:

- Revisar los recursos de aprendizaje recomendados en el plan docente de la asignatura.
- Plantear ejercicios y problemáticas similares a las expuestas en los contenidos de la guía.
- Revisar las preguntas que conforman los cuestionarios de repaso del bimestre planteados en el Entorno Virtual de Aprendizaje.

¿Estamos listos? Avancemos…

Segundo bimestre

Resultado de aprendizaje 3:

Aplica estándares web en el desarrollo de aplicaciones.

El resultado de aprendizaje ayuda a comprender el desarrollo de aplicaciones web desde el punto de vista de lado del servidor mediante dos lenguajes de programación: PHP y Python. Además, se estudiará el uso del patrón modelo-vista-controlador con el desarrollo de aplicaciones a través del framework de ambiente web Django.

Contenidos, recursos y actividades de aprendizaje recomendadas

Recuerde revisar de manera paralela los contenidos con las actividades de aprendizaje recomendadas y actividades de aprendizaje evaluadas.

Semana 9

Unidad 4. Programación en el servidor *web*

Estimados estudiantes, para el inicio del segundo bimestre vamos a revisar la información referente a la temática Programación en el servidor *web*; es por tal razón que se analizan algunas características del lenguaje de alto nivel PHP y su ejemplificación a través del uso de un servidor *web*.

4.1. Desarrollo de aplicaciones con lenguaje PHP

Un servidor *web*, almacena los archivos de un sitio web (como documentos HTML, hojas de estilo CSS, *scripts*, JavaScript, imágenes, etc.), y los entrega a los navegadores de los usuarios cuando estos los soliciten. Actúa como un intermediario entre el navegador del usuario y los archivos del sitio,

procesando las peticiones HTTP y enviando las respuestas correspondientes. Es una parte fundamental en la arquitectura de cualquier aplicación web, ya que permite que los sitios sean accesibles a través de *Internet*.

Existen varias formas de tener un servidor *web* funcionando en un entorno local.

1. **Opción 1**

   XAMPP es un paquete de *software* de código abierto y multiplataforma que integra varios componentes esenciales para el desarrollo de aplicaciones *web* locales. Su nombre es un acrónimo de:

   - X (cualquier sistema operativo).
   - Apache (servidor *web*).
   - MariaDB (base de datos, anteriormente MySQL).
   - PHP (lenguaje de programación del lado del servidor) y.
   - Perl (otro lenguaje de *scripting*).

   La principal ventaja de XAMPP es que proporciona un entorno de servidor *web* completamente funcional y preconfigurado en una única instalación, lo que simplifica el proceso de configuración para los desarrolladores.

   Además de los componentes principales, XAMPP incluye otras herramientas útiles: phpMyAdmin: una interfaz web para la administración de bases de datos MySQL/MariaDB, que simplifica la creación de tablas, la inserción de datos y la ejecución de consultas; Servidor FTP (FileZilla): paquetes de XAMPP incluyen un servidor FTP para facilitar la transferencia de archivos entre el entorno local y servidores remotos, y, un servidor de correo *(Mercury Mail Transport System*) que permite simular el envío de correos electrónicos desde aplicaciones web locales para pruebas.

2. **Opción 2**

Instalar las herramientas por separado en cada computador personal. En primer lugar, instalar PHP ; luego el gestor de base de datos, ejemplo MariaDB, y finalmente, con relación a los servidores, puede elegir entre Apache y Nginx. Se deja el proceso de instalación de Apache para que sea replicado en su entorno local.

3. **Opción 3**

   Se comparte el proceso de instalación a través del uso de Docker Compose, funcional para cualquier sistema operativo, que debe tener previamente instalado Docker.

Se detallan a continuación algunos ejemplos básicos, con relación al desarrollo de aplicaciones con lenguaje PHP.

Ejemplo 1

Verificación del funcionamiento del servidor con lenguaje PHP. Se recuerda que todo el código de PHP debe ir entre las etiquetas:

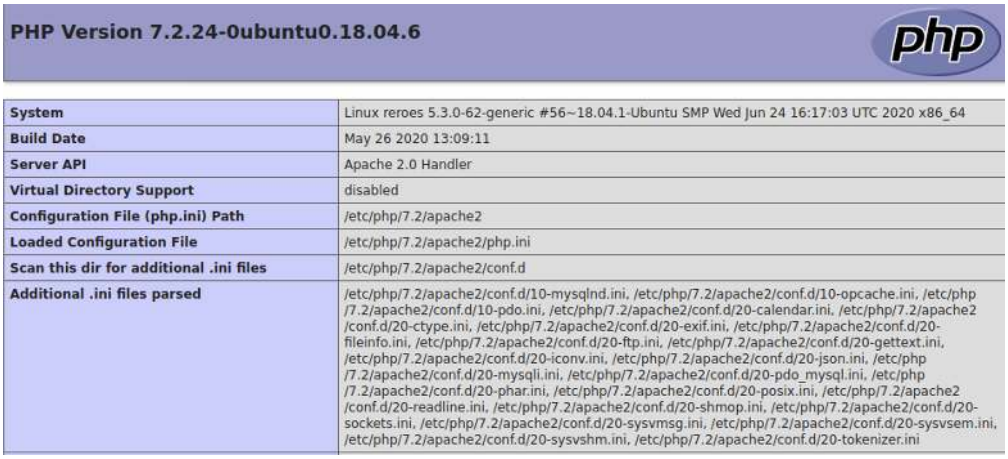
- <?php …… ?>

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width" />
<title>Verificación funcionamiento PHP</title>
</head>
<body>
<?php
····
phpinfo();
····
?>
</body>
</html>
```

La salida debe ser como se muestra a continuación figura 32:

Figura 32

Verificación del funcionamiento del servidor con lenguaje PHP



PHP Version 7.2.24-0ubuntu0.18.04.6

System	Linux reroes 5.3.0-62-generic #56~18.04.1-Ubuntu SMP Wed Jun 24 16:17:03 UTC 2020 x86_64
Build Date	May 26 2020 13:09:11
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini

Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 4.1_1

Ejemplo 2

Generar un ejemplo que haga uso de:

- Concatenación de cadenas a través de sprintf
- Uso de funciones
- Ciclos repetitivos y contadores
- Hojas de estilo

Se presenta el siguiente código:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width" />
<link rel="stylesheet" href="css/estilos.css" type="text/css" />
<title>Uso básico de PHP</title>
</head>
```

```
<body>

<table>
<tr>
<td>Multiplicado</td>
<td>Símbolo</td>
<td>Multiplicador</td>
<td>Resultado</td>
</tr>
<?php
// función en PHP
function operacionMultiplicacion($num1, $num2){
return $num1 * $num2;
}
$tabla = 5;
$inicio = 1;
echo '<h3>Tabla del '. $tabla, "</h3>";
// uso de ciclo repetitivo
while($inicio <= 12){
$formato = "<tr>
<td>%d</td><td>%s</td><td>%d</td><td class='colorcelda'>%d</td>
</tr>";
// concatenar cadenas a través de sprintf
echo sprintf($formato, $tabla, "*", $inicio,·
operacionMultiplicacion($tabla, $inicio));·
// contador
$inicio++;
}
····
?>
</table>
</body>
</html>
```

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 4.1_2

El resultado es el expuesto en la Figura 33.

Figura 33

Uso básico de lenguaje PHP

Tabla del 5

Multiplicado	Símbolo	Multiplicador	Resultado
5	*	1	5
5	*	2	10
5	*	3	15
5	*	4	20
5	*	5	25
5	*	6	30
5	*	7	35
5	*	8	40
5	*	9	45
5	*	10	50
5	*	11	55
5	*	12	60

Nota. Elizalde, R., 2025.

Una parte importante en los lenguajes de alto nivel para generar aplicaciones web es el uso de formularios. Se deja un ejemplo de cómo se puede trabajar con formularios HTML en PHP.

Ejemplo 3

En el siguiente ejemplo se evidencia el envío de parámetros entre dos páginas en lenguaje PHP, haciendo uso de un formulario mediante el método POST. Se solicita a través de un formulario nombre, apellido, la tabla de multiplicar y el límite de la tabla; con base en los valores ingresados, presentar el nombre, apellido y la tabla generada.

Se utiliza el siguiente código:

Página uno.php

```
uno.php
1     <!DOCTYPE html>
2    <html>
3    <head>
4    <meta charset="utf-8" />
5    <meta name="viewport" content="width=device-width" />
6    <link rel="stylesheet" href="css/estilos.css" type="text/css" />
7    <title>Uso básico de PHP con formularios</title>
8    </head>
9    <body>
10   <h2>Ingrese datos para generación</h2>
11   <div class="miformulario">
12   <form action="dos.php" method="post">
13   <label for="nombre">Ingrese su nombre</label>
14   <input type="text" name="nombre" id="nombre" required/>
15   <br/>
16   <br/>
17   <label for="apellido">Ingrese su apellido</label>
18   <input type="text" name="apellido" id="apellido" required/>
19   <br/>
20   <br/>
21   <label for="tabla">Ingrese la tabla a generar</label>
22   <input type="number" name="tabla" id="tabla" required/>
23   <br/>
24   <br/>
25   <label for="limite">Ingrese el límite</label>
26   <input type="number" name="limite" id="limite" required/>
27   <br/>
28   <br/>
29   <input type="submit" name="enviar" id="enviar" value="Enviar" />
```

```
30    </form>
31    </div>
32    </body>
33    </html>
```

Página dos.php

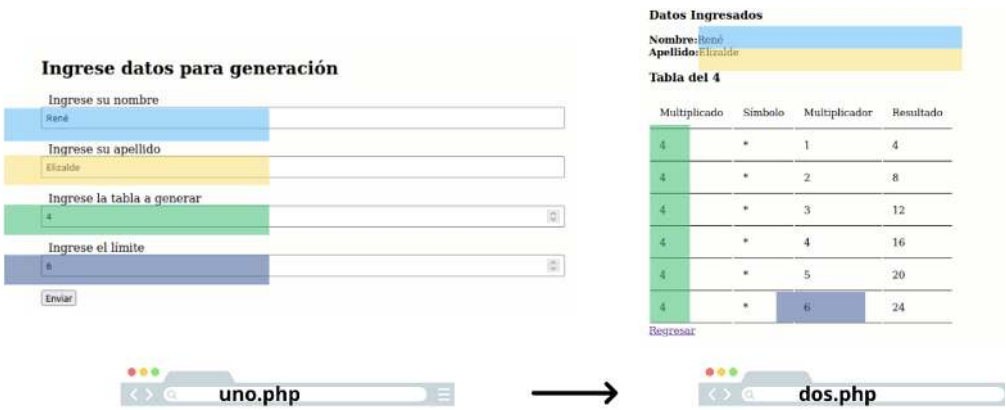
```
dos.php
1     <!DOCTYPE html>
2     <html>
3     <head>
4     <meta charset="utf-8" />
5     <meta name="viewport" content="width=device-width" />
6     <link rel="stylesheet" href="css/estilos.css" type="text/css" />
7      <title>Uso básico de PHP</title>
8      </head>
9      <body>
10     <h3>Datos Ingresados</h3>
11     <?php
12     $nombre = $_REQUEST["nombre"];
13     $apellido = $_REQUEST["apellido"];
14     $datos = "<b>Nombre:</b>%s<br/><b>Apellido:</b>%s<br/>";·
15     echo sprintf($datos, $nombre, $apellido) ; 16
17     ?>
18     <table>
19     <tr>
20     <td>Multiplicado</td>
21     <td>Símbolo</td>
22     <td>Multiplicador</td>
23     <td>Resultado</td>
24     </tr>
25     <?php
26     // función en PHP
27     function operacionMultiplicacion($num1, $num2){
28     return $num1 * $num2;
29     }
30     $tabla = intval($_REQUEST['tabla']);
31     $inicio = 1;
32     $limiteTabla = intval($_REQUEST['limite']);·
33     echo '<h3>Tabla del '. $tabla, "</h3>";
34     // uso de ciclo repetitivo
```

```
35    while($inicio <= $limiteTabla){
36    $formato = "<tr>
37    <td>%d</td><td>%s</td><td>%d</td><td class='colorcelda'>%d</td>
38    </tr>";
39    // concatenar cadenas a través de sprintf
40    echo sprintf($formato, $tabla, "*", $inicio,·
41    operacionMultiplicacion($tabla, $inicio));·
42    // contador
43    $inicio++;
44    }
45 ····
46    ?>
47    </table>
48    <a href="uno.php">Regresar</a>
49    </body>
50    </html>
```

La salida al ejecutar es la expuesta en la Figura 34:

Figura 34

Interacción entre páginas con lenguaje PHP



Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 4.1_3

Semana 10

Unidad 4. Programación en el servidor *web*

4.2. Acceso a base de datos relacional desde una aplicación en PHP

Para la comprensión de este apartado, estimado estudiante, es fundamental que usted ya posea un dominio claro de los conceptos básicos de bases de datos relacionales, adquiridos en ciclos académicos previos. Esta sección tiene como objetivo recordar las características esenciales y el proceso de acceso a bases de datos relacionales desde una aplicación web sencilla, así como el uso del lenguaje SQL para la creación de tablas y la selección de información.

Ejemplo 1

Generar una entidad en base de datos para describir vehículos con las siguientes características: marca, placa, año de fabricación, tipo (particular o público) y propietario. Se usa el gestor de base de datos MariaDB.;

```
# Creación de la base de datos create database proyectoejemplo;
# Creación de la tabla vehículos create table `vehiculos` (
`id` int(10) NOT NULL auto_increment primary key,
`marca` varchar(100) not null,
`placa` varchar(100) not null unique,
`anio_fabricacion` int(100) not null,
`tipo` varchar(100) not null,
`propietario` varchar(100) not null
);
# Ingreso de datos a la tabla vehículos insert into `vehiculos`(
`marca`,
`placa`,
`anio_fabricacion`,
`tipo`,
`propietario`
) values (
“chevrolet”,
```

```
"LCH0101",
1990,
"particular",
"Patricio Rojas"
);
```

A través de una aplicación web generada a través del lenguaje PHP:

- Acceder a la base de datos y listar los vehículos ingresados.
- Agregar nuevos vehículos.

Base de datos.php

```
basedatos.php
1 <?php
2   // datos para enlace la base de datos
3   $servidor = "localhost";
4   $usuario = "susuario";
5   $password = "suclave";
6   $base_datos = "proyectoejemplo";
7   $conectaBD = new mysqli($servidor, $usuario, $password,
$base_datos);
8   ?>
```

Archivo index.php

```
index.php
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width" />
6   <link rel="stylesheet" href="css/estilos.css" type="text/css" />
7   <title>Uso básico de PHP</title>
8   </head>
9   <body>
10  <h3>Listado de Vehículos</h3>
11  <table>
12  <tr>
13  <td>Propietario</td>
14  <td>Placa</td>
```

```
15  <td>Marca</td>
16  <td>Año Fabricación</td>
17  <td>Tipo</td>
18  </tr>
19  <?php
20  include("basedatos.php");
21  function convertirMayuscula($dato){
22  // función que permite·
23  // pasar una cadena a mayúscula·
24  return strtoupper($dato);
25  }
26  // se realizar la consulta a la base de datos
27  $consultaBD = $conectaBD -> query("Select * from vehiculos");
28  while($registro = $consultaBD -> fetch_array(MYSQLI_ASSOC)){
29  $formato = "<tr>
30  <td>%s</td><td>%s</td><td>%s</td><td>%d</td><td>%s</td>
31  </tr>";
32  // se agrega una fila a la tabla HTML
33  echo sprintf($formato,
convertirMayuscula($registro['propietario']),·
34  convertirMayuscula($registro['placa']),
35  convertirMayuscula($registro['marca']),
36  convertirMayuscula($registro['anio_fabricacion']),·
37  convertirMayuscula($registro['tipo']));
38  }
39  ?>
40  </table>
41  <a href="nuevo.php">Nuevo vehículo</a>
42  </body>
43  </html>
```

Archivo nuevo.php

```
nuevo.php
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width" />
6   <link rel="stylesheet" href="css/estilos.css" type="text/css" />
7   <title>Uso básico de PHP con formularios</title>
```

```
8   </head>
9   <body>
10  <?php
11  if($_GET['error']){
12  echo "<h5>". $_GET['error'] ."</h5>";
13  }
14 ?>
15  <h2>Ingrese datos vehículo</h2>
16  <div class="miformulario">
17  <form action="nuevo_agrega.php" method="post">
18  <label for="marca">Ingrese marca de vehículo</label>
19  <input type="text" name="marca" id="marca" required/>
20  <br/>
21  <br/>
22  <label for="placa">Ingrese placa del vehículo</label>
23  <input type="text" name="placa" id="placa" required/>
24  <br/>
25  <br/>
26  <label for="anio_fabricacion">Ingrese año de fabricación de
vehículo</ label>
27  <input type="number" name="anio_fabricacion" id="anio_fabricacion"
required/>
28  <br/>
29  <br/>
30  <label for="tipo">Ingrese tipo de vehículo</label>
31  <input type="text" name="tipo" id="tipo" required/>
32  <br/>
33  <br/>
34  <label for="propietario">Ingrese nombres de propietario</label>
35  <input type="text" name="propietario" id="propietario" required/>
36  <br/>
37  <br/>
38
39  <input type="submit" name="enviar" id="enviar" value="Enviar" />
40  </form>
41  </div>
42  </body>
43  </html>
```

Archivo nuevo_agrega.php

nuevo_agrega.php

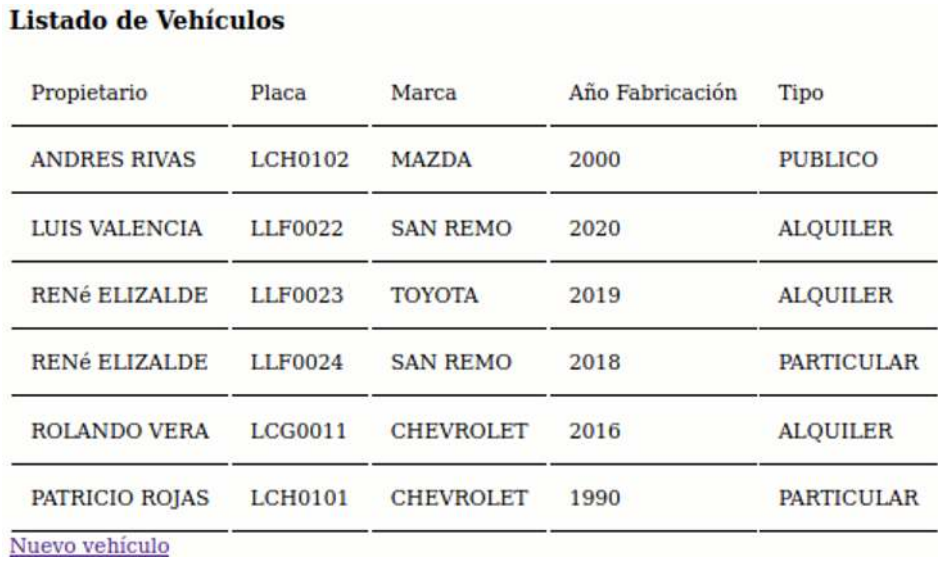
```
<?php
   // proceso para guardar un nuevo registro a la base de datos
   include("basedatos.php");
   $marca = $_REQUEST['marca'];
   $placa = $_REQUEST['placa'];
   $tipo = $_REQUEST['tipo'];
   $propietario = $_REQUEST['propietario'];
   $anio = intval($_REQUEST['anio_fabricacion']);
   $cadena = "insert into `vehiculos`( `marca`,
`placa`,`anio_fabricacion`,·
  `tipo`, `propietario`) values ('%s', '%s', %d, '%s', '%s');";·
  $cadena = sprintf($cadena, $marca, $placa, $anio, $tipo,
$propietario);
  echo $cadena;
  $consultaBD = $conectaBD -> query($cadena);
 ......
  if($consultaBD){
  // si hay existo con la inserción,·
  // se realiza un redirect a index.php
  header("Location: index.php");
  }else{
  // si exite un error
  // se captura el error
  // se hace un redirect a nuevo.php
  // además se envía el mensaje de error
  // como parámetro
  $mensaje = $conectaBD -> error;
  header("Location: nuevo.php?error=".$mensaje);
  }
?>
```

Luego de ejecutar los archivos anteriores, se realizaron tareas como:

- Listar la información de la tabla vehículos a través del archivo index.php (ver figura 35).

Figura 35

Listado de registros de la tabla vehículos



Listado de Vehículos

Propietario	Placa	Marca	Año Fabricación	Tipo
ANDRES RIVAS	LCH0102	MAZDA	2000	PUBLICO
LUIS VALENCIA	LLF0022	SAN REMO	2020	ALQUILER
RENé ELIZALDE	LLF0023	TOYOTA	2019	ALQUILER
RENé ELIZALDE	LLF0024	SAN REMO	2018	PARTICULAR
ROLANDO VERA	LCG0011	CHEVROLET	2016	ALQUILER
PATRICIO ROJAS	LCH0101	CHEVROLET	1990	PARTICULAR

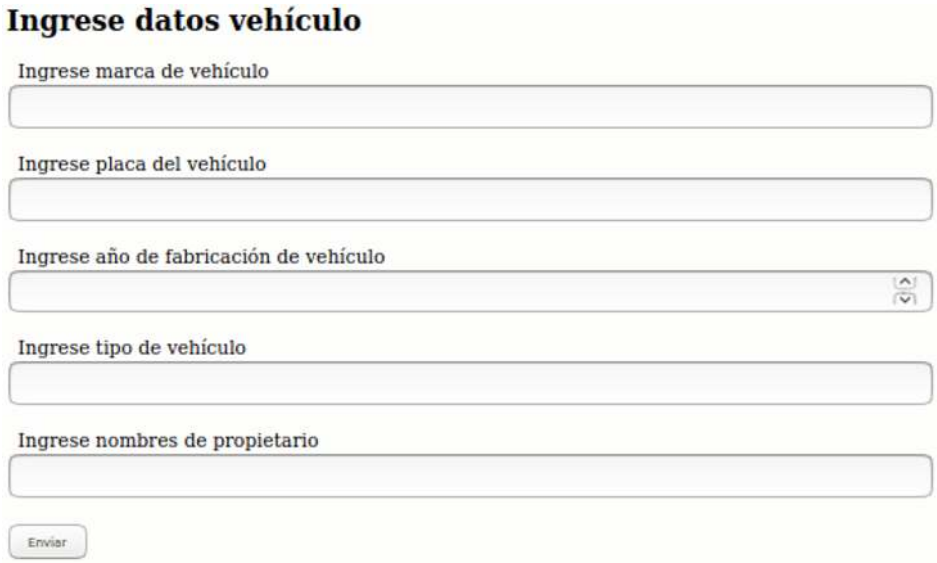
Nuevo vehículo

Nota. Elizalde, R., 2025.

- Generar nuevos registros a través de los archivos: nuevo.php y nuevo_agrega.php; ver Figura 36.

Figura 36

Formulario de ingreso de datos para un nuevo vehículo



Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 4.2_1

Para enriquecer su conocimiento, realice las actividades que se presentan a continuación:

Actividades de aprendizaje recomendadas

1. El repositorio Ejemplo PHP y Base de Datos es un proyecto académico diseñado para ejemplificar el proceso de ingreso y listado de datos utilizando PHP y una base de datos MySQL/MariaDB. Para su uso, es necesario clonar el repositorio y crear una base de datos llamada “mobiliaria” en MySQL/MariaDB, importando las entidades y registros del archivo data.sql. Luego, la carpeta ejemplo-php-base-datos debe agregarse a un servidor *web* local como Xampp, Apache o Nginx, y el archivo basedatos.php debe ser modificado con las credenciales correctas de usuario y clave para la conexión a la base

de datos. Una vez configurado, el proyecto puede ser accedido a través de un navegador ingresando a index.php. El repositorio permite entender el ciclo de vida de los datos en una aplicación de forma tradicional

2. Realice la autoevaluación 4, donde se presentan un conjunto de preguntas con los temas abordados en la unidad 4, 'Programación en el servidor *web*'. Para prepararse, se recomienda revisar los temas y subtemas de la unidad 4 de la presente guía

   Esta actividad le permitirá identificar su nivel de comprensión actual y detectar aquellos aspectos que requieren un estudio más profundo para alcanzar los resultados de aprendizaje esperados. Asimismo, podrá proponer interrogantes o ejercicios similares para consolidar lo analizado y estudiado.

Autoevaluación 4

Instrucción: lea detenidamente cada uno de los enunciados de las preguntas y seleccione la o las respuestas correctas.

1. **Identifique el nombre de dos servidores *web*.**

   a. WampServer, XAMPP.
   b. Apache, WampServer.
   c. Nginx, Apache.

2. **Indique el código que, al ejecutarse en un servidor web, presente en el navegador la cadena: verificando conocimientos.**

   a.
```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width" />
```

```
    <title>Verificación funcionamiento PHP</title>
</head>
<body>
    echo "Verificando conocimientos" ;
</body>
</html>
```

b.
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Verificación funcionamiento PHP</title>
</head>
<body>
<%
    echo "Verificando conocimientos" ;
%>
</body>
</html>
```

c.
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
```

```
    <meta name="viewport" content="width=device-width" />
    <title>Verificación funcionamiento PHP</title>
  </head>
  <body>
  <?php
    echo "Verificando conocimientos" ;
  ?>
  </body>
</html>
```

3. **Dados los siguientes archivos, ¿cuál es la salida que se puede visualizar a través de la ejecución en un navegador?**

```
index.php
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />
  <link rel="stylesheet" href="css/estilos.css" type="text/
css" />
  <title>Uso básico de PHP</title>
 </head>
 <body>
 <table>
 <tr>
```

```
   <td>Nro 1</td>
   <td>Operador</td>
   <td>Nro 2</td>
   <td>Resultado</td>
   while($inicio <= 5){
   $formato = "<tr>
   <td>%d</td><td>%s</td><td>%d</td><td class='celda'>%d</td>
   </tr>";
   echo sprintf($formato, $inicio, "+", $tabla,
operacion($tabla, $inicio));
   $inicio++;
   }
   ?>
 </table>
 </body>
</html>
```

```
 css/estilos.css
th, td {
   padding: 15px;
   border-bottom: 1px solid black;
}
 tr:hover {
   background-color: #258EAD;
}
/* manejo de las características de las clases*/
```

```
.celda{
background: #9999FF; color: #000000;
font-family: Arial, Helvetica, sans-serif; font-size: 14px;
font-weight: bold;
}
```

Opciones de respuesta:

a

Tabla del 3

Nro. 1	Operador	Nro. 2	Resultado
3	+	1	4
3	+	2	5
3	+	3	6
3	+	4	7
3	+	5	8

b

Tabla del 3

Nro. 1	Operador	Nro. 2	Resultado
3	+	1	6
3	+	2	6
3	+	3	6
3	+	4	6
3	+	5	6

c

Tabla del 3

Nro. 1	Operador	Nro. 2	Resultado
1	+	3	4
2	+	3	5
3	+	3	6
4	+	3	7
5	+	3	8

4. **Dado el siguiente archivo, ¿cuál es la salida que se puede visualizar a través de la ejecución en un navegador?**

```
<!DOCTYPE html>

<html>

 <head>
```

```
   <meta charset=”utf-8” />
   <meta name=”viewport” content=”width=device-width” />
   <link rel=”stylesheet” href=”css/estilos.css” type=”text/
css” />
   <title>Uso básico de PHP</title>
 </head>
 <body>
 <?php
   function operacion($num1){ return $num1 * 10;
   }
 echo “<hr/>”;
 for ($i = 0; $i <= 6; $i++) {
 if ($i>=3) {
 $cadena = “Valor generado: <b>%s</b><br/>”;
 }else{
 $cadena = “Valor generado: %s<br/>”;
 }
 echo sprintf($cadena, operacion($i));
 }
 echo “<hr/>”;
 ?>
 </table>
 </body>
</html>
```

a. Valor generado: **0**

Valor generado: **10**

Valor generado: **20**

Valor generado: 30

Valor generado: 40

Valor generado: 50

Valor generado: 60

b. Valor generado: 0

Valor generado: 10

Valor generado: 20

Valor generado: **30**

Valor generado: **40**

Valor generado: **50**

Valor generado: **60**

c. Valor generado: **0**

Valor generado: **10**

Valor generado: **20**

Valor generado: **30**

Valor generado: **40**

Valor generado: **50**

Valor generado: **60**

5. **Dados los siguientes archivos, ¿cuál es la salida que se puede visualizar a través de la ejecución en un navegador?**

```
<!DOCTYPE html>
 <html>
  <head>
   <meta charset=”utf-8” />
   <meta name=”viewport” content=”width=device-width” />
   <link rel=”stylesheet” href=”css/estilos.css” type=”text/
css” />
   <title>Uso básico de PHP</title>
  </head>
  <body>
  <?php
   function operacion($num1){ return $num1 * 2;
  }
   function operacion2($num1){ return $num1 - 1;
  }
 echo “<hr/>”;
 for ($i = 0; $i <= 6; $i++) {
   $cadena = “<span class=’presentacion’>Valor generado:
%s<span/><br/>”;
   echo sprintf($cadena, operacion2(operacion($i)));
  }
  echo “<hr/>”;
  ?>
  </table>
  </body>
```

```
</html>
```

a. VALOR GENERADO: -1

VALOR GENERADO: 1

VALOR GENERADO: 3

VALOR GENERADO: 5

VALOR GENERADO: 7

VALOR GENERADO: 9

VALOR GENERADO: 11

b. Valor generado: -1

Valor generado: 1

Valor generado: 3

Valor generado: 5

Valor generado: 7

Valor generado: 9

Valor generado: 11

c. VALOR GENERADO: 1

VALOR GENERADO: 3

VALOR GENERADO: 5

VALOR GENERADO: 7

VALOR GENERADO: 9

VALOR GENERADO: 11

VALOR GENERADO: 13

6. **Dado el siguiente archivo denominado uno.php, identifique el archivo dos.php correcto.**

```
uno.php

<!DOCTYPE html>

<html>

  <head>

    <meta charset=”utf-8” />

    <meta name=”viewport” content=”width=device-width” />

    <link rel=”stylesheet” href=”css/estilos.css” type=”text/
css” />

    <title>Uso básico de PHP con formularios</title>

  </head>

  <body>

  <h2>Ingrese datos para generación</h2>

  <div class=”miformulario”>

    <form action=”dos.php” method=”post”>

    <label for=”ciudad”>Ingrese su ciudad</label>

    <input type=”text” name=”ciudad” id=”ciudad” required/>

    <br/>

    <br/>

    <label for=”provincia”>Ingrese su provincia</label>

    <input type=”text” name=”provincia” id=”provincia”
required/>

    <br/>

    <br/>
```

```
    <input type="submit" name="enviar" id="enviar"
value="Enviar" />
   </form>
   </div>
  </body>
</html>
```

Repuestas

a. dos.php

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width" />
   <link rel="stylesheet" href="css/estilos.css"
type="text/css" />
   <title>Uso básico de PHP</title>
  </head>
  <body>
  <h3>Datos Ingresados</h3>
  <?php
   $ciudad = ["ciudad"];
   $provincia = ["provincia"];
   $datos = "<b>Provincia:</b>%s<br/><b>Ciudad:</b>%s<br/
>";
```

```
    echo sprintf($datos, $ciudad, $provincia) ;
    ?>
    </body>
</html>
```

b. dos.php

```
<!DOCTYPE html>
<html>
 <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <link rel="stylesheet" href="css/estilos.css"
type="text/css" />
    <title>Uso básico de PHP</title>
 </head>
 <body>
    <h3>Datos Ingresados</h3>
    <?php
    $ciudad = $_REQUEST[ciudad];
    $provincia = $_REQUEST[provincia];
    $datos = "<b>Provincia:</b>%s<br/><b>Ciudad:</b>%s<br/
>";
    echo sprintf($datos, $ciudad, $provincia) ;
    ?>
```

```
    </body>

</html>
```

c. dos.php

```
<!DOCTYPE html>

<html>

 <head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width" />

    <link rel="stylesheet" href="css/estilos.css"
type="text/css" />

    <title>Uso básico de PHP</title>

 </head>

 <body>

 <h3>Datos Ingresados</h3>

 <?php

    $ciudad = $_REQUEST["ciudad"];

    $provincia = $_REQUEST["provincia"];

    $datos = "<b>Provincia:</b>%s<br/><b>Ciudad:</b>%s<br/
>";

    echo sprintf($datos, $ciudad, $provincia) ;

 ?>

 </body>

</html>
```

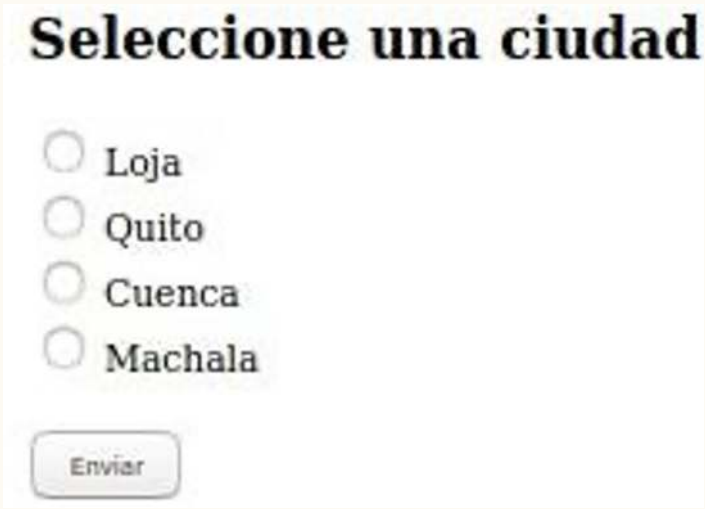
7. **¿Cuál de las siguientes sentencias permite crear una conexión a base de datos MariaDB o MySQL?**

   a. $conectaBD = new MysqlDB($servidor, $usuario, $password, $base_datos);
   b. $conectaBD = new mysqli($servidor, $usuario, $password, $base_datos);
   c. $conectaBD = mysqli($servidor, $usuario, $password, $base_datos);

8. **¿Cuál de las siguientes sentencias permite realizar una redirect a una página llamada destino.php a la cual se envía un parámetro llamado texto?**

   a. header("Location_destino.php?texto="mensaje");
   b. header("destino.php?texto="mensaje");
   c. header("Location: destino.php?texto="mensaje");

9. **¿Cuál de los siguientes archivos genera una salida como la que se muestra en la figura?**



   a. `<!DOCTYPE html>`

```
<html>
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width" />
<title>Uso básico de PHP con formularios</title>
</head>
<body>
<h2>Seleccione una ciudad</h2>
<div class="miformulario">
<form action="dos.php" method="post">
<?php
function informacion($v1){
$variable = '<input type="radio" id="%s"
name="ciudades" value="%s">
<label for="%s">%s</label>
<br/>';
$variable = sprintf($variable, $v1, $v1, $v1, $v1);
return $variable;
}
$ciudades = array('Loja', 'Quito', 'Cuenca',
'Machala'); for ($i = 0; $i < 4; $i++) {
echo informacion($ciudades[$i]);
}
```

```
    ?>
    <br/>
    <input type="submit" name="enviar" id="enviar"
value="Enviar" />
    </form>
  </div>
</body>
</html>
```

b.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Uso básico de PHP con formularios</title>
  </head>
  <body>
  <h2>Seleccione una ciudad</h2>
  <div class="miformulario">
    <form action="dos.php" method="post">
    <?php
    function informacion($v1){
    $variable = '<input type="radio" id="%s"
name="ciudades" value="%s">
```

```
    <label for="%s">%s</label>

    <br/>';

    $variable = sprintf($variable, $v1); return $variable;

    }

    $ciudades = array('Loja', 'Quito', 'Cuenca',
'Machala'); for ($i = 0; $i < 4; $i++)      {

    echo informacion($ciudades[$i]);

    }

    ?>

    <br/>

    <input type="submit" name="enviar" id="enviar"
value="Enviar" />

    </form>

  </div>

 </body>

</html>
```

c.
```
<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width" />

    <title>Uso básico de PHP con formularios</title>

  </head>
```

```
<body>
<h2>Seleccione una ciudad</h2>
<div class="miformulario">
<form action="dos.php" method="post">
<?php
function informacion($v1){
$variable = '<input type="radio" id="%s" name="ciudades" value="%s">
<label for="%s">%s</label>
<br/>';
$variable = sprintf($variable, $v1, $v1, $v1, $v1);
}
$ciudades = array('Loja', 'Quito', 'Cuenca', 'Machala');
for ($i = 0; $i < 4; $i++) {
echo informacion($ciudades[$i]);
}
?>
<br/>
<input type="submit" name="enviar" id="enviar" value="Enviar" />
</form>
</div>
</body>
```

```
</html>
```

10. **Dados los siguientes archivos, identifique la salida correcta.**

```
<!DOCTYPE html>

<html>

  <head>

    <meta charset=”utf-8” />

    <meta name=”viewport” content=”width=device-width” />

    <title>Uso básico de PHP con formularios</title>

  </head>

  <body>

  <h2>Seleccione una ciudad</h2>

  <div class=”miformulario”>

   form action=”dos.php” method=”post”>

    <label for=”fecha”>Fecha</label>

    <input type=”date” id=”fecha” name=”fecha”
max=”2020-12-31”><br><br>

    <input type=”submit” name=”enviar” id=”enviar”
value=”Enviar” />

   </form>

   </div>

 </body>

</html>
```

```
<!DOCTYPE html>

<html>

  <head>
```

```
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <link rel="stylesheet" href="css/estilos.css" type="text/
css" />
    <title>Uso básico de PHP</title>
  </head>
  <body>
  <h1>Presentar Datos</h1>
  <?php
   function mifecha($v){
   return date('Ymd',strtotime($v));
  }
  $fecha = $_REQUEST["fecha"];
  $fecha2 = mifecha($fecha);
  $datos = "<h2>Fecha</h2><h3>%s</h3>";
  echo sprintf($datos, $fecha2) ;
  ?>
 </body>
</html>
```

a. Presentar Datos

Fecha

07.25.20

b. Presentar Datos

Fecha

25,7,2020

c. Presentar Datos

Fecha

20200725

Ir al solucionario

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 11

Unidad 5. Uso de *frameworks* de ambiente *web*

5.1. Modelo-vista-controlador base de los *frameworks*

Según **Pérez Martínez (2015)**, el patrón Modelo-Vista-Controlador (MVC) es un paradigma fundamental en el desarrollo de aplicaciones que estructura el trabajo en tres niveles distintos para una mejor organización y separación de responsabilidades. En este esquema, el **Modelo** se encarga de definir cómo se manipulan los datos de la aplicación, especificando tanto su tipo como la lógica de negocio asociada a su gestión y almacenamiento en la capa de persistencia. La **Vista,** por su parte, se refiere al componente visual de la aplicación, determinando la forma en que esta interactuará con el usuario y presentará la información. Finalmente, el **Controlador** actúa como el intermediario principal, gestionando las acciones iniciadas por el usuario y comunicándolas de manera efectiva a los niveles del Modelo y la Vista para coordinar la respuesta adecuada de la aplicación.

Además, Gutiérrez González y López Goytia (2016), señalan algunas ideas con relación al uso de *framework* de ambiente web, que se detallan a continuación.

- Cuando se habla de un *framework* se abarcan temas como: lenguaje de programación, librerías, bibliotecas, herramientas y metodologías de desarrollo.

- Se resalta el uso del patrón Modelo-Vista-Controlador que permita reutilización de código, facilitar los procesos de desarrollo y mejorar el mantenimiento a futuro de las aplicaciones.
- La rapidez en los tiempos de desarrollo a través del uso de propiedades de cada *framework*.
- Se facilita la corrección de posibles errores en la construcción de la solución.
- Generación de test de desarrollo para probar las funcionalidades.
- La seguridad implícita en los *frameworks* permite generar calidad en los productos entregados.

5.2. Clasificación de *frameworks* de ambiente web según los lenguajes de programación

En el presente ítem de estudio se describen los más importantes *frameworks* de acuerdo con los lenguajes de programación. Resaltar que para la comprensión de un determinado *framework* existen dos puntos claves:

- Manejo de conceptos de: paradigma de programación de orientación a objetos, HTML, CSS y JavaScript.
- Entender el Modelo-Vista-Controlador explicado en el apartado anterior de la guía didáctica.

La lista de *frameworks* es enorme en función de los lenguajes de programación. En la siguiente infografía se listan los *frameworks* más importantes con documentación actualizada.

Frameworks y Plantillas por Lenguaje

5.3. Desarrollo de aplicaciones mediante el *framework* Django

Estimado estudiante, para el estudio del presente apartado relacionado con el *framework* Django, se hará uso del libro denominado La guía definitiva de Django – desarrolla aplicaciones *web* de forma rápida y sencilla.

5.3.1. Introducción

El capítulo Introducción a Django del libro destaca algunas características importantes del *framework* Django en el marco de la generación de aplicación web; se mencionan algunas de ellas:

- Ahorra tiempo en el desarrollo de aplicaciones.
- Genera atajos para algunas tareas frecuentes.
- Maneja el patrón Modelo-Vista-Controlador mediante su filosofía Modelo-Vista-Template.
- Existe un acoplamiento débil en el desarrollo de componentes.

La documentación oficial de Django siempre será un recurso importante para profundizar en algunas temáticas.

5.3.2. Creación de proyecto y modelado

Usted debe revisar el capítulo 2 del libro recomendado, cuya denominación es *Empezando, donde se detalla información respecto a los procesos de instalación de lenguaje Python y del framework Django*. Para la instalación de Django se recomienda hacer uso del gestor de paquetes PIP de Python.

Se puede instalar la última versión de Django, mediante el siguiente comando: pip install Django

Ejemplo 1:

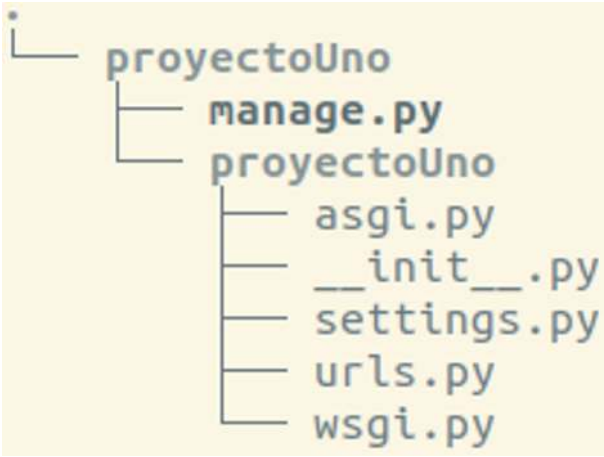
A continuación, se realiza la explicación de un ejemplo para la creación de un proyecto en Django. Se inicia el proyecto usando el siguiente comando:

- django-admin startproject proyectoUno

  Los archivos generados son los expuestos en la figura 37:

Figura 37

Archivos generados por el comando que crea un proyecto en Django



Nota. Elizalde, R., 2025.

Donde:

- **proyectoUno** se convierte en la carpeta que contiene los archivos del proyecto.
- **manage.py** es usado para interactuar a través de la línea de comandos con el proyecto; se puede crear aplicaciones, acceder a una consola interactiva, actualizar la base de datos, crear usuarios administradores para el proyecto.
- **__init__.py** es un archivo que permite indicar a Python que el directorio sea considerado como paquete.
- **settings.py,** en este archivo se tienen las configuraciones del proyecto de Django.
- **urls.py,** se ubica el conjunto de direcciones internas del proyecto de Django.
- **wsgi.py** y asgi.py son archivos usados cuando se requiera ubicar el proyecto en un servidor como Apache o Nginx.

Indicar que en el desarrollo de este ejemplo demostrativo se usará la base de datos Sqlite. En el archivo setting.py se busca la variable DATABASES, misma que indica el gestor de base de datos para usar.

```
DATABASES = {
 'default': {
   'ENGINE': 'django.db.backends.sqlite3',
   'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
 }
}
```

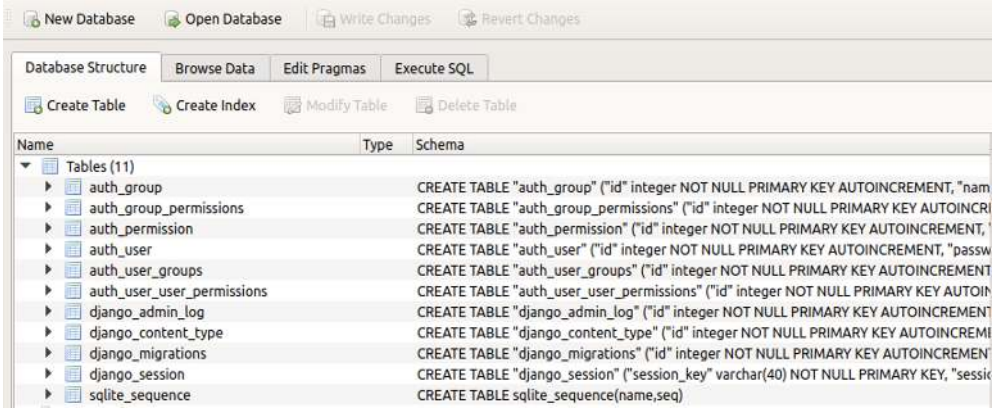
Se crean las tablas por defecto del proyecto a través del comando:

- python manage.py migrate

Las tablas creadas en la base de datos son las descritas en la Figura 38:

Figura 38

Tablas generadas por con el comando python manage.py migrate

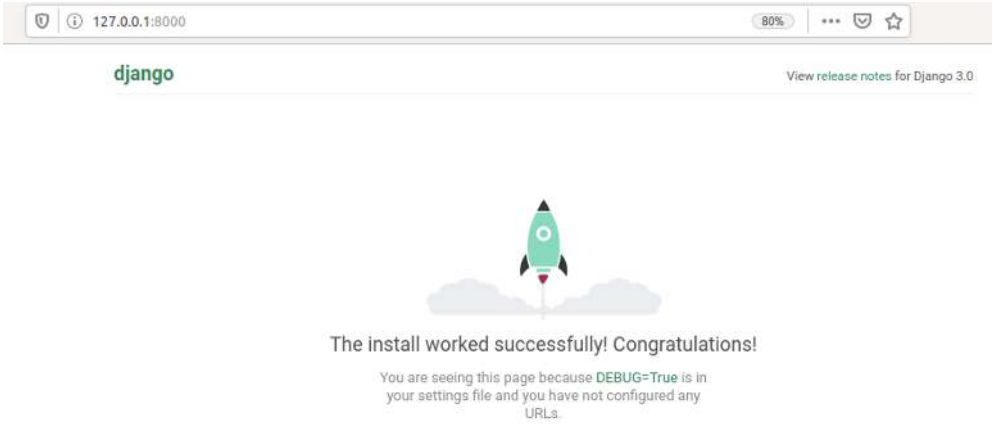


Nota. Elizalde, R., 2025.

Se hace uso del servidor ligero para revisar el funcionamiento de la aplicación; el comando es python manage.py runserver, y se muestra su ejecución en la Figura 39:

Figura 39

Levantar la aplicación a través del servidor propio de Django, usando el comando python manage.py runserver



Nota. Elizalde, R., 2025.

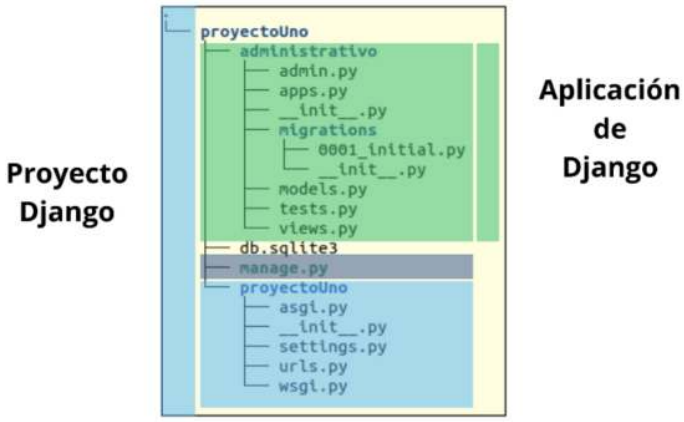
Se procede a crear una **aplicación** para el proyecto a través del comando:

- python manage.py startapp administrativo

Lo anterior agrega una nueva estructura al proyecto (Figura 40).

Figura 40

Estructura inicial de un proyecto y una aplicación en Django



Nota. Elizalde, R., 2025.

- **administrativo** es la carpeta que contiene los archivos de la aplicación.
- **admin.py** permite agregar las clases o entidades que serán puestas en el proceso de administración del proyecto
- **__init__.py,** es el archivo que permite indicar a Python que el directorio sea considerado com paquete.
- **models.py,** se ubican las clases en Python que luego se convierten tablas de la base de datos.
- **views.py** se gestiona principalmente a través de funciones.

Luego de tener generada la aplicación, se procede a ingresar el modelo en la misma; se hace uso del archivo models.py; se recuerda que el modelado se lo hace mediante clases estructuradas en lenguaje Python, de manera similar al proceso realizado en SqlAlchemy.

Clases estructuradas en lenguaje Python

```
1 from django.db import models
2
3 # Create your models here.
4
5   class Estudiante(models.Model):
```

```
6    nombre = models.CharField(max_length=30)
7    apellido = models.CharField(max_length=30)
8    cedula = models.CharField(max_length=30, unique=True)
9
10   def   str   (self):
11   return "%s %s %s" % (self.nombre,·
12   self.apellido,
13   self.cedula)
14
15   class NumeroTelefonico(models.Model):
16   telefono = models.CharField(max_length=100)
17   tipo = models.CharField(max_length=100)
18   estudiante = models.ForeignKey(Estudiante,
on_delete=models.CASCADE)
19
20   def   str   (self):
21   return "%s %s" % (self.telefono, self.tipo)
```

En el modelo generado se ha creado dos entidades Estudiante y NumeroTelefonico.

- La entidad Estudiante hereda de la clase Model y tiene los siguientes atributos:
  - nombre; de tipo cadena representado en Django por campo CharField, se le asigna un máximo de 30 caracteres.
  - apellido; de tipo cadena representado en Django por campo CharField, se le asigna un máximo de 30 caracteres.
  - cedula; de tipo cadena representado en Django por campo CharField, se le asigna un máximo de 30 caracteres y a través del atributo unique se indica que los valores deben ser únicos.
- La entidad NumeroTelefonico hereda de la clase Model y tiene los siguientes atributos:
  - telefono; de tipo cadena representado en Django por campo CharField se le asigna un máximo de 100 caracteres.
  - tipo; de tipo cadena representado en Django por campo CharField, se le asigna un máximo de 100 caracteres.

- estudiante, atributo de tipo clave foránea representado a través de ForeignKey. Se resalta que al atributo se le establece la entidad con cual se genera la relación (Estudiante). Cuando se genere un objeto de tipo NumeroTelefonico, al atributo estudiante se le debe asignar como valor un objeto de tipo Estudiante.

En las clases expresadas en el modelo, no se ha especificado un atributo como clave primaria - primary_key; en estos casos, Django crea automáticamente un campo AutoField de tipo entero, no es necesario agregar este atributo cuando se generan objetos.

En la documentación se detalla información referente a Modelo Field.

Es necesario agregar la aplicación en el arreglo representado por la variable INSTALLED_APPS en el archivo settings.py. Se procede a ejecutar los comandos para almacenar en archivos de control propios de Django las migraciones:

- python manage.py makemigrations administrativo

```
Opcional: se usa el comando python manage.py sqlmigrate administrativo
0001 para revisar en formato SQL el makemigration ejecutado
BEGIN;
--
--Create model Estudiante
--
CREATE TABLE "administrativo_estudiante" ("id" integer NOT NULL PRIMARY
KEY AUTOINCREMENT, "nombre" varchar(30) NOT NULL, "apellido"
varchar(30) NOT NULL, "cedula" varchar(30) NOT NULL UNIQUE);
--
--Create model NumeroTelefonico
--
CREATE TABLE "administrativo_numerotelefonico" ("id" integer NOT NULL
PRIMARY KEY AUTOINCREMENT, "telefono" varchar(100) NOT NULL, "tipo"
varchar(100) NOT NULL, "estudiante_id" integer NOT NULL REFERENCES
"administrativo_estudiante" ("id") DEFERRABLE INITIALLY DEFERRED);
CREATE INDEX "administrativo_numerotelefonico_estudiante_id_a580aa13"
ON "administrativo_numerotelefonico" ("estudiante_id");
COMMIT;
```

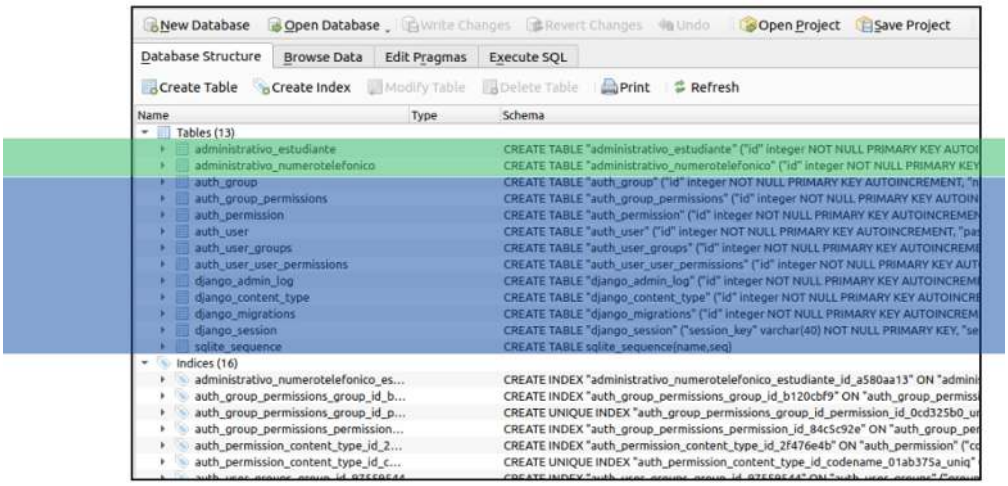
Luego, se necesita realizar los cambios en la base de datos a través del comando:

- python manage.py migrate

La base de datos tiene la apariencia expuesta en la Figura 41:

Figura 41

Modelo de la aplicación migrado a la base de datos



Nota. Elizalde, R., 2025.

Es momento de usar el shell de Django para interactuar con la base de datos a través del uso del ORM de Django mediante las clases del modelo. Para acceder se usa el comando:

- python manage.py shell

```
# La sentencias mostradas a continuación se usando el shell de Django
# Primero, se importa las clases del modelo
from administrativo.models import Estudiante, NumeroTelefonico
# Inserción de datos
# se crea un objeto de tipo Estudiante
# se hace referencia a los atributos de la clase
e1 = Estudiante(nombre=”Luisa”, apellido=”Tene”, cedula=”1100991133”)
e2 = Estudiante(nombre=”Rolando”, apellido=”Sarango”,
cedula=”1100991122”)
```

```
# se guarda el objeto en la base de datos, mediante el método save()
e1.save()
e2.save()
# se crea un objeto de tipo NumeroTelefonico
# para el atributo estudiante del objeto t, se usa el objeto de tipo
Estudiante e
t = NumeroTelefonico(telefono="09988776655", tipo="particular",
estudiante=e1) # Consulta de datos
# Seleccionar todos los datos del modelo Estudiante
estudiantes = Estudiante.objects.all()
for e in estudiantes:
print(e)
#salida:
####
# Rolando Sarango 1100991122 # Luisa Tene 1100991133
####
# Filtrar todos los estudiantes que tiene como valor en nombre:
"Rolando"
estudiantes = Estudiante.objects.filter(nombre="Rolando").all()
# Filtrar todos los estudiantes que tengan un número de teléfono con la
secuencia "8888"
# numerotelefonico es la clase asociada, se recuerda la relación
Estudiante.objects.filter(numerotelefonico telefono contains="8888")
# Filtrar todos los números telefónicos que pertenecen a estudiantes
que tengan en su apellido la secuencia "Te"
telefonos =
NumeroTelefonico.objects.filter(estudiante__apellido__contains="Te")
for t in telefonos:
print("%s - %s" % (t, t.estudiante))
#salida
##
# 09988776655 particular - Luisa Tene 1100991133
##
```

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 5.3_1

Se sugiere la revisión de la documentación oficial del framework Django en relación a manejo de relaciones entre entidades:

- Relación muchos a uno

- Relación uno a uno
- Relación muchos a muchos
- Manejo de ORM de Django

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 12

Unidad 5. Uso de *frameworks* de ambiente *web*

5.3. Desarrollo de aplicaciones mediante el *framework* Django

5.3.3. Uso de interfaz administrativa

En el capítulo 6, denominado *El sitio de administración*, del libro *La guía definitiva de Django – Desarrolla aplicaciones web de forma rápida y sencilla*, se detallan los procesos para manejar la interfaz de administración de proyectos en Django. Se solicita una lectura de dicho apartado.

Se hace uso de los metadatos de los modelos generados en las aplicaciones para crear una interfaz de usuario con prestaciones listas para ser usadas por el usuario que disponga de los permisos respectivos.

Para la activación se sugiere los siguientes pasos:

- Verificar en el archivo setting.py, en la variable INSTALLED_APPS que tiene en lista la opción: django.contrib.admin.
- Mediante el comando: python manage.py createsuperuser, se crea un superusuario del proyecto.
- Editar el archivo admin.py de la aplicación para indicar los modelos que serán accesibles para administración desde la interfaz.

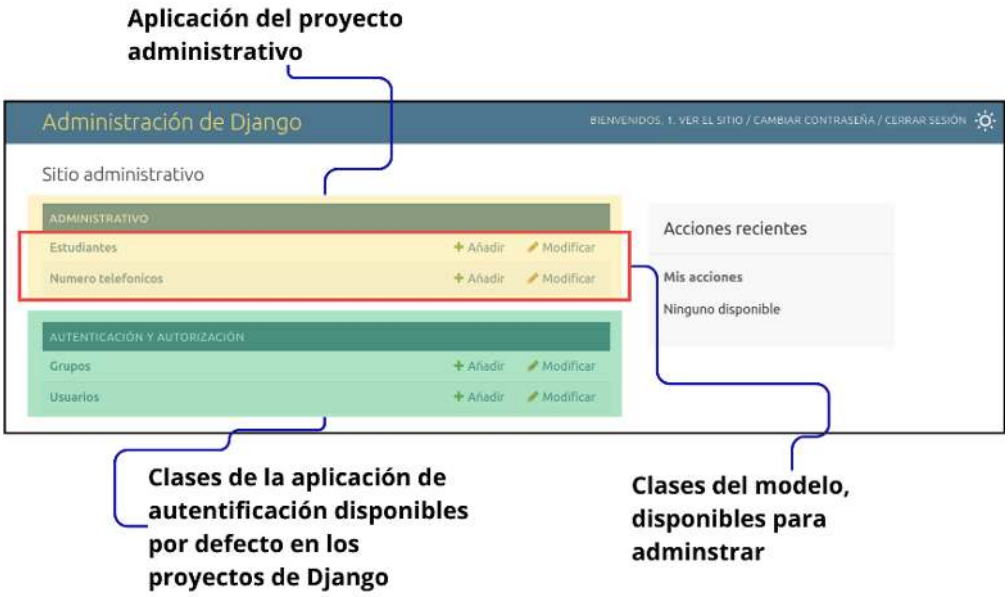
```
1 from django.contrib import admin
2
3 # Importar las clases del modelo
4 from administrativo.models import Estudiante, NumeroTelefonico
5
```

```
# Agregar la clase Estudiante para administrar desde
# interfaz de administración
admin.site.register(Estudiante)

# Agregar la clase NumeroTelefonico para administrar desde
# interfaz de administración
admin.site.register(NumeroTelefonico)
```

Se digita el comando que permite levantar el servidor propio de Django (python manage.py runserver) y se ingresa a la siguiente dirección en un navegador web (http://127.0.0.1:8000/admin/).Se solicitará el ingreso del ***Username*** y ***Password***, recordar los datos ingresados en el comando ***createsuperuser***. Se visualiza una pantalla que lista las clases o entidades disponibles en la interfaz de administrador para manipular (ver figura 42).

Figura 42

Interfaz de administración de Django

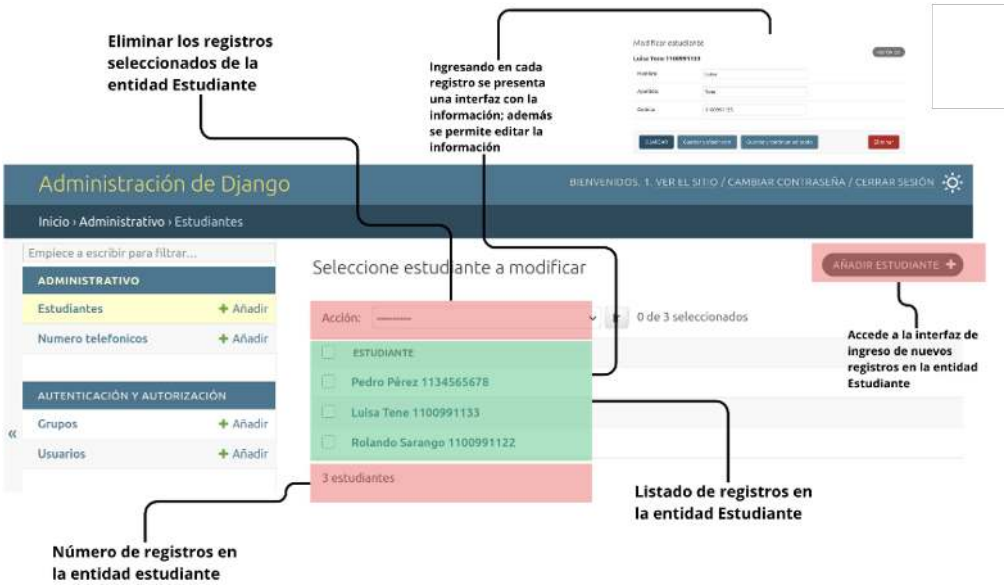


Nota. Elizalde, R., 2025.

Se puede seleccionar cualquiera de las entidades listadas para acceder a las opciones dadas por la interfaz. Si se selecciona la entidad **Estudiantes**, se visualiza una pantalla con algunas opciones (ver Figura 43).

Figura 43

Interfaz de administración de una clase específica



Nota. Elizalde, R., 2025.

Para agregar un registro a la clase o entidad **Estudiante** se tiene la siguiente pantalla ver Figura 44.

Figura 44

Ingreso de un registro a la clase o entidad Estudiante

Nota. Elizalde, R., 2025.

A continuación, se representa el proceso de ingreso de un registro de la clase NumeroTelefonico, para demostrar el potencial del componente administrativo de Django; se recuerda que la clase NumeroTelefonico requiere como datos o propiedades teléfono, tipo y estudiante; este último es de tipo Estudiante; ver Figura 45.

Figura 45

Manejo de relaciones en interfaz administrativa de Django

Nota. Elizalde, R., 2025.

A continuación, se muestran algunas opciones de personalización para la interfaz administrativa; se trabaja sobre el archivo admin.py.

- Cuando se desplieguen los registros de la clase Estudiante, se debe mostrar los registros por columnas con base a las propiedades que se especifiquen.

```
from django.contrib import admin

# Importar las clases del modelo
from administrativo.models import Estudiante, NumeroTelefonico

# Agregar la clase Estudiante para administrar desde
# interfaz de administración
# admin.site.register(Estudiante)

  # Se crea una clase que hereda
  # de ModelAdmin para el modelo
  # Estudiante
  class EstudianteAdmin(admin.ModelAdmin):
  # listado de atributos que se mostrará
  # por cada registro
  # se deja de usar la representación (str)·
  # de la clase·
  list_display = ('nombre', 'apellido', 'cedula')

  # admin.site.register se lo altera
  # el primer argumento es el modelo (Estudiante)
  # el segundo argumento la clase EstudianteAdmin
  admin.site.register(Estudiante, EstudianteAdmin)

  # Agregar la clase NumeroTelefonico para administrar desde
  # interfaz de administración
  admin.site.register(NumeroTelefonico)
```

En la Figura 46 se indica cómo se visualizan los registros con los cambios realizados.

Figura 46

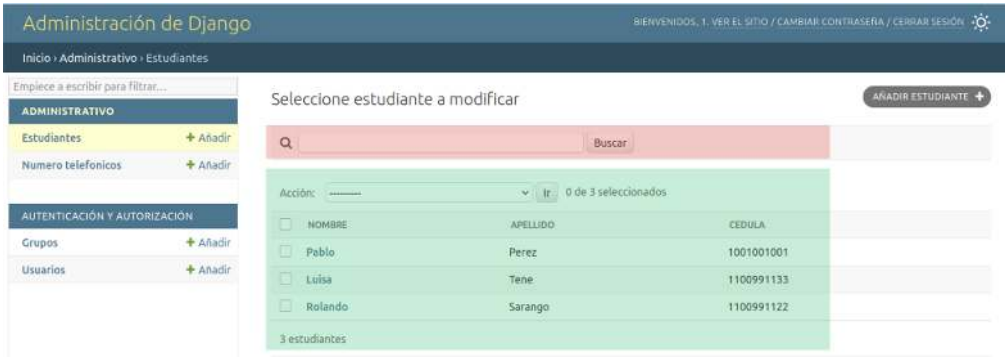
Listar los registros de la clase Estudiante mediante columnas

Nota. Elizalde, R., 2025.

- Agregar la opción de búsqueda con base a los atributos de la clase. Se agrega el atributo search_fields, al cual se da como valor una tupla de atributos que serán los campos de búsqueda para cada registro (Figura 47).
    - Para el modelo estudiante se tiene: search_fields = (“nombre”, “cedula”)

Figura 47

Agregar la opción de búsqueda al listado de un modelo en la interfaz administrativa



Nota. Elizalde, R., 2025.

- Cuando se crea un registro de tipo NumeroTelefonico, al momento de asignar el valor del atributo estudiante (clave foránea), se debe seleccionar de un listado de estudiantes a través de un ***select***. Se puede considerar otra forma de seleccionar el campo ***estudiante***. En una clase que hereda de

ModelAdmin se agrega el atributo ***raw_id_fields*** que permite acceder a una interfaz para buscar los estudiantes y seleccionar el que se desee relacionar.

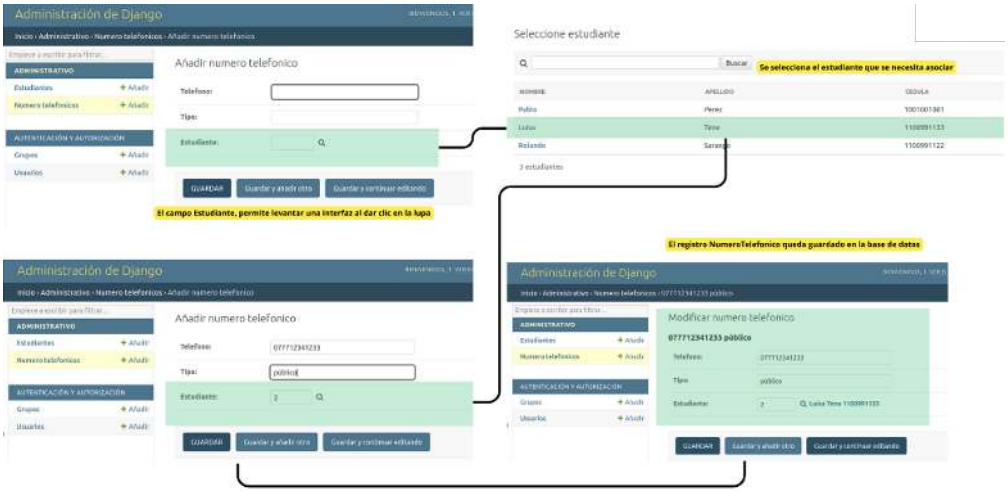
```
from django.contrib import admin

# Importar las clases del modelo
from administrativo.models import Estudiante, NumeroTelefonico

# Agregar la clase Estudiante para administrar desde
# interfaz de administración
# admin.site.register(Estudiante)

 # Se crea una clase que hereda
 # de ModelAdmin para el modelo
 # Estudiante
 class EstudianteAdmin(admin.ModelAdmin):
 # listado de atributos que se mostrará
 # por cada registro
 # se deja de usar la representación (str)·
 # de la clase·
 list_display = ('nombre', 'apellido', 'cedula')
 search_fields = ('nombre', 'cedula')

 # admin.site.register se lo altera
 # el primer argumento es el modelo (Estudiante)
 # el segundo argumento la clase EstudianteAdmin
 admin.site.register(Estudiante, EstudianteAdmin)

 # Agregar la clase NumeroTelefonico para administrar desde
 # interfaz de administración
 # admin.site.register(NumeroTelefonico)

 # Se crea una clase que hereda
 # de ModelAdmin para el modelo
 # NumeroTelefonico
 class NumeroTelefonicoAdmin(admin.ModelAdmin):
 # listado de atributos que se mostrará
 # por cada registro
 # se deja de usar la representación (str)·
 # de la clase·
```

```
38  list_display = ('telefono', 'tipo', 'estudiante')
39  # se agrega el atributo
40  # raw_id_fields que permite acceder a una interfaz
41  # para buscar los estudiantes y seleccionar el que
42  # se desee
43  raw_id_fields = ('estudiante',)
44
45 admin.site.register(NumeroTelefonico, NumeroTelefonicoAdmin)
```

Como se aprecia en la Figura 48, esta configuración habilita un buscador que facilita la selección del estudiante asociado, evitando desplegar listas extensas en el campo foráneo.

Figura 48

Uso del atributo raw_id_fields en un atributo llave foránea



Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: [Desarrollo del ejemplo 5.3.3_1]

Ahora, para reforzar lo aprendido, realice las siguientes actividades:

Actividades de aprendizaje recomendadas

1. En el repositorio "Crear entorno virtual - Pip - Django" se explica cómo configurar un entorno de desarrollo para proyectos de Python y Django. Se demuestra la creación de un directorio de proyecto, la configuración de un entorno virtual para aislar las dependencias del proyecto, y la activación/desactivación de este entorno. También se muestra el uso de Pip para instalar paquetes como Django, resaltando cómo estas dependencias y sus ejecutables se gestionan dentro del entorno virtual específico. Se concluye explicando que los pasos vistos son fundamentales antes de iniciar un proyecto con Django.
2. Realice la autoevaluación 5, donde se presentan un conjunto de preguntas en función de las temáticas descritas en la unidad 5. Uso de *frameworks de ambiente web*. Usted puede revisar los siguientes temas:

   - Ítems 5.1, 5.2, 5.3 de la presente guía.
   - Libro: La guía definitiva de Django – desarrolla aplicaciones *web* de forma rápida y sencilla.

     - Capítulo 1: Introducción a Django.
     - Capítulo 2: Empezando.
     - Capítulo 6: El sitio de administración.

   La autoevaluación le permitirá identificar el nivel de aprendizaje alcanzado y los ítems que se deben volver a revisar para lograr los resultados de aprendizaje propuestos. Además, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Autoevaluación 5

Instrucción: lea detenidamente cada uno de los enunciados de las preguntas y seleccione la o las respuestas correctas.

1. **Identifique tres temáticas que se abordan cuando se estudia y aplica un *framework*.**

   a. Lenguaje de programación, librerías y bibliotecas.
   b. Lenguaje de programación, sistema operativo y bibliotecas.
   c. Lenguaje de programación, librerías e IDE de programación.

2. **Identifique dos sistemas de plantillas usados por los *frameworks* con lenguaje de programación PHP.**

   a. Twing, Thymeleaf.
   b. Twing, Blade.
   c. Apache FreeMarker, Jinja.

3. **¿Cuál es el comando para crear un proyecto en el *framework* Django?**

   a. Django-admin startproject nombre-proyecto.
   b. Django-admin start_project nombre-proyecto.
   c. Django-admin project nombre-proyecto.

4. **Identifique el archivo generado en los proyectos de Django que permite: interactuar a través de la línea de comandos con el proyecto para crear aplicaciones, actualizar la base de datos, crear usuarios administradores para el proyecto.**

   a. Settings.py.
   b. Manage.py.
   c. Wsgi.py.

5. **Identifique el comando que permite crear y actualizar las tablas en la base de datos en un proyecto de Django.**

a. Python manage.py engine.
b. Python manage.py sql.
c. Python manage.py migrate.

6. **¿Cuál es el comando usado en Django para crear una aplicación?**

a. Python manage.py startapp nombre-app.
b. Python manage.py start administrativo.
c. Python manage.py start_app administrativo.

7. **Dado un modelo con denominación Universidad, identifique el comando que permite obtener todos los registros de la base de datos mediante el ORM de Django.**

a. Universidad.session().objects.all()
b. Universidad.objects.all()
c. Universidad.objects.filter()

8. **La entidad Universidad posee un atributo llamado dirección de tipo cadena. ¿Cuál es el comando que permite buscar todos los registros que contengan la cadena "Loja" en alguna parte del valor del atributo?**

a. Universidad.objects.filter(direccion contains="Loja").all()
b. Universidad.objects.filter(direccion.contains="Loja").all()
c. Universidad.objects.filter(direccion contains("Loja")).all()

9. **¿Cuál es el atributo que se debe agregar en una clase que hereda de admin.ModelAdmin, que permita mostrar atributos seleccionados de un modelo?**

a. Losta_display.
b. List_display.
c. Listdisplay.

10. **¿Cuál es el comando que permite crear un superusuario administrador en un proyecto en Django?**

a. Python manage.py createsuperuser.
b. Python manage.py create_superuser.
c. Python manage.py create_supe_ruser.

Ir al solucionario

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 13

Unidad 5. Uso de *frameworks* de ambiente *web*

5.3. Desarrollo de aplicaciones mediante el *framework* Django

Es momento de realizar una revisión detenida del capítulo 7. ***Procesamiento de formularios*** del libro: *La guía definitiva de Django – desarrolla aplicaciones web de forma rápida y sencilla*.

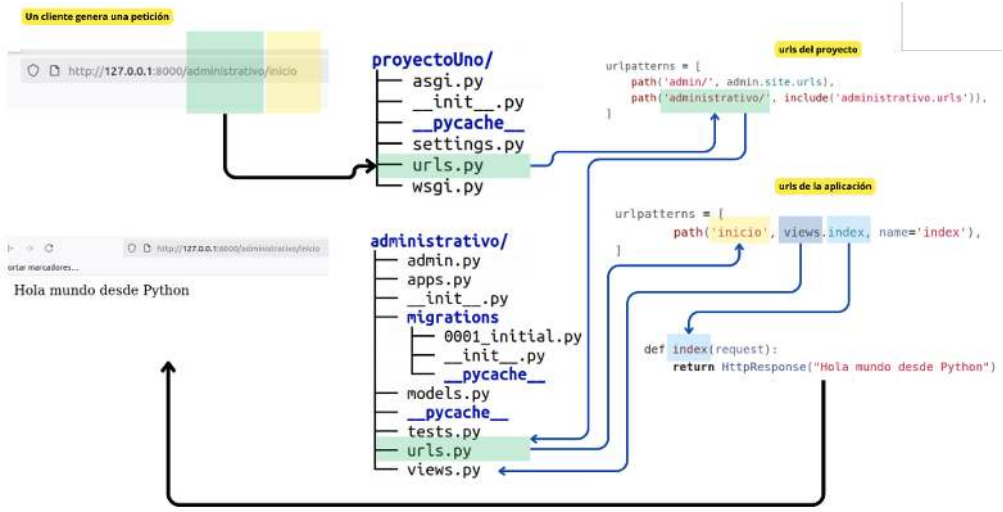
5.3.4. Manejo de vistas, templates

El manejo de vistas en los proyectos en Django se los realiza editando el archivo views.py de las aplicaciones del proyecto. Se recuerda que en este módulo se genera la lógica de la aplicación que permite acceder a la base de datos y construir una respuesta para ser visualizada en una plantilla. Las funciones que se codifican en el archivo **views.py** tienen como primer argumento un objeto de tipo HttpRequest. Se considera un estándar usar la palabra **request**; en dicho objeto se agrupa un conjunto de información acerca de la petición generada.

En la figura 49 se describe el flujo de información al momento que se realiza una petición a un proyecto/aplicación de Django.

Figura 49

Flujo de información mediante una petición a Django



Nota. Elizalde, R., 2025.

Al momento de realizar una petición a través de una URL en Django, esta busca el patrón expuesto en la URL en el archivo urls.py del proyecto; luego se pasa al archivo urls.py de la aplicación, en el ejemplo se busca que exista un path en a la aplicación con la cadena inicio; si existe, se busca la vista referenciada en el path, para el ejemplo, se busca la función llamada **index** en el archivo **views.py**; finalmente, la lógica expuesta en la función se procesa y a través del módulo HttpResponse de Django, pasa la información al navegador que generó la petición.

Las aplicaciones de Django necesitan saber el directorio donde se ubican los **templates** que son referenciados desde las funciones del archivo views.py. Para ello en el archivo settings.py se debe especificar el nombre del directorio haciendo uso de la variable TEMPLATES a través de la llave **DIRS**.

```
55 TEMPLATES = [
56  {
57  'BACKEND': 'django.template.backends.django.DjangoTemplates',
58
59  'DIRS': [os.path.join(BASE_DIR, 'templates')],
```

```
60  'APP_DIRS': True,
61  'OPTIONS': {
62  'context_processors': [
63  'django.template.context_processors.debug',
64  'django.template.context_processors.request',
65  'django.contrib.auth.context_processors.auth',
66  'django.contrib.messages.context_processors.messages',
67  ],
68  },
69  },
70 ]
```

La estructura de la aplicación **administrativo** queda de la siguiente forma. Se define el nombre del directorio como **templates**:

```
administrativo/
├── admin.py
├── apps.py
├── __init__.py
├── migrations
│   ├── 0001_initial.py
│   ├── __init__.py
│   └── __pycache__
├── models.py
├── __pycache__
├── templates
│   └── listadoEstudiantes.html
├── tests.py
├── urls.py
└── views.py
```

a. **Ejemplo 1**

A continuación, se define un proceso que permita **listar los registros del modelo Estudiante** obtenidos de la base de datos.

Archivo views.py

```
views.py
1 from django.shortcuts import render
2 from django.http import HttpResponse
```

```
from django.template import RequestContext
from django.shortcuts import render

# importar las clases de models.py
from administrativo.models import *

# Create your views here.

  def index(request):
  # return HttpResponse("Hola mundo desde Python")
  return HttpResponse("Hola mundo desde Python<br/>%s" %
(request.path))

def listadoEstudiantes(request):
  """
  Listar los registros del modelo Estudiante,
  obtenidos de la base de datos.
  """
  # a través del ORM de django se obtiene
  # los registros de la entidad; el listado obtenido
  # se lo almacena en una variable llamada
  # estudiantes
  estudiantes = Estudiante.objects.all()
  informacion_template = {'estudiantes': estudiantes,
'numero_estudiantes': len(estudiantes)}
  return render(request, 'listadoEstudiantes.html',
informacion_template)

```

Archivo listadoEstudiantes.html

```
templates/listadEstudiantes.html
   Número de estudiantes {{ numero_estudiantes }}
   <br/><br/>

{% for e in estudiantes %}

{{e}}<br/>

{% endfor %}
```

Archivo urls.py

```
urls.py de la aplicación
1 """
2   Manejo de urls para la aplicación
3   administrativo
4 """
5   from django.urls import path
6   # se importa las vistas de la aplicación
7   from . import views
8
9
10  urlpatterns = [
11  path('inicio', views.index, name='index'),
12  path('listado-estudiantes', views.listadoEstudiantes,
13  name='listadoEstudiates'),
14 ]
```

Al momento de ejecutar el servidor con los cambios realizados se obtiene una imagen como de la Figura 50.

Figura 50

Pantalla final del proceso de generación de información en views.py y utilización de templates

Nota. Elizalde, R., 2025.

Algunas observaciones de la función **listadoEstudiantes** del archivo views.py:

- **estudiantes = Estudiante.objects.all()**; a través del ORM de django se obtienen los registros de la entidad; el listado obtenido se lo almacena en una variable llamada *estudiantes*; se entiende que la variable "estudiantes" es un listado.

- **informacion_template**; es la variable tipo diccionario donde se agregará la información que estará disponible en el *template* para presentar.
- ***{'estudiantes': estudiantes, 'numero_estudiantes': len(estudiantes)}***; es el valor asignado a la variable **informacion_template**; se generan dos llaves, la primera llamada '**estudiantes**' cuyo valor específico es la variable estudiantes, la segunda es '**numero_estudiantes**' que tiene asignado el número de elementos que contienen la lista llamada **estudiantes**.
- En la parte final de la función está expresado una sentencia de retorno:
    - **return render**; la función render recibe como argumentos: el objeto request, la plantilla o template donde se desplegará la lógica desarrollada en la función y un diccionario con datos que necesitamos pasar a la plantilla.
    - **request**, el objeto que tiene los datos de la petición.
    - 'listadoEstudiantes.html', se detalla la ruta y la plantilla asociada a la función.
    - informacion_template, es un diccionario con todo el contexto que estará disponible en el template.

Ahora se detalla lo realizado en el archivo listadoEstudiantes.html, se recuerda que dicho archivo está ubicado en la carpeta **templates** dentro la aplicación **administrativo**.

En los templates o plantillas de Django se puede usar HTML, JAVASCRIPT, CSS, junto a:

- **Variables.-** son representadas de la siguiente forma **{{ nombre- de-la-variable }}**. Dentro del template evalúa si una variable está bien estructura y presenta en valor que tenga en ese momento.
- **Filtros.-** una característica que permite modificar la presentación de una variable. Se usa el carácter pipe (|) después del nombre de la variable; por ejemplo **{{ nombre-de- variable | lower}}**, presenta el valor de variable en minúscula. Existen filtros establecidos en el ***framework*** y existe la posibilidad de crear filtros personalizados.

- **Tags.-** la estructura que sigue un tag es **{% nombre-tag %}.** Según la funcionalidad, para algunos tags se necesita establecer el inicio y fin:

```
{% nombre-tag %}
{% end-nombre-tag %}
```

A través de los tags se puede implementar ciclos repetitivos mediante for, condicionales **if, elif y else**.

En el archivo listadoEstudiantes.html se expresa lo siguiente:

- En la línea 1, se usa la representación de una variable mediante ***{{ numero_estudiantes }}***
- En la línea 5, se inicia un ciclo repetitivo mediante el tag **{% for e in estudiantes %}**, que permite recorrer una secuencia de información (listado de **estudiantes**), que llega a la plantilla desde la función generada en views.py a través del diccionario.
- La expresión **{{ e }}** de la línea 5, presenta el valor de la variable que itera en el **ciclo for**.
- Se finaliza el tag del ciclo for en la línea 9, mediante **{% endfor %}**.

En el archivo urls.py se agrega un nuevo PATH, que asocia '**listado estudiantes**' a la vista **listadoEstudiantes**.

b. **Ejemplo 2**

A continuación, se define un proceso que permita **listar los registros del modelo Estudiante** y los números telefónicos asociados a cada estudiante (ver Figura 51).

Archivo view.py

```
views.py
32  def listadoEstudiantesDos(request):
33  """
34  Listar los registros del modelo Estudiante,
35  obtenidos de la base de datos.
36  """
```

```
37  estudiantes = Estudiante.objects.all()
38  # en la variable tipo diccionario llamada informacion_template
39  # se agregará la información que estará disponible
40  # en el template
41  informacion_template = {'estudiantes': estudiantes,
'numero_estudiantes': len(estudiantes)}
42  return render(request, 'listadoEstudiantesDos.html',
informacion_template)
```

Archivo listadoEstudiantesDos.html

```
listadoEstudiantesDos.html
1   <h2>Listado de Estudiantes</h3>
2   <h3>Número de estudiantes {{ numero_estudiantes }}</h3>
3
4   {% for e in estudiantes %}
5   ....
6   {{e}}<br/>
7   <ul>
8   {% for numero in e.numerotelefonico_set.all%}
9   <li>
10  Número telefónico {{numero}}<br/>
11  </li>
12  {% endfor %}
13  </ul>
14  <br/>
15  {% endfor %}
```

Archivo urls.py

```
urls.py
1 """
2   Manejo de urls para la aplicación
3   administrativo
4   """
5   from django.urls import path
6   # se importa las vistas de la aplicación
7   from . import views
8
9
10  urlpatterns = [
```

```
11  path('inicio', views.index, name='index'),
12  path('listado-estudiantes', views.listadoEstudiantes,·
13  name='listadoEstudiates'),
14  path('listado-estudiantes-dos', views.listadoEstudiantesDos,·
15  name='listadoEstudiatesDos'),
```

Figura 51

Listar los registros del modelo Estudiante y los números telefónicos asociados a cada estudiante

Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: [Desarrollo del ejemplo 5.3.4_1]

Ahora, continuemos con el estudio del siguiente tema que trata sobre la creación y manejo de formularios en Django utilizando ModelForm, una herramienta que facilita generar formularios.

Formularios con ModelForm en Django

5.3.5. Manejo de formularios

En Django existen varias formas para crear formularios. Asociadas a la filosofía del *framework* se recomienda usar la **clase Form** y la clase ModelForm. La segunda se la revisará en este ítem de estudio. A partir de los modelos expresados en el archivo models.py se pueden generar formularios.

Ejemplo 1

Generar un proceso que permita crear un registro de tipo Estudiante haciendo uso de la clase **ModelForm**.

- Se crea un archivo bajo la denominación **forms.py en el directorio de la aplicación**, en el cual se genera una clase llamada **EstudianteForm** que hereda de **ModelForm**; dentro de ella se adiciona una clase interna Meta, que permite indicar en el atributo **model** la entidad del archivo models.py para generar el formulario. Además, mediante el atributo **fields** se establece el listado de atributos de la clase del modelo que se necesita expresar en el modelo.

```
1   from django.forms import ModelForm
2   from administrativo.models import Estudiante
3
4
5   class EstudianteForm(ModelForm):·
6   class Meta:
7   model = Estudiante·
8   fields = ['nombre', 'apellido', 'cedula']·
9
```

- En el archivo views.py se agrega una nueva función denominada **crearEstudiante**.
    - En la línea 51 se realiza un condicional que verifica que la petición realizada sea un 'POST' (request. method); si esto es verdadero, se crea un objeto de tipo EstudianteForm llamado formulario, al cual se le envía como argumentos los parámetros que tenga el request. POST (los valores ingresados en el formulario, asociados a los atributos de la clase). Si el formulario es válido (formulario.is_valid), a través de la

función save() se guarda la información en la base de datos. Finalmente se realiza un redirect al función listadoEstudiantesDos.

- Si el método recibido a través del objeto request es diferente de 'POST'; se crea un objeto llamado formulario de tipo EstudianteForm() sin enviar argumentos.
- En la línea 59, se crea un diccionario al que se le asocia una llave 'formulario' cuyo valor es la variable formulario de tipo EstudianteForm
- La función hace referencia al archivo template crearEstudiante.html

```
48  def crearEstudiante(request):
49  """
50  """
51  if request.method=='POST':
52  formulario = EstudianteForm(request.POST)
53  print(formulario.errors)
54  if formulario.is_valid():
55  formulario.save()
56  return redirect(listadoEstudiantesDos)
57  else:
58  formulario = EstudianteForm()
59  diccionario = {'formulario': formulario}
60
61 return render(request, 'crearEstudiante.html', diccionario)
62
```

- En el archivo crearEstudiante.html se crea un formulario en HTML, se especifica en el atributo method el valor "POST".
  - En el interior del formulario se hace referencia a la variable formulario, que es recibida desde la función del views.py, mediante {{formulario.as_p}}. La expresión {{formulario.as_p}} genera el siguiente HTML

```
'<p><label for="id_nombre">Nombre:</label> <input type="text"
name="nombre" maxlength="30" required id="id_nombre"></
p>\n<p><label for="id_ apellido">Apellido:</label> <input
type="text" name="apellido" maxlength="30" required
```

```
id=”id_apellido”></p>\n<p><label for=”id_cedula”>Cedula:</label>
<input type=”text” name=”cedula” maxlength=”30” required
id=”id_cedula”></p>’
```

- Cuando se usa formularios en Django se debe usar el tag {% csrf_token %}; Django usa esta directiva para prevenir ataques de sitios que generan solicitudes falsas.

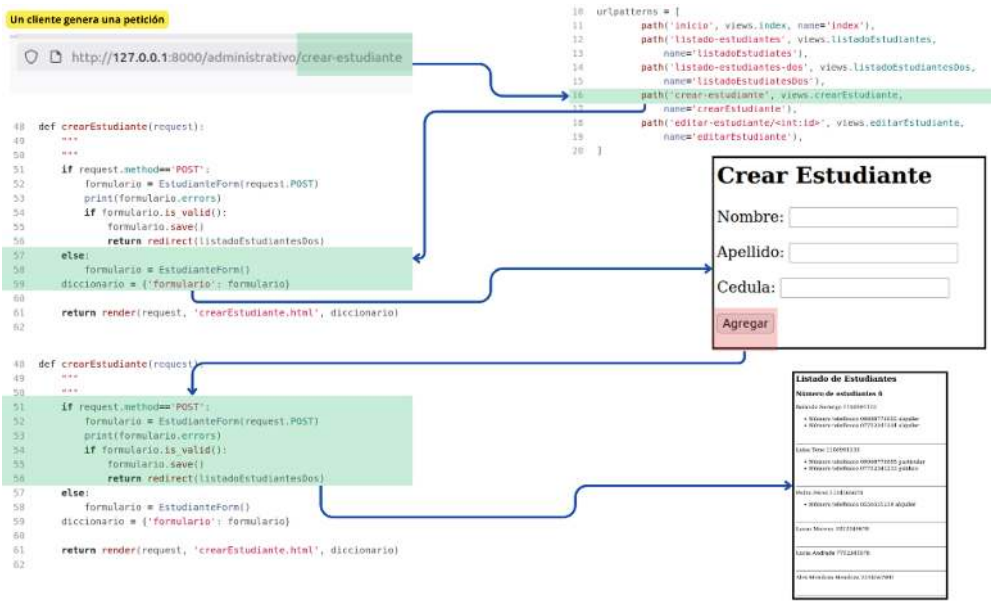
```
1 <h2>Crear Estudiante</h3>
2
3   <form action=”” method=”post”>
4   {% csrf_token %}
5   {{formulario.as_p}}
6   <p><input type=’submit’ value=’Agregar’/></p>
7   </form>
~
```

- En el archivo urls.py se agrega la nueva ruta llamada crear- estudiante que se asocia a la función **crearEstudiante** agregada en views.py.

  En la Figura 52, se indica el flujo de información expresada en las líneas anteriores al momento de ingresar la dirección en un navegador web http://127.0.0.1:8000/administrativo/crear- estudiante.

Figura 52

Proceso que permita crear un registro de tipo Estudiante, haciendo uso de la clase ModelForm



Nota. Elizalde, R., 2025.

Ejemplo 2

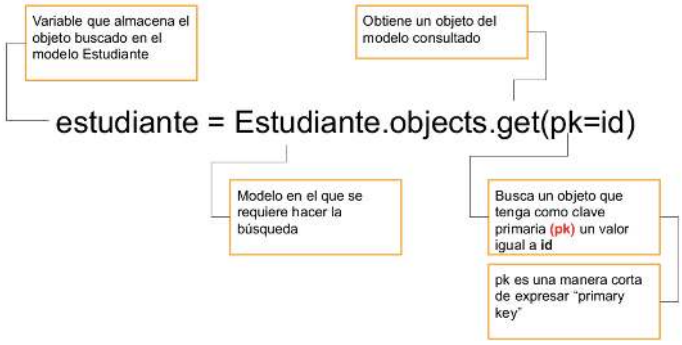
Generar un proceso que permite editar un registro de tipo Estudiante, haciendo uso de la clase **ModelForm**.

- Se usa la clase llamada EstudianteForm del archivo forms.py
- En el archivo views.py se agrega una nueva función denominada **editarEstudiante**.
    - La función tiene dos parámetros;
        - El primero el objeto tipo request.
        - El segundo parámetro permite recibir en la función una variable que permita buscar el objeto o registro de tipo Estudiante mediante el ORM.

- Para buscar un estudiante a través del ORM con base al modelo Estudiante se usa la sentencia de la línea 67 (ver Figura 53).

Figura 53

Consulta de un objeto de un modelo mediante la clave primaria



Nota. Elizalde, R., 2025.

- En la línea 68 se realiza un condicional que verifica que la petición realizada sea un 'POST' (request. method); si esto es verdadero, se crea un objeto de tipo EstudianteForm llamado formulario, al cual se le envía como argumentos los parámetros que tenga el request. POST (los valores ingresados en el formulario, asociados a los atributos de la clase) y en el atributo instance se envía la instancia del objeto buscado. Si el formulario es válido (formulario.is_valid), a través de la función save() se guarda la información en la base de datos. Finalmente se realiza un redireccionamiento (redirect) a la función listadoEstudiantesDos.
- Si el método recibido a través del objeto request es diferente de 'POST'; se crea un objeto llamado formulario de tipo EstudianteForm, enviando como argumento la instancia del objeto buscado (instance=estudiante); con el objetivo de cargar el formulario con los datos del objeto y proceder a realizar la edición.
- En la línea 76, se crea un diccionario al cual se le asocia una llave 'formulario' cuyo valor es la variable formulario de tipo EstudianteForm
- La función hace referencia al archivo template editarEstudiante.html

```
64  def editarEstudiante(request, id):
65  """
66  """
67  estudiante = Estudiante.objects.get(pk=id)
68  if request.method=='POST':
69  formulario = EstudianteForm(request.POST,
instance=estudiante)
70  print(formulario.errors)
71  if formulario.is_valid():
72  formulario.save()
73  return redirect(listadoEstudiantesDos)
74  else:
75  formulario = EstudianteForm(instance=estudiante)
76  diccionario = {'formulario': formulario}
77
78 return render(request, 'editarEstudiante.html', diccionario)
```

- En el archivo editarEstudiante.html se crea un formulario en HTML y se renderiza el formulario enviado desde la vista.

```
1 <h2>Editar Estudiante</h3>
2
3   <form action="" method="post">
4   {% csrf_token %}
5   {{formulario.as_p}}
6   <p><input type='submit' value='Actualizar'/></p>
7   </form>
```

- En el archivo urls.py se agrega la nueva ruta llamada **editar- estudiante que permite un parámetro tipo entero** que se asocia a la función editarEstudiante agregada en views.py (Figura 54).

Figura 54

Generar un path en el archivo urls.py con un argumento tipo entero

Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: [Desarrollo del ejemplo 5.3.5_1]

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 14

Unidad 5. Uso de *frameworks* de ambiente *web*

5.3. Desarrollo de aplicaciones mediante el *framework* Django

5.3.6. Herencia de plantillas

Para el estudio de este apartado se recomienda seguir lo descrito en el capítulo 4, denominado ***El sistema de plantillas*** del libro: *La guía definitiva de Django – desarrolla aplicaciones* web de forma rápida y sencilla, de manera particular los apartados relacionados con ***Herencia de plantillas***.

Como se menciona en el libro, la herencia de plantillas es una característica que permite crear un esqueleto – plantilla - *template* base -, donde se agregan partes que se pudieran usar en todas las plantillas de la aplicación, dejando expresado el conjunto de “bloques”, secciones que se pueden personalizar en las plantillas secundarias que heredan del *template* base.

A continuación, se detalla una primera versión de una plantilla base con el nombre **master.html**. El archivo debe estar en la carpeta *templates*.

```
<!DOCTYPE html>
<html>
<head>
<meta charset=”utf-8” />
<meta name=”viewport” content=”width=device-width” />
<title>
{% block title %}
{% endblock %}
</title>
</head>
<body>
<h1>Aplicación Administración </h1>
<hr>
{% block content %}
{% endblock %}

<hr>
<footer>
<p>Loja-Ecuador</p>
<p>{% now “jS F Y H:i” %}</p>
</footer>
</body>
</html>
```

En las líneas descritas en el archivo master.html, se aprecia la estructura total de una página web; pero existen los denominados bloques, expresados con **{% block inicio %} … {% endblock%}**. Estos bloques pueden ser particularizados en cada plantilla que herede de master.html. Todo lo que no está expresado entre un *tag* ***block*** será renderizado en la plantilla hija.

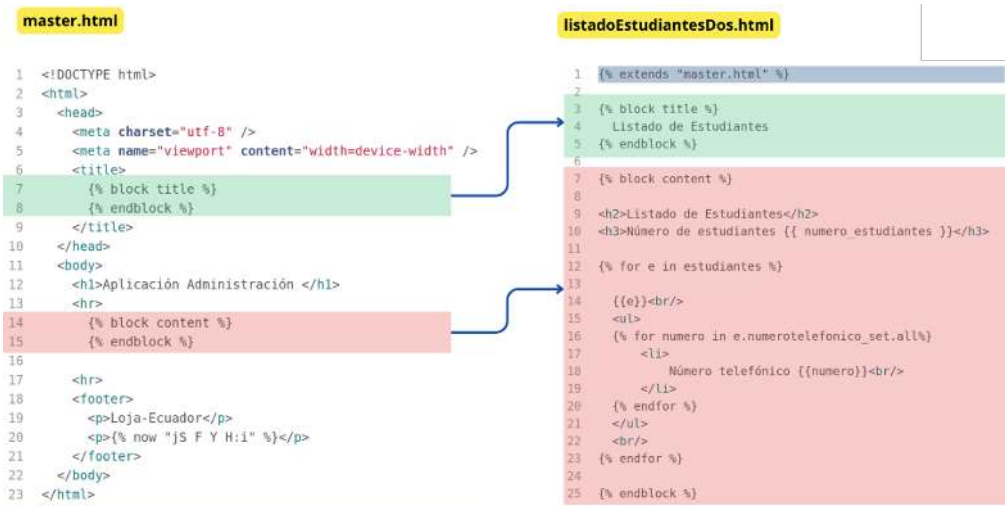
La plantilla de nombre listadoEstudiantesDos.html queda de la siguiente manera:

```
1 {% extends "master.html" %}
2
3 {% block title %}
4 Listado de Estudiantes
5 {% endblock %}
6
7 {% block content %}
8
9 <h2>Listado de Estudiantes</h3>
10 <h3>Número de estudiantes {{ numero_estudiantes }}</h3>
11
12  {% for e in estudiantes %}
13  ....
14  {{e}}<br/>
15  <ul>
16  {% for numero in e.numerotelefonico_set.all%}
17  <li>
18  Número telefónico {{numero}}<br/>
19  </li>
20 {% endfor %}
21 </ul>
22 <br/>
23 {% endfor %}
24
25 {% endblock %}
```

- La primera línea utiliza el tag **extends** junto al nombre de la plantilla base "master.html"; con lo cual se hereda la estructura de la plantilla. En el archivo se personalizan los bloques "title" y "*content*" con el contenido que se requiere (ver figura 55).

Figura 55

Herencia de plantillas en Django

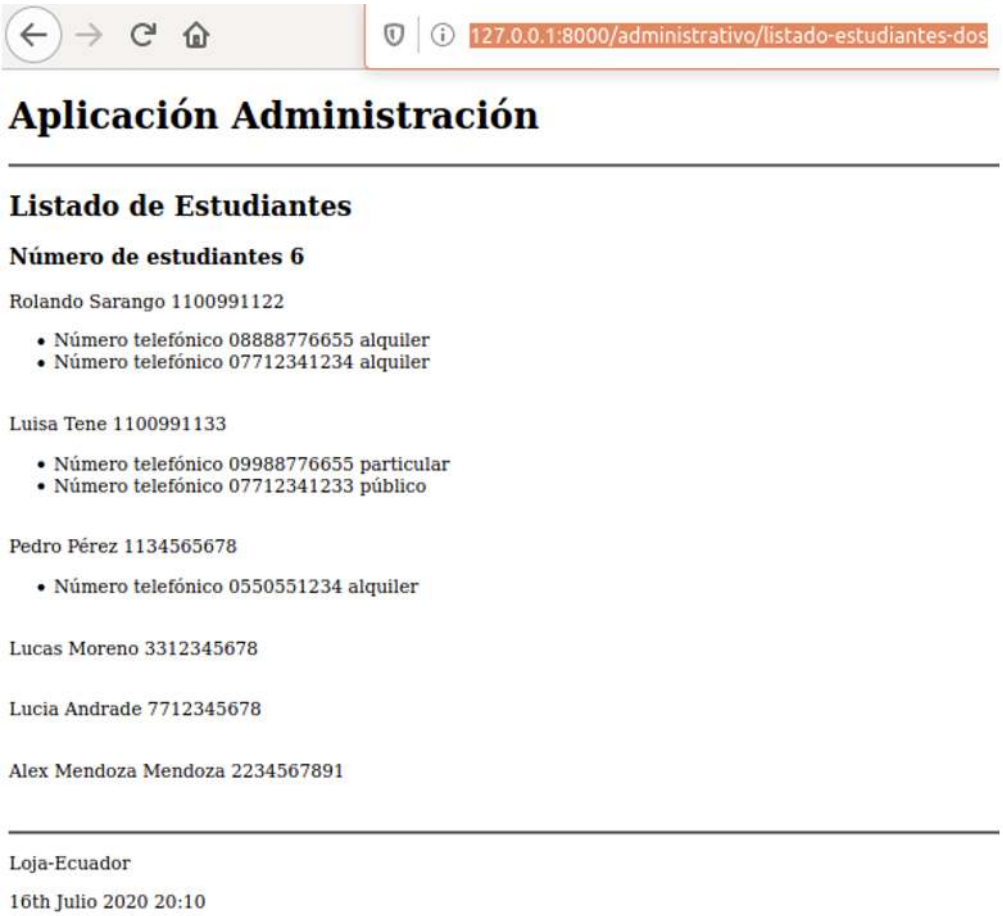


Nota. Elizalde, R., 2025.

Si usted inicia-levanta el proyecto y accede a la dirección: http://127.0.0.1:8000/administrativo/listado-estudiantes-dos, observará la pantalla expuesta en la siguiente figura:

Figura 56

Visualización de una plantilla que hereda de "master.html"



Nota. Elizalde, R., 2025.

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 5.3.6_1

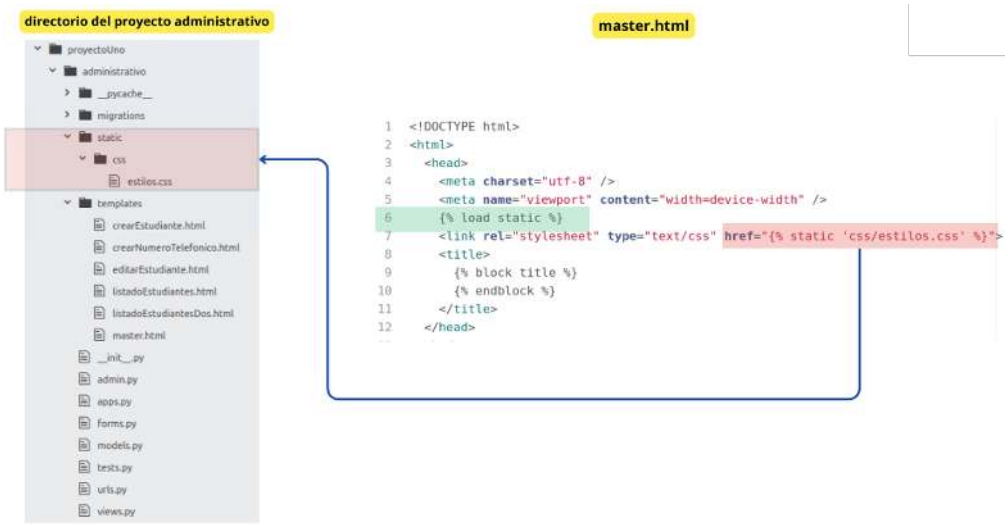
5.3.7. Agregar hojas de estilo a plantillas

Ahora que se entiende el proceso de manejo de plantillas mediante herencia, es momento de agregar a la aplicación acceso a hojas de estilo - CSS.

Para las hojas de estilo se necesita agregar las rutas de los archivos CSS en la plantilla “**master.html**”. En el directorio de la aplicación se crea una estructura de directorios **static – css**; es ahí donde se debe agregar los archivos de las hojas de estilo. En el archivo “**master. html**” se usa el tag **load** para cargar los archivos del directorio **static** (ver Figura 57).

Figura 57

Agregar CSS a las plantillas de una aplicación en Django

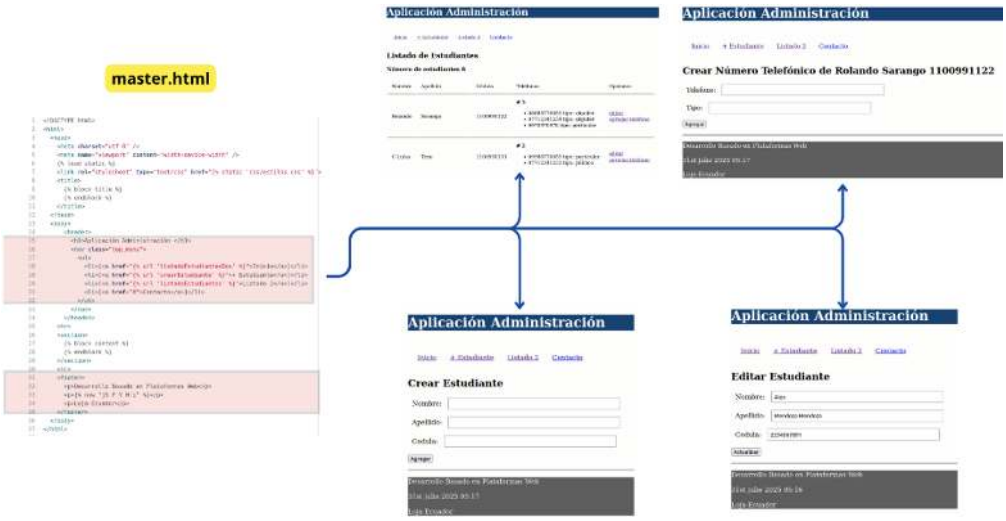


Nota. Elizalde, R., 2025.

Luego de agregar el archivo con las propiedades CSS en el archivo máster, todas las plantillas que heredan del template principal quedan de la manera indicada en la Figura 58.

Figura 58

Aplicación de hojas de estilo en plantillas que heredan de master. Html



Nota. Elizalde, R., 2025.

A continuación, se detalla las opciones del pequeño sistema generado:

- Parte administrativa de la aplicación:
  - http://127.0.0.1:8000/admin
- Listado de estudiantes:
  - http://127.0.0.1:8000/administrativo/listado- estudiantes-dos
  - Ruta del archivo urls.py:
    - path('listado-estudiantes-dos', views.listadoEstudiantesDos, name='listadoEstudiantesDos'),
  - Función del views.py
    - listadoEstudiantesDos
  - Template
    - listadoEstudiantesDos.html

- Agregar nuevo estudiante
  - http://127.0.0.1:8000/administrativo/crear-estudiante
  - Ruta del archivo urls.py:
    - path('crear-estudiante', views.crearEstudiante, name='crearEstudiante'),
  - Función del views.py
    - crearEstudiante
  - Template
    - crearEstudiante.html
- Editar estudiante
  - http://127.0.0.1:8000/administrativo/editar-estudiante/1
  - Ruta del archivo urls.py:
    - path('editar-estudiante', views.editarEstudiante, name='editarEstudiante'),
  - Función del views.py
    - editarEstudiante
  - Template
    - editarEstudiante.html
- Agregar número telefónico
  - http://127.0.0.1:8000/administrativo/crear-telefono/1
  - Ruta del archivo urls.py:
    - path('crear-telefono', views.crearTelefono, name='crearTelefono'),

- Función del views.py
  - crearTelefono
- Template
  - crearTelefono.html

Estimado/a estudiante, puede descargar el ejemplo desde el siguiente repositorio Git e instalarlo en su entorno local: Desarrollo del ejemplo 5.3.7_1.

Ahora, para reforzar lo aprendido, realice las siguientes actividades:

Actividades de aprendizaje recomendadas

1. Reutilizar una plantilla HTML en Django es fundamental para desarrollar aplicaciones web de manera eficiente y consistente. Permite definir la estructura visual común (encabezado, navegación, pie de página) una sola vez en un master.html, del cual todas las demás páginas heredan, garantizando uniformidad y ahorrando tiempo al evitar la repetición de código. Esta práctica no solo simplifica las actualizaciones y el mantenimiento, sino que también fomenta una separación de responsabilidades entre la lógica del negocio (vistas), la presentación (plantillas) y los estilos/ interactividad (archivos estáticos), facilitando la gestión de contenido dinámico y la colaboración en equipo. Es necesario que se tome el ejemplo django-add-template, realice un fork al repositorio y luego lo inicialice en su entorno local.
2. Realizar la autoevaluación 6, donde se presentan un conjunto de preguntas en función de las temáticas descritas en la unidad 5. *Uso de frameworks de ambiente web*. En la parte inicial de la actividad se presentan los contenidos de la guía didáctica y Recursos Educativos Abiertos que se deben revisar para contestar las interrogantes. Usted puede revisar los siguientes temas:
   - Ítems 5.3.4, 5.3.5, 5.3.6 y 5.3.7 de la presente guía.

- Libro: La guía definitiva de Django – desarrolla aplicaciones *web* de forma rápida y sencilla.
    - Capítulo 7: Procesamiento de formularios.
    - Capítulo 4: El sistema de plantillas - apartado herencia de plantillas.

La autoevaluación le permitirá identificar el nivel de aprendizaje alcanzado y los ítems que se deben volver a revisar para lograr los resultados de aprendizaje propuestos. Además, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Autoevaluación 6

Instrucción: lea detenidamente cada uno de los enunciados de las preguntas y seleccione la o las respuestas correctas.

1. **¿Cuál es el archivo en donde se maneja la lógica en el *framework* Django?**

    a. Views.py.
    b. Models.py.
    c. Def.py.

2. **Dada la siguiente estructura, identifique el nombre de la aplicación.**

```
├── manage.py
├── ministerio
│   ├── asgi.py
│   ├── __init__.py
│   ├── __pycache__
│   ├── settings.py
│   ├── urls.py
```

```
| └── wsgi.py
└── rrhh
    ├── admin.py
    ├── apps.py
    ├── __init__.py
    ├── migrations
    | └── __init__.py
    ├── models.py
    ├── tests.py
    └── views.py
```

a. Ministerio.
b. Apps.
c. Rrhh.

3. **Dadas las siguientes líneas de código que forman parte de una función del archivo views.py, identifique qué tipo de datos se pasan como contexto al *template* o plantilla.**

```
estudiantes =

Persona.objects.all() mensaje

= "Listado de Personas"

numeroPersonas =

len(estudiantes)

informacion_template = {'personas': estudiantes,
'numero_estudiantes':

numeroPersonas, 'titulo': mensaje}
```

```
return render(request, 'listadoEstudiantes.html',
informacion_template)
```

a. 3 variables: 1) listado de objetos de tipo Persona; 2) numero_estudiantes tipo entero; 3) titulo de tipo cadena
b. 3 variables: 1) listado de objetos de tipo Estudiante; 2) numero_estudiantes tipo entero; 3) titulo de tipo cadena
c. 3 variables: 1) listado de objetos de tipo Persona; 2) numero_estudiantes tipo cadena; 3) titulo de tipo cadena

4. **En los *templates* de Django, ¿cómo se representan las variables?**

a. { variable }
b. {{ variable }}
c. [[variable]]

5. **A través del uso de tags en Django, identifique el ciclo repetitivo que se puede representar en los *templates*.**

a. While.
b. Do-while.
c. For.

6. **¿Cuál es la clase que se puede usar para crear un formulario a partir de una clase definida en el modelo?**

a. Form.
b. ModelForm.
c. ModelForms.

7. **¿Cuál es la directiva que se usa en Django para prevenir ataques de sitios que generan solicitudes falsas?**

a. {% csrf_token%}
b. {{ csrf_token }}
c. {% csrfToken%}

8. **Dado el siguiente archivo y, en función de una petición generada a través de la URL 127.0.0.1:8000/administrativo/1/ reporte-general, identifique la vista a la cual se hace referencia.**

```
url.py del proyecto

urlpatterns = [

  path('administrativo/', include('administrativo.urls')),

]
```

```
url.py de la aplicación

 urlpatterns = [

  path('', views.listadoEstudiantesDos,

    name='listadoEstudiantesDos'),

  path('editar-estudiante/<int:id>', views.editarEstudiante,

    name='editarEstudiante'),

  path('<int:id>/reporte-general', views.crearReporte,

    name='crearReporte'),

]
```

a. Views.crearReporte.
b. Views.
c. CrearReporte.

9. **Dado el siguiente archivo llamado principal.html, identifique la plantilla que hereda de forma correcta del *template* principal. html.**

```
1   <!DOCTYPE html>

2   <html>

3   <head>

4   <meta charset="utf-8" />
```

```
5   <meta name=”viewport” content=”width=device-width” />
6   <title>
7   {% block title %}
8   {% endblock %}
9   </title>
10  </head>
11  <body>
12  <h1>Aplicación Autoevaluación </h1>
13  <hr>
14  {% block content %}
15  {% endblock %}
16
17  <hr>
18  <footer>
19  <p>Loja-Ecuador</p>
20  <p>{% now “jS F Y H:i” %}</p>
21  </footer>
22  </body>
23  </html>
```

a. `{% extends “principal.html” %}`

`{% title %}`

`Crear Estudiante`

`{% endblock %}`

`{% content %}`

```
<h2>Crear Estudiante</h3>

<form action="" method="post">

{% csrf_token %}

{{formulario.as_p}}

<p><input type='submit' value='Agregar'/></p>

</form>

{% endblock %}
```

b.
```
{% extends "principal.html" %}

{% block title %} Crear Estudiante

{% end %}

{% block content %}

<h2>Crear Estudiante</h3>

<form action="" method="post">

{% csrf_token %}

{{formulario.as_p}}

<p><input type='submit' value='Agregar'/></p>

</form>

{% end %}
```

c.
```
{% extends "principal.html" %}

{% block title %} Crear Estudiante

{% endblock %}

{% block content %}

<h2>Crear Estudiante</h3>
```

```
<form action="" method="post">
{% csrf_token %}
{{formulario.as_p}}
<p><input type='submit' value='Agregar'/></p>
</form>
{% endblock %}
```

10. **Dado el siguiente modelo de un proyecto en Django, identifique la clase que permita generar un formulario a través de ModelForm.**

```
class Docente(models.Model):
  nombre = models.CharField(max_length=30)
  apellido = models.CharField(max_length=30)
  identificacion = models.CharField(max_length=30, unique=True)
  def   str  (self):
   return "%s %s %s" %
    (self.nombre, self.apellido,
    self.identificacion)
```

a.
```
class DocenteForm(models.Model):
  class Meta:
    model = Docente
    fields = ['nombre', 'apellido', 'identificacion']
```

b.
```
class DocenteForm(ModelForm):
  model = Docente
  fields = ['nombre', 'apellido', 'identificacion']
```

c.
```
class DocenteForm(ModelForm):
```

```
class Meta:
    model = Docente
    fields = ['nombre', 'apellido', 'identificacion']
```

Ir al solucionario

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 15

Actividades finales del bimestre

En la semana 15 de estudio se recomienda repasar los contenidos del segundo bimestre, prestando especial atención a los temas que hayan generado dudas. Asimismo, se sugiere volver a realizar los ejercicios propuestos y responder nuevamente las autoevaluaciones, con el fin de consolidar el aprendizaje y fortalecer las competencias desarrolladas. Para ello, complete las siguientes actividades recomendadas.

Actividades de aprendizaje recomendadas

1. La puesta en producción de una aplicación *web* desarrollada en lenguaje Python constituye un aspecto importante y de gran relevancia en el ciclo de vida del desarrollo de *software*. Para comprender este proceso, se recomienda la revisión y análisis de los pasos a seguir para desplegar aplicaciones, usando la combinación de un servidor *web* (Nginx) con un servidor de aplicaciones Python (como Gunicorn). Aunque las configuraciones y comandos pueden variar según el sistema operativo (siendo Ubuntu un ejemplo común), la recreación de estos entornos en sus sistemas locales reforzará los conceptos generales relacionados con el ámbito del desarrollo y despliegue de aplicaciones *web*".

2. Realice nuevamente las autoevaluaciones 4, 5 y 6, donde se explica y ejemplifica la generación de aplicaciones sencillas con PHP y el uso de *framework* de ambiente *web* con Python Django. Además, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Contenidos, recursos y actividades de aprendizaje recomendadas

Semana 16

Actividades finales del bimestre

Distinguidos estudiantes.

Hemos llegado al final de nuestro estudio, la semana 16, que es un período de tiempo destinado para que usted pueda reforzar temáticas no comprendidas en el transcurso del segundo bimestre, con el objetivo de prepararse de la mejor manera para rendir la evaluación presencial del bimestre. Por ello, se plantea el repaso de las siguientes unidades, generando resúmenes, cuadros sinópticos, mapas conceptuales, entre otros.

- Unidad 4. Programación en el servidor web.
- Unidad 5. Uso de *frameworks* de ambiente web.

Además, es importante realizar las siguientes acciones:

Revisar los recursos de aprendizaje recomendados en el plan docente de la asignatura.

Plantear ejercicios y problemáticas similares a las expuestas en los contenidos de la guía.

Revisar las preguntas que conforman los cuestionarios de repaso del bimestre planteados en el Entorno Virtual de Aprendizaje.

4. Solucionario

Autoevaluación 1

Pregunta	Respuesta	Retroalimentación
1	b	Evalúa si el *software* cumple correctamente sus requerimientos y funciona sin errores; es un indicador directo de calidad.
2	c	Proteger la aplicación contra ataques como SQL injection forma parte de asegurar la integridad y confidencialidad de los datos.
3	c	Se enfoca de igual manera en garantizar el acceso a *Internet* a cualquier persona, independiente de su condición.
4	b	HTTPS (*HyperText Transport Protocol Secure*), permite utilizar una conexión con servidores seguros de Internet. El puerto usado por el protocolo es 443.
5	b	Una URL, *Uniform Resource Locator* está formada por Protocolo: HTTP o HTTPS; Dominio; Ruta. Estas partes permiten identificar el método de acceso, el servidor de destino y el recurso específico dentro de él.
6	c	Permite a los usuarios acceder a aplicaciones y almacenamiento sin depender de infraestructura local, optimizando flexibilidad y escalabilidad.
7	a	HTML estructura contenido en la web mediante etiquetas, no ejecuta lógica como un lenguaje de programación.
8	a	Algunos complementos para XML: XSL, XPath, XLink, XPointer y XQL.
9	b	Se indica que el hosting es el proceso de alojar una página web en un servidor.
10	c	Un servidor centraliza almacenamiento y procesamiento, y puede atender solicitudes de múltiples clientes simultáneamente.

Ir a la autoevaluación

Autoevaluación 2

Pregunta	Respuesta	Retroalimentación
1	b	Las pruebas con usuarios reales permiten evaluar la usabilidad y experiencia auténtica de la aplicación.
2	a	Esta declaración (DOCTYPE) indica al navegador cómo renderizar la página según la versión especificada de HTML.
3	c	Se recomienda usar la herramienta ***Markup Validation Service*** para asegurar que el código HTML cumpla con los estándares de la W3C, garantizando accesibilidad y compatibilidad de los contenidos generados en la aplicación web.
4	c	UTF-8 garantiza que se interpreten correctamente caracteres especiales y acentos en la página web.
5	b	Usar ***class*** permite aplicar estilos CSS definidos en un archivo externo, como colores y fondos personalizados.
6	b	En JavaScript, cuando se usa el método innerText para establecer un valor, el mismo se representa tal cual fue establecido.
7	a	En HTML, para cargar jQuery desde una CDN se utiliza la etiqueta *<script>* con el atributo src, lo que permite que el navegador obtenga y ejecute el archivo JavaScript desde la URL especificada.
8	a	CouchDB utiliza un modelo orientado a documentos, ideal para almacenar datos en formato JSON, mientras que Neo4J emplea un modelo basado en grafos, óptimo para representar relaciones complejas entre nodos.
9	c	Fauxton es la interfaz gráfica oficial de CouchDB que permite la administración y consulta de datos; la ruta local por defecto incluye /_utils/ para acceder a esta interfaz desde el navegador.
10	c	Un documento válido en CouchDB debe usar claves y valores en formato JSON correctamente escritos, con comillas dobles en las cadenas y en los nombres de las claves.

Ir a la autoevaluación

Autoevaluación 3

Pregunta	Respuesta	Retroalimentación
1	a	Un ORM permite mapear objetos de clases en un lenguaje de programación hacia registros en una base de datos relacional, facilitando la persistencia de datos sin escribir SQL manualmente.
2	b	Una ventaja clave de los ORM es la abstracción del motor de base de datos, lo que permite cambiarlo con mínimo impacto en el código.
3	b	Doctrine es un ORM ampliamente utilizado en PHP que ofrece herramientas para trabajar con bases de datos relacionales de forma orientada a objetos.
4	a	La forma correcta de instalar una librería en Python mediante el gestor de paquetes **pip** es escribir en la terminal: pip install [nombre-librería].
5	a	Se crea una variable que haga uso del módulo create_engine, a la cual se envía como parámetro una cadena de texto que indica el motor a utilizar.
6	b	El método **Base.metadata.create_all(engine)** genera las tablas a partir de las clases definidas, vinculándolas con la conexión establecida.
7	c	En la sentencia: **datos = session.query(Hospital).filter(Hospital.numero_camas==80).all()** Se filtran todos los objetos de tipo Hospital que tengan el atributo numero_camas igual a 80.
8	c	La sentencia: **datos = session.query(Hospital).order_by(Hospital.numero_pisos).all()** Permite obtener todos los objetos de tipo Hospital ordenados de menor a mayor en función del atributo numero_pisos.
9	b	La sentencia: **datos = session.query(Hospital).filter(Hospital.nombre.like("%Ay%")).all()** Permite obtener los objetos de tipo Hospital que contengan en el valor del atributo nombre la cadena "Ay".
10	a	En SqlAlchemy la forma correcta de identificar a una columna como clave foránea es a través de: persona_id = Column(Integer, **ForeignKey**('persona.id'))}.

Pregunta	Respuesta	Retroalimentación

Ir a la autoevaluación

Autoevaluación 4

Pregunta	Respuesta	Retroalimentación
1	c	Apache es conocido por su flexibilidad y compatibilidad, mientras que Nginx destaca por su alto rendimiento y bajo consumo de recursos.
2	c	El código PHP debe ir entre las etiquetas: <?php ?> Por tal razón, la opción correcta es la que tiene la porción de código de la siguiente forma: <?php echo "Verificando conocimientos" ; ?>

3	c	La porción de código en PHP
		<?php
		function operacion($num1, $num2){ return $num1 + $num2;
		}
		$tabla = 3;
		$inicio = 1;
		echo ' Tabla del '. $tabla, " "; while($inicio <= 5){
		$formato = "<tr>
		<td>%d</td><td>%s</td><td>%d</td><td class='celda'>%d</td>
		</tr>";
		echo sprintf($formato, $inicio, "+", $tabla, operacion($tabla, $inicio));
		$inicio++;
		}
		?>
		de acuerdo con la lógica, permite generar una salida como la expuesta en la opción c.

4	b	El proceso repetitivo expuesto en el código PHP junto a la estructura condicional genera una salida como la imagen de la opción b.
5	a	El proceso repetitivo expuesto en el código PHP y el uso de las funciones operación y operacion2, genera una salida como la imagen de la opción b.
6	c	Para capturar los valores enviados a través de un formulario con el método POST es a través del uso de $_REQUEST y para acceder a las variables necesarias se lo hace con el nombre del atributo como cadena.
7	b	La forma correcta de crear un objeto que permita crear un enlace a una base de datos MariaDB o Mysql es haciendo uso de la clase: mysqli

Pregunta	Respuesta	Retroalimentación
8	c	La forma correcta de realizar un redirect en PHP es: header(“Location: archivo-destino.php?variable=”valor”);
9	a	En el archivo se genera un formulario que permite crear un conjunto de tags: input type=radio y label. Se hace uso de un ciclo repetitivo que en cada iteración llama una función llamada información que retorna una cadena.
10	c	Haciendo uso de date(‘Ymd’,strtotime($v)); se presenta una fecha de la forma: Añomesdia – 20200725

Ir a la autoevaluación

Autoevaluación 5

Pregunta	Respuesta	Retroalimentación
1	a	Un *framework* integra un lenguaje de programación, librerías y bibliotecas que agilizan el desarrollo, proporcionando código reutilizable y estructuras predefinidas.
2	b	Twing y Blade son motores de plantillas usados en PHP; Blade es nativo de Laravel y Twing es una implementación de Twig para PHP.
3	a	El comando **django-admin startproject [nombre-proyecto]** crea la estructura inicial de un proyecto Django, incluyendo archivos de configuración y manejo.
4	b	Luego de haber creado el proyecto, el uso del manage. py permite generar aplicaciones, actualizaciones, crear usuarios administradores, entre otras tareas.
5	c	**python manage.py migrate** aplica las migraciones, creando o modificando tablas según los modelos definidos en el proyecto.
6	a	**python manage.py startapp [nombre-aplicación]** genera la estructura de carpetas y archivos base para una nueva aplicación dentro de un proyecto Django.
7	b	Dada una entidad llamada «Universidad», la forma correcta de obtener los registros guardados en la base de datos es mediante la sentencia **Universidad.objects.all()**, la cual devuelve un QuerySet con todos los registros de la tabla correspondiente al modelo «Universidad».
8	a	Para verificar si una cadena está dentro de los valores de un atributo, se usa: **nombreAtributo contains="cadena"**
9	b	Además de definir qué columnas se muestran, list_display puede combinarse con métodos personalizados del modelo para mostrar datos calculados o transformados.
10	a	Al crear un superusuario con **createsuperuser**, Django te solicita un nombre de usuario, correo y contraseña. Este usuario no solo accede al panel de administración, sino que tiene permisos completos sobre todos los modelos y aplicaciones del proyecto, lo que permite gestionar usuarios, permisos y datos sin restricciones.

Ir a la autoevaluación

Autoevaluación 6

Pregunta	Respuesta	Retroalimentación
1	a	Este archivo contiene la lógica que procesa las solicitudes del usuario y devuelve respuestas, generalmente conectando modelos y plantillas; se puede pensar como el "controlador" en el patrón MVC.
2	c	Dada la estructura y asumiendo que se ha creado una aplicación en función del comando correcto, el nombre de la aplicación es: rrhh
3	a	El contexto enviado al template permite acceder a datos dinámicos; Django convierte los objetos del ORM en estructuras que la plantilla puede mostrar fácilmente.
4	b	La forma correcta para representar el valor de un variable en un template de Django es {{ }}. La doble llave indica a Django que se renderice el valor de la variable; también soporta filtros para transformar datos directamente en la plantilla.
5	c	En Django, a través del sistema de plantillas, se puede representar un ciclo repetitivo for {% for …. %} {% endfor %}
6	b	La clase **ModelForm** facilita la creación automática de formularios a partir de modelos existentes, evitando definir manualmente cada campo y validación; asegura consistencia entre modelo y formulario.
7	a	La representación de un tag en un *template* de Django es bajo la forma: {% tag %} Django usa el tag con nombre csrf_token para prevenir ataques de sitios que generan solicitudes falsas.
8	c	Cuando se realiza una petición, se recibe la misma en el archivo url.py del proyecto; path('administrativo/', include('administrativo.urls')), luego en el archivo url de la aplicación, la petición coincide con el path: path('<int:id>/ reporte-general ', views.crearReporte, name='crearReporte') Por lo tanto el nombre de la vista es: crearReporte
9	c	La forma correcta de indicar un bloque para personalizar es: **{% block inicio %} … {% endblock%}**; donde inicio es el nombre del bloque.

Pregunta	Respuesta	Retroalimentación
10	c	La opción que implementa correctamente un formulario mediante **ModelForm** es la opción c, la cual utiliza la subclase **Meta** con los atributos **model** y **fields**, indicando a Django qué modelo usar y qué campos incluir, lo que genera automáticamente validaciones y *widgets* coherentes con los tipos de datos del modelo.

Ir a la autoevaluación

5. Referencias bibliográficas

Colomina Pardo, O., Arques Corrales, P., & Montoyo-Bojo, J. (2011). Tecnologías Web. Tema 4: Frameworks JavaScript. Jquery.

jQuery UI (2020). Recuperado de http://learn.jquery.com/jquery-ui/gettingstarted/

Herrera, H. A., & Valenzuela, C. R. (2016). NoSQL, la nueva tendencia en el manejo de datos. Tecnología Investigación Y Academia, 4(1), 147-150.

D.Robles, M. Sánchez, R. Serrano, B. Adárraga & D. Heredia. "¿Qué características tienen los esquemas NoSQL?"

Investigación y Desarrollo en TIC, vol. 6, no. 1, pp. 40-44, 2015.

Museros Cabedo, L., & Sanz Blasco, I. (2010). Tema 6: Otros Modelos de Bases de Datos. El modelo orientado a objetos y objeto- relacional.

Perez-Martin, J. Object Relational Mapping.

Viñals, J. T. (2012). Del cloud computing al big data. Barcelona. universitat oberta de catalunya.

Learning sqlalchemy (2020). Recuperado de https://riptutorial.com/Download/sqlalchemy.pdf

Django, la guía definitiva (2020). Recuperado de https://pythonizame.s3.amazonaws.com/media/Book/guia-definitiva-django-18/file/34ba425e-5985-11e5-964d-04015fb6ba01.pdf

Gutiérrez González, Á., & López Goytia, J. L. (2016). Desarrollo y programación en entornos web. México. Alfaomega.

Casas, R. J., Nin, G. J., & Julbe, L. F. (2019). Big data: Análisis de datos en entornos masivos. Editorial UOC.

Pérez Martínez, E. (2015). Desarrollo de aplicaciones mediante el Framework de Spring: (ed.). RA-MA Editorial.

6. Anexos

Anexo 1. Ejemplo de consulta de datos con ORM SqlAlchemy

1. Se genera un archivo de Python para crear la base de datos; la entidad Saludo tiene tres atributos:

   - Id, que la clave primaria; no es necesario agregarla en cada instancia; se agrega de forma autoincremental por defecto.
   - Mensaje, de tipo cadena.
   - Tipo, con tipo de dato cadena.

```
1 from sqlalchemy import create_engine
2 from sqlalchemy.ext.declarative import declarative_base
3 from sqlalchemy.orm import sessionmaker
4 from sqlalchemy import Column, Integer,
String 5
6 # se genera enlace al gestor de base de
7 # datos
8 # para el ejemplo se usa la base de datos
9 # sqlite
10 engine =
create_engine('sqlite:///demobase2.db') 11
12 Base =
declarative_base() 13
14 class Saludo(Base):
15      _tablename_ = 'saludo'
16       id = Column(Integer, primary_key=True)
17       mensaje = Column(String)
18       tipo =
Column(String) 19
20 Base.metadata.create_all(engine)
```

2. Se agregan registros a la base de datos, a través de otro archivo de Python.

```
1 from sqlalchemy import create_engine
2 from sqlalchemy.orm import
sessionmaker 3
4 # se importa la clase(s) del·
5 # archivo genera_tablas
6 from genera_tablas import Saludo
7
8 # se genera enlace al gestor de base de
9 # datos
10 # para el ejemplo se usa la base de datos
11 # sqlite
12 engine =
create_engine('sqlite:///demobase2.db') 13
14 Session = sessionmaker(bind=engine)
15 session =
Session() 16
17 # se crea un objetos de tipo
Saludo 18
19 saludo1 = Saludo()
20 saludo1.mensaje = "Hola que tal"
21 saludo1.tipo =
"informal" 22
23 saludo2 = Saludo()
24 saludo2.mensaje = "Buenos días"
25 saludo2.tipo =
"formal" 26
27 saludo3 = Saludo()
28 saludo3.mensaje = "Que hay"
29 saludo3.tipo= "informal"
30
31 saludo4 = Saludo()
32 saludo4.mensaje = "Buenas noches"
33 saludo4.tipo = "formal"
```

```
34
35 # se agrega los objetos de tipo Saludo
36 # a la sesión
37 # a la espera de un commit
38 # para agregar un registro a la base de·
39 # datos demobase2.db
40 session.add(saludo1)
41 session.add(saludo2)
42 session.add(saludo3)
43 session.add(saludo4)
44
45 # se confirma las transacciones
46 session.commit()
```

3. Se generan varios archivos de Python, que permiten realizar diversos tipos de consultas.

   - Selección de todos los registros de la clase Saludo.

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker 3
# se importa la clase(s) del·
# archivo genera_tablas
from genera_tablas import Saludo

# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine('sqlite:///demobase2.db') 13
Session = sessionmaker(bind=engine)
session = Session()
```

```

# Obtener todos los registros de·
# la tabla saludo
losSaludos = session.query(Saludo).all()
# la consulta con .all(), devuelve
# una lista de objetos de tipo Saludo
# que se le asigna como valor a la variable
# losSaludos
# Se recorre la lista a través de un ciclo
# repetitivo for en python

for s in losSaludos:
    print(s.mensaje)
    print(s.tipo)
    print("                ")
```

Respuesta:

```
Hola que tal
informal

_________
Buenos      días
formal

_________
Que hay
informal

_________
Buenas noches formal
```

   - Selección de todos los registros de la clase Saludo, ordenados en función del atributo mensaje.

```
1 from sqlalchemy import create_engine
2 from sqlalchemy.orm import
sessionmaker 3
4 from genera_tablas import
Saludo 5
6 # datos
7 # para el ejemplo se usa la base de datos
8 # sqlite
9 engine =
create_engine('sqlite:///demobase2.db') 10
11 Session = sessionmaker(bind=engine)
12 session =
Session() 13
14 # Obtener todos los registros de·
15 # la tabla saludo ordenados por el atribut
16 # mensaje
17 losSaludos = session.query(Saludo).order_by(Saludo.mensaje)
18 # la consulta con .order_by, ordena los resultados en función del·
19 # atributo de la clase Saludo, mensaje,·
20 # Devuelve una lista de objetos de tipo Saludo
21 # y se le asigna como valor a la variable
22 # losSaludos
23
24 # Se recorre la lista a través de un ciclo
25 # repetitivo for en
python 26
27 for s in losSaludos:
28     print("Mensaje: %s" % (s.mensaje))
29     print("Tipo: %s" % (s.tipo))
30     print("id: %s" % (s.id))
31     print("_____")
```

Respuesta:

```
Mensaje: Buenas noches
Tipo: formal
id: 4
_____
Mensaje: Buenos
días Tipo: formal
id: 2
_____
Mensaje: Hola que
tal Tipo: informal
id: 1
_____
Mensaje: Que
hay Tipo:
informal
id: 3
_____
```

- Obtener todos los registros de la tabla saludo que tengan como valor en el atributo tipo la expresión “formal”.

```
from sqlalchemy import create_engine
from sqlalchemy.orm import
sessionmaker from genera_tablas
import Saludo
# datos
# para el ejemplo se usa la base de
datos # sqlite
engine = create_engine('sqlite:///demobase2.db')
Session = sessionmaker(bind=engine)
session = Session()
# Obtener todos los registros de
# la tabla saludo que tengan como valor
en # el atributo tipo la expresión “formal”
losSaludos =
session.query(Saludo).filter(Saludo.tipo==”formal”) # la
consulta con .filter con argumento
# Saludo.tipo==”formal”
```

```
# Devuelve una lista de objetos de tipo
Saludo # y se le asigna como valor a la
variable
# losSaludos
# Se recorre la lista a través de un ciclo
# repetitivo for en python
for s in losSaludos: print("Mensaje: %s"
% (s.mensaje)) print("Tipo: %s" %
(s.tipo))
print("id: %s" % (s.id))
print("    ")
```

Respuesta:

```
Mensaje:  Buenos  días
Tipo: formal
id: 2
_____
Mensaje:  Buenas  noches  Tipo:
formal

id: 4
```

- Obtener todos los registros de la tabla saludo que tengan la vocal "o" en el atributo mensaje y sean de tipo "informal".

```
1 from sqlalchemy import create_engine
2 from sqlalchemy.orm import sessionmaker
3 from sqlalchemy import and_ # se importa el operador and
4 from genera_tablas import
Saludo 5
6 # datos
7 # para el ejemplo se usa la base de datos
8 # sqlite
9 engine =
create_engine('sqlite:///demobase2.db') 10
11 Session = sessionmaker(bind=engine)
12 session =
Session() 13
14 # Obtener todos los registros de·
15 # la tabla saludo que tengan la vocal "o" en el atributo mensaje
```

```
16 # y sean de tipo “informal”
17 losSaludos = session.query(Saludo).filter(and_(Saludo.mensaje.like(“%o%”), \
18 Saludo.tipo==”informal”))·
19 # a la consulta con .filter·
20 # se le agrega una expresión lógica AND
21 # la primera parte de la expresión usa el·
22 # operador like para determinar todos los registros
23 # que tengan la vocal “o” en el atributo mensaje;
24 # la segunda parte de la expresión, filtra todos·
25 # los registros que tienen como valor en el atributo tipo
26 # “informal”
27 # Devuelve una lista de objetos de tipo Saludo
28 # y se le asigna como valor a la variable
29 # losSaludos
 30
31 # Se recorre la lista a través de un ciclo
32 # repetitivo for en
python 33
34 for s in losSaludos:
35    print(“Mensaje: %s” % (s.mensaje))
36     print(“Tipo: %s” % (s.tipo))
37     print(“id: %s” % (s.id))
38     print(“               ”)
```

Respuesta:

```
 Mensaje: Hola que tal Tipo:
 informal
id: 1
```