

Programación Avanzada

Guía didáctica



Unidad Académica Técnica y Tecnológica

Tecnología Superior en Transformación Digital de Empresas

Programación Avanzada

Guía didáctica

Carrera	PAO Nivel
▪ <i>Tecnología Superior en Transformación Digital de Empresas</i>	III

Autor:

Elizalde Solano René Rolando



D S O F _ 2 0 4 9

Asesoría virtual
www.utpl.edu.ec

Universidad Técnica Particular de Loja

Programación Avanzada

Guía didáctica

Elizalde Solano René Rolando

Diagramación y diseño digital:

Ediloja Cía. Ltda.

Telefax: 593-7-2611418.

San Cayetano Alto s/n.

www.ediloja.com.ec

edilojacialtda@ediloja.com.ec

Loja-Ecuador

ISBN digital - 978-9942-39-804-8



Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional (CC BY-NC-SA 4.0)

Usted acepta y acuerda estar obligado por los términos y condiciones de esta Licencia, por lo que, si existe el incumplimiento de algunas de estas condiciones, no se autoriza el uso de ningún contenido.

Los contenidos de este trabajo están sujetos a una licencia internacional Creative Commons – **Reconocimiento-NoComercial-CompartirIgual 4.0 (CC BY-NC-SA 4.0)**. Usted es libre de **Compartir** – copiar y redistribuir el material en cualquier medio o formato. **Adaptar** – remezclar, transformar y construir a partir del material citando la fuente, bajo los siguientes términos: **Reconocimiento** – debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciante. **No Comercial** – no puede hacer uso del material con propósitos comerciales. **Compartir igual** – remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original. No puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otras a hacer cualquier uso permitido por la licencia. <https://creativecommons.org/licenses/by-nc-sa/4.0/>

Índice

1. Datos de información.....	7
1.1. Competencias genéricas de la UTPL.....	7
1.2. Competencias específicas de la carrera	7
1.3. Problemática que aborda la asignatura	7
2. Metodología de aprendizaje.....	8
Primer bimestre.....	9
Resultado de aprendizaje 1.....	9
Contenidos, recursos y actividades de aprendizaje.....	9
Semana 1	9
 Unidad 1. Ciclo de vida de software (ISO/IEC 12207:2017)	9
1.1. Etapas del ciclo de vida.....	9
1.2. Modelos del ciclo de vida.....	10
1.3. Ventajas de los modelos de ciclo de vida	13
Semana 2	14
1.4. Gestión del modelo de ciclo de vida	14
1.5. Estándar ISO/IEC 12207:2017	19
Actividades de aprendizaje recomendadas	21
Autoevaluación 1	22
 Resultado de aprendizaje 2.....	25
Contenidos, recursos y actividades de aprendizaje.....	25
 Unidad 2. Desarrollo de aplicaciones con base de datos	25
Semana 3	25
2.1. Control de versiones.....	25
Semana 4	32
2.2. Control de versiones.....	32
Actividades de aprendizaje recomendadas	44
Autoevaluación 2.....	45

Semana 5	49
2.3. Programación con bases de datos	49
Semana 6	62
2.4. Programación con bases de datos	62
Actividades de aprendizaje recomendadas	73
Autoevaluación 3	75
Semana 7	80
Semana 8	83
Actividades finales del bimestre	83
Segundo bimestre	85
Resultado de aprendizaje 3.....	85
Contenidos, recursos y actividades de aprendizaje.....	85
Semana 9	85
Unidad 3. Conceptos avanzados de programación	85
3.1. Programación en capas	85
Semana 10	90
3.2. Programación en capas	90
Actividades de aprendizaje recomendadas	98
Autoevaluación 4	100
Semana 11	103
3.3. Programación multihilo	103
Semana 12	107
3.4. Programación multihilo	107
Actividades de aprendizaje recomendadas	112
Autoevaluación 5	113

Semana 13	117
3.5. Programación de interfaces e interactiva	117
Semana 14	120
3.6. Programación de interfaces e Interactiva	120
Actividades de aprendizaje recomendadas	134
Autoevaluación 6.....	135
Semana 15	139
Semana 16	145
Actividades finales del bimestre	145
3. Solucionario	147
4. Referencias bibliográficas	156
5. Anexos	157



1. Datos de información

Presentación de la asignatura



1.1. Competencias genéricas de la UTPL

- Vivencia de los valores universales del humanismo de Cristo.

1.2. Competencias específicas de la carrera

- Desarrolla aplicaciones empresariales aplicando enfoques centrados en la nube.

1.3. Problemática que aborda la asignatura

La asignatura de Programación Avanzada aborda la problemática de cómo el desarrollo de aplicaciones web y móviles, el análisis de datos, la aplicación de técnicas se complementa con la arquitectura de software para proporcionar a los estudiantes las habilidades y herramientas necesarias para desarrollar soluciones tecnológicas innovadoras que

permitan a las empresas competir y adaptarse en un entorno empresarial cada vez más digitalizado.



2. Metodología de aprendizaje

Para el desarrollo de los contenidos en la asignatura en el proceso de enseñanza aprendizaje, 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 consideración personalizada que ayuda en la adquisición de los resultados de aprendizaje propuestos. Para revisar la información relacionada con la metodología descrita, usted puede ingresar al enlace que [contiene los aspectos teóricos más importantes](#).



Primer bimestre

Resultado de aprendizaje 1

- Explica el ciclo de vida del software.

El resultado de aprendizaje busca la comprensión del proceso establecido en el ciclo de vida del software, a través del estudio de conceptos y ejemplificaciones de cada fase que involucre la construcción de un software. Se conocerá los estándares establecidos y su evolución en la creación de software. Finalmente, se conocerá la diferencia entre las metodologías de desarrollo tradicional y metodologías de desarrollo ágiles.

Contenidos, recursos y actividades de aprendizaje



Semana 1

Unidad 1. Ciclo de vida de software (ISO/IEC 12207:2017)

1.1. Etapas del ciclo de vida

Iniciar el estudio del ciclo de vida de software, invita a reflexionar sobre algunos inconvenientes que se han presentado en la historia reciente en cuanto al desarrollo de software. Martínez (2014), indica algunos estudios realizados por empresas como [Standish Group](#) y similares, donde se desprenden algunas situaciones como causas que han afectado de manera negativa, tal como:

- Desde niveles superiores de la empresa, se realizan cambios en los objetivos planificados.
- Poca o nula utilización de un proceso riguroso en metodologías de trabajo para los equipos de desarrollo.
- Conflictos personales en los grupos, que conlleva a la generación de problemas humanos.

Las diversas metodologías aplicadas al ciclo de desarrollo de software, buscan eliminar estos errores.

De acuerdo a lo manifestado por Ojeda y Fuentes (2012), es muy común que exista confusión entre las frases: proceso de desarrollo de software y ciclo de vida de software; pero, las expresiones tienen el mismo significado. A través del proceso de desarrollo de software se realizan actividades como: especificaciones de requerimientos, diseño, codificación, validación y mantenimiento. El objetivo de las actividades anteriores es poder describir la vida de un producto o proyecto de software. Existe una comunicación efectiva, mediante entrevistas o instrumentos similares, entre el cliente como quien solicita una solución y el líder del equipo de desarrollo quien diseña y propone una solución ante la problemática planteada.

1.2. Modelos del ciclo de vida

Los modelos del ciclo de vida de software indican el orden en el que se van a efectuar las actividades descritas en el punto anterior. A un modelo se lo puede denominar como *paradigma del proceso*.

De acuerdo a lo indicado por Ojeda y Fuentes (2012), existe una clasificación de tres modelos o paradigmas: modelo en cascada, modelos de desarrollo evolutivo, modelos de componentes reutilizables.

a. Modelo en cascada

El proceso se lo lleva adelanta mediante fases separadas y secuenciales; a este modelo algunos autores lo denominan *modelo lineal*.

Las fases son las siguientes:

- Especificación.
 - Análisis y definición de requerimientos.
- Implementación.
 - Diseño.
 - Codificación.
 - Validación.

- Mantenimiento.

Revisar la información de la tabla 1, donde se detallan algunas acciones por cada fase.

Tabla 1.

Fases del modelo en cascada.

Fases	Acciones
Especificación - Análisis y definición de requerimientos.	<ul style="list-style-type: none"> Se busca encontrar el dominio de la aplicación, mediante conversaciones con los clientes y usuarios finales. Se genera el documento denominado: especificación de requerimientos del sistema.
Diseño.	<ul style="list-style-type: none"> Se obtiene los requerimientos de <i>software</i> y <i>hardware</i>. Se realiza la primera aproximación de la arquitectura del sistema. Es posible la identificación de subsistemas.
Codificación.	<ul style="list-style-type: none"> Pasar a código lo expresado en el diseño por cada subsistema. Generar diversas pruebas para verificar que se cumpla con lo solicitado.
Validación.	<ul style="list-style-type: none"> Unir los subsistemas y generar pruebas de validación para verificar los requerimientos de los proyectos.
Mantenimiento.	<ul style="list-style-type: none"> Se pone en producción el sistema para que sea validado por los usuarios finales. Como parte del mantenimiento se debe corregir algún error que se pueda encontrar. No se debe confundir con el hecho que existan nuevas funcionalidades solicitadas.

Nota. Elizalde R, 2023

b. Modelos de desarrollo evolutivo.

Estos modelos se los denomina *iterativos*. El objetivo es generar versiones más completas en cada iteración, en función de las actividades de especificación, desarrollo y validación. Las primeras versiones se las realiza con los requerimientos que tengan mayor prioridad. De acuerdo a Ojeda y Fuentes (2012), se puede hablar de dos tipos, revisar tabla 2:

Tabla 2.*Tipo de desarrollo evolutivo.*

Tipo de desarrollo evolutivo	Acciones
Desarrollo exploratorio.	La versión que se presenta al usuario, tiene como base los requerimientos que han sido entendidos completamente por el equipo de desarrollo. El objetivo es recibir la retroalimentación para afinar la solución de la próxima iteración.
Prototipos desecharables.	Cuando se tiene duda en la comprensión de los requerimientos, se puede generar un prototipo con una simulación de la funcionalidad.

Nota. Elizalde R, 2023

c. Modelos de componentes reutilizables

En este tipo de modelos, la idea es reusar ideas, arquitecturas, diseños, código de una aplicación previa funcional. El reto es tratar de integrar a la nueva solución estos módulos, sin que se vea afectado el sistema principal. Existen algunas etapas definidas en el modelo según Ojeda y Fuentes (2012), ver tabla 3.

Tabla 3.*Etapas de componentes reutilizables.*

Etapa	Función
Análisis de componentes.	Se busca algún componente previamente desarrollado, que ayude en la construcción de la especificación de requerimientos.
Modificación de requerimientos.	Los posibles componentes encontrados para reutilizar, deben adaptarse a los requerimientos de la solución. Si no hay una adaptación, se debe modificar los requerimientos, siempre que no se salga del objetivo final de la solución.
Diseño del sistema con reutilización.	Se realiza una planificación tomando en consideración los componentes que se reutilizan junto a los componentes nuevos.
Desarrollo e integración.	Todos los componentes que se tengan que desarrollar se los debe integrar a los componentes reutilizados.

Nota. Elizalde R, 2023

Martínez (2014), realiza otra clasificación de modelos de ciclo de vida, tomando en consideración la relevancia e impacto. Se la detalla a continuación:

- **Modelo en cascada**, se conoce como el ciclo de vida básico o lineal. Las fases que incluye son: análisis y definición de requisitos; diseño del sistema; implementación y pruebas de unidad; integración y prueba del sistema, y funcionamiento y mantenimiento.
- **Modelo en V**, en el proceso usa las mismas etapas que el modelo en cascada, con la principal diferencia que se adelanta el proceso de pruebas. En el modelo en cascada es posible que los errores se los encuentre tarde en el ciclo de vida; lo que conlleva problemas para la entrega de la solución en el tiempo acordado.
- **Modelo iterativo**, trata de reducir los posibles problemas que puedan existir, por no cumplir con las expectativas que el usuario final tenga del producto desarrollado. Se genera procesos iterativos de un conjunto de ciclos en cascada. Se puede usar este modelo, cuando no existe una compresión lógica inicial de los requisitos.
- **Modelo incremental**, en este modelo se empieza a presentar soluciones muy básicas, pero funcionales. A medida que se avanza en la solución se va incrementando las funcionalidades.
- **Modelo en espiral**, trata de adquirir madurez mediante ciclos repetitivos hasta lograr la entrega final. Se asume un ciclo por cada fase: especificaciones de requerimientos, diseño, codificación, validación y mantenimiento.
- **Modelo ágil**, en este modelo intervienen el desarrollo iterativo e incremental. Se basa en el trabajo en equipo, organización y responsabilidad. Se busca la entrega rápida de software de calidad. Entre las características se tiene: incertidumbre, equipos auto-organizados, autonomía, fases del desarrollo sobrepuertas, control no estricto.

1.3. Ventajas de los modelos de ciclo de vida

Utilizar de manera adecuada el ciclo de vida a través de cualquiera de los modelos presentados, seguro traerá algunas ventajas en el desarrollo de soluciones de software. A continuación, se describen algunas consideraciones, según lo expresado por Martínez (2014), Ojeda y Fuentes (2012).

- Tener los requisitos comprendidos permite generar una solución robusta, aplicando un modelo del ciclo de vida.
- El equipo de desarrollo está preparado para los cambios de requisitos.
- Se cubren carencias de los desarrolladores.
- A través de las fases se trata vincular a los desarrolladores con los usuarios finales.
- Se puede generar mejores estimaciones para el desarrollo.
- Se adapta a proyectos pequeños, medianos y grandes.
- Se puede realizar una distribución eficiente del personal.
- Entrega rápida de la solución funcional.
- Se puede detectar errores en tempranas fases, que eviten pérdidas al final.
- De acuerdo al tamaño del proyecto, se puede adaptar.

Avancemos a la siguiente semana de estudio.



Semana 2

1.4. Gestión del modelo de ciclo de vida

Es importante mencionar que los modelos de desarrollo de software y las metodologías de desarrollo no tienen el mismo concepto. Las metodologías dan una estructura al desarrollo de software, cumpliendo con las fases, a través de la aplicación de un conjunto notaciones, reglas, diagramas, etc.

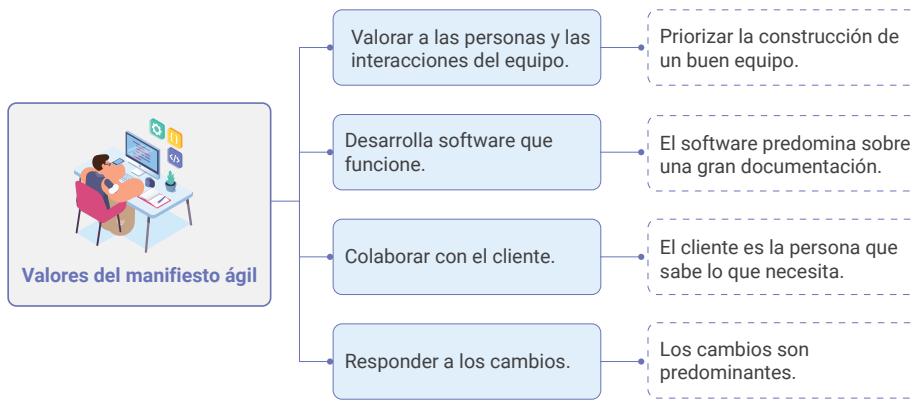


En este apartado se va a detallar algunas metodologías de desarrollo ágiles que existen y que puede ser usadas en proyectos. Para ello se sugiere que exista una lectura comprensiva del recurso denominado [Ciclo de vida de desarrollo ágil de software seguro](#), en lo que relaciona a la unidad dos “El desarrollo ágil de software”.

En el recurso, se realiza una introducción conceptual al manifiesto ágil. Dicho manifiesto tiene como objetivo generar software de calidad, en

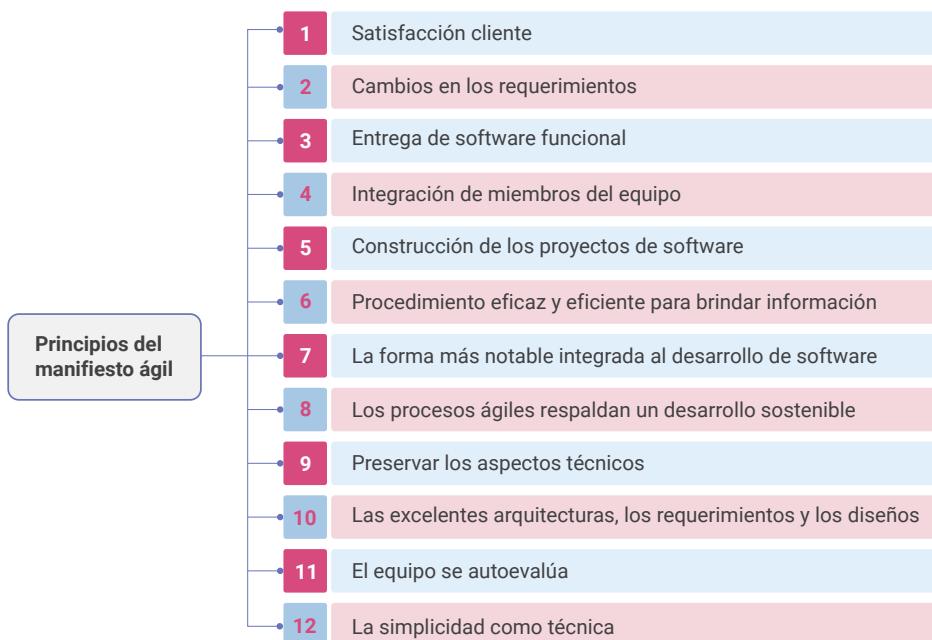
tiempo cortos y que se adapte a cambios de requisitos; para alcanzar su objetivo emite valores, ver figura 1, y principios, ver figura 2, que todos deben seguir.

Figura 1.
Valores del manifiesto ágil



Nota. Adaptado de *Ciclo de vida de desarrollo ágil de software seguro* (p. 21), por Hernández Bejarno, M., 2020, Fundación Universitaria Los Libertadores.

Figura 2.
Principios del manifiesto ágil



Nota. Adaptado de *Ciclo de vida de desarrollo ágil de software seguro* (p.22), por Hernández Bejarno, M., 2020, Fundación Universitaria Los Libertadores.

La finalidad y objetivo del proyecto tiene mucha importancia en la selección de la metodología ágil. Este tipo de metodologías permiten enfrentar posibles cambios en los requerimientos en etapas tempranas. Se requiere que el cliente sea parte del proceso a través de una comunicación efectiva.

El recurso educativo menciona algunas características que se deben considerar para la selección de una metodología ágil:

- Experiencia de la empresa que asume el proceso de desarrollo.
- Experiencia del equipo de desarrollo.
- El número de personas que conforman el equipo, con relación a la extensión del proyecto.

Las metodologías ágiles resaltan el trabajo en equipo, por tal razón, existe un mayor protagonismo por parte de los programadores y clientes; y la interacción con el líder del proyecto aumenta.

A continuación, se detalla algunas metodologías ágiles y sus características:

- **Metodología Scrum**

Busca generar dinamismo en el desarrollo de los proyectos, a través de equipos autoorganizados, comunicación constante; mediante roles y prácticas. Se resalta el hecho que Scrum tiene como base los ciclos de vida iterativo e incremental; generando entregas periódicas, buscando incrementar las funcionalidades respecto a la entrega anterior.

Existen algunos conceptos dentro de esta metodología, ver tabla 4, donde se describen las características más importantes:

Tabla 4.

Conceptos de la metodología Scrum.

Concepto	Características
<i>Sprint.</i>	<ul style="list-style-type: none">▪ Proceso en el cual se realiza un avance del proyecto.▪ El promedio de extensión de tiempo de un Sprint es de 4 semanas.▪ A través de reuniones se puede planificar, revisar y retroalimentar un Sprint.

Concepto	Características
Product backlog.	<ul style="list-style-type: none"> Está conformado por un conjunto de requisitos funcionales o historias de usuario. Los <i>sprint</i> toman información del Product backlog para trabajar.
Rol: Scrum master.	<ul style="list-style-type: none"> La persona que asume liderato del proyecto; entre sus funciones está el llevar adelante el proceso de desarrollo de forma adecuada y buscar soluciones a los diversos problemas que se pueda enfrentar el equipo.
Rol: Product Owner.	<ul style="list-style-type: none"> Es el encargado de mantener una comunicación fluida entre los clientes y el equipo Scrum. Debe estar pendiente de los posibles cambios en los requisitos funcionales y no funcionales.
Rol: Equipo Scrum.	<ul style="list-style-type: none"> Lo conforman los programadores y equipo calidad, quienes están en capacidad de plasmar las especificaciones de negocio de la solución.
Burndown chart.	<ul style="list-style-type: none"> Proceso para visualizar el avance y pendientes del proyecto en construcción.

Nota. Elizalde R, 2023

▪ **Metodología XP**

Esta metodología posee características como: sencillez, comunicación y retroalimentación. Se basa en cambios incrementales, calidad de software y *feedback* (comentarios).

Los roles que destacan en la metodología son:

- Programador.
- Cliente.
- Tester (quien se encarga de las pruebas).
- Tracker (quien tiene como objetivo dar seguimiento a los cambios sugeridos).
- Coach.
- Consultor y gestor.

Hay cuatro fases que conforman la metodología, ver tabla 5.

Tabla 5.*Fases de la metodología XP.*

Fases	Actividades
Fase de exploración.	<ul style="list-style-type: none"> ▪ Se detalla el alcance del proyecto. ▪ El cliente explica lo que necesita como resultado del proyecto, mediante las historias de usuarios. ▪ Se estima los tiempos necesarios de desarrollo por parte de los programadores.
Fase de planificación.	<ul style="list-style-type: none"> ▪ Es un escenario muy corto en tiempo. ▪ Se generan reuniones entre todos los involucrados, lo que permitirá identificar el orden en el que serán implementadas las historias de usuario.
Fase de iteraciones.	<ul style="list-style-type: none"> ▪ Es la fase de mayor importancia en el proceso de la metodología ▪ Se generan las soluciones funcionales. ▪ Cada iteración genera una solución en función de la historia de usuario asignada. ▪ Se resalta la participación activa del cliente en toda la fase.
Fase de puesta en producción.	<ul style="list-style-type: none"> ▪ El objetivo de la fase es la de entregar módulos totalmente funcionales y libre de errores. ▪ Existe la posibilidad de hacer algunos ajustes.

Nota. Elizalde R, 2023

- **Metodología Kanban**

Metodología que inició en la empresa [TOYOTA](#), le permitió mantener el control de una línea de producción en sus instalaciones. Se maneja los conceptos como: desarrollo incremental, historias de usuario, tareas principalmente.

En el proceso la metodología se hace uso de un tablero denominado [KANBAN](#), que tiene como prioridad exponer a los miembros del equipo de desarrollo una idea visual de la actualidad del software.

A continuación, se presentan algunos beneficios de aplicar la metodología:

- Se incrementa y fomenta la organización y colaboración.
- Se genera espacios de trabajo eficaces y confianza.
- Existe y se consolida una comunicación horizontal entre los miembros del equipo.

En la tabla 6, se presenta una comparación entre las metodologías expuestas, de acuerdo a lo indicado en el recurso educativo.

Tabla 6.

Comparativa de metodologías ágiles. Adaptada de Hernández (2020).

Características	Metodologías		
	XP	Scrum	Kanban
Cumplimiento de los requisitos.	Sí.	Sí.	Sí.
Aumento de productividad.	Sí.	Sí.	Sí.
Iteraciones cortas.	Sí.	Sí.	Sí.
Centrado en los usuarios.	Sí.	Sí	Sí.
Integración de cambios.	Sí.	Sí.	Sí
Los requerimientos funcionales pueden cambiar.	Sí.	Sí.	Sí.
El plan de trabajo puede variar.	Sí.	No.	Sí.
Intercambio de conocimiento.	Alto.	Bajo.	Bajo.
Tamaño de proyectos.	Pequeño.	Pequeño/ Grande.	Pequeño.
Tamaño del equipo.	Pequeño.	Pequeño.	Pequeño.
Grado de iteración en el equipo de trabajo.	Alto.	Alto.	Alto.
Organización del equipo.	Autoorganizado.	Autoorganizado.	Autoorganizado.
Definición de requisitos.	Sí.	Sí.	Sí.
Modelado.	Sí.	Sí.	Sí.
Codificación.	Sí.	Sí.	Sí.
Pruebas unitarias.	Sí.	Sí.	Sí.
Pruebas de integración.	Sí.	Sí.	Sí.

Nota. Elizalde R, 2023

1.5. Estándar ISO/IEC 12207:2017

De acuerdo a lo indicado por Gómez (2018), en su documento, el estándar es tratado como marco de trabajo común para realizar las diversas actividades del ciclo de vida del software; se abarca de forma conjunta desde las actividades iniciales hasta las finales en el proceso. Su objetivo es que sea entendido por todos los involucrados en el proceso.

En la información dada por Vázquez-Ingelmo (2019), se hace una descripción de a quienes está dirigido el estándar, aquí se nombra a los que adquieren software, productos, suministros, mantenimiento de un sistema.

En la tabla 7 se indican algunas características del estándar desde su creación, de acuerdo a lo mencionado por Gómez (2018).

Tabla 7.

Historia de ISO/IEC 12207:2017.

Año de publicación	Características
1995.	<ul style="list-style-type: none">▪ Generar una hoja de ruta que sea replicable en aspectos relacionados con: adquirir, desarrollar, operar productos de software.▪ Listado de actividades a considerar para que procesos generados en cada grupo desarrollo se adapten al proceso.▪ El estándar se divide en 3 grupos y 17 procesos.
2008.	<ul style="list-style-type: none">▪ Se lo utiliza principalmente en conjunto con el estándar ISO/IEC 15288.▪ Adapta sus procesos para que puedan ser evaluados por la ISO/IEC 15504-2.▪ El estándar se divide en 7 grupos y 43 procesos.
2017.	<ul style="list-style-type: none">▪ Se simplifica el proceso, a través de la unión de ideas con la ISO/IEC 15288.

Nota. Elizalde R, 2023

Los procesos bajo el estándar ISO/IEC 12207:2017, de acuerdo a lo indicado por Gómez (2018), se los puede revisar en la siguiente infografía:

[**Procesos en ISOIEC 12207-2017**](#)

En Gómez (2018), se realiza una descripción de cada proceso:

- Los procesos de acuerdo, son realizados en el ámbito fuera de la vida del proyecto.
- Los procesos organizacionales se relacionan con identificar las formas para que el proyecto cumpla con las necesidades.
- Los procesos de gestión técnica, sus actividades se enfocan en articular lo realizado en los procesos de acuerdo y procesos organizacionales.

- Los procesos técnicos son el conjunto de acciones que se van a ejecutar a lo largo de la construcción de la solución.



Actividades de aprendizaje recomendadas

- **Actividad 1:** revisar el siguiente recurso denominado [Los Roles de Scrum](#). Luego de analizar la exposición dada en el recurso, se evidencia un resumen de las principales tareas que realizan las personas, según el rol en un proyecto Scrum. Los roles que se abordan en el recurso son: Product Owner, Scrum Master, Equipo. Se sugiere efectuar un organizador gráfico de lo propuesto en el recurso para un mejor entendimiento.
- **Actividad 2:** revisar el siguiente recurso denominado [SCRUM: cuáles son sus características](#). En el recurso sugerido, usted puede reforzar el entendimiento de las principales características de la metodología SCRUM. Entre algunas de las características que se explican están: entrega programada de los resultados, manejo de las expectativas del cliente, flexibilidad, armonía entre el cliente y el equipo de desarrollo. Se recomienda realizar un organizador gráfico de lo propuesto en el recurso para un mejor entendimiento y posterior estudio.
- **Actividad 3:** resolver la autoevaluación 1; en la actividad se presentan un conjunto de preguntas basadas en las temáticas descritas en la unidad 1: ciclo de vida de software (ISO/IEC 12207:2017). En la parte introductoria de la actividad se listan los contenidos de la guía didáctica, que se debe revisar para contestar las interrogantes. Recuerde que la autoevaluación le permitirá identificar el nivel de aprendizaje y comprensión alcanzado; así mismo, determinar los ítems que se debe volver a revisar para lograr los resultados de aprendizaje propuestos.

Adicionalmente, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Finalmente, después de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección Solucionario de la guía didáctica.



Autoevaluación 1

Es momento de medir el entendimiento de algunos conceptos estudiados en la unidad 1: ciclo de vida de software (ISO/IEC 12207:2017). Antes de realizar la autoevaluación, se requiere las siguientes acciones:

- Revisar el contenido de las temáticas indicadas en la unidad.
- Leer y analizar los recursos educativos recomendados.
- Hacer organizadores gráficos para la organización de la información.

La autoevaluación le permitirá reafirmar los conceptos estudiados. Los enunciados tienen una sola opción correcta, lea de forma atenta cada respuesta y seleccione la opción que usted considere correcta.

1. Identifique las actividades del proceso de desarrollo de software

- a. Especificaciones de requerimientos, diseño, codificación, validación y mantenimiento.
- b. Especificaciones de requerimientos, codificación, validación y mantenimiento.
- c. Diseño, codificación, validación y mantenimiento.

2. ¿Cuál es otra forma para denominar a los modelos del ciclo de vida del software?

- a. Paradigma de ambiente web.
- b. Paradigma del lenguaje de programación.
- c. Paradigma del proceso.

3. Identifique los tipos de modelos del ciclo de vida

- a. Modelo en funcional, modelos de desarrollo evolutivo, modelos de componentes reutilizables.
- b. Modelo en cascada, modelos de desarrollo evolutivo, modelos orientados a objetos.
- c. Modelo en cascada, modelos de desarrollo evolutivo, modelos de componentes reutilizables.

- 4. A nivel del modelo en cascada, identifique las fases del modelo en cascada**
 - a. Especificación (análisis y definición de requerimientos); implantación (diseño, codificación, validación), mantenimiento.
 - b. Discusión (análisis y definición de requerimientos); implantación (diseño, codificación, validación), mantenimiento.
 - c. Especificación (análisis y definición de requerimientos); implantación (diseño, codificación, validación), pruebas.
- 5. Identifique la fase del modelo en cascada a la que pertenece la acción que aborda la realización de la primera aproximación de la arquitectura del sistema.**
 - a. Especificación.
 - b. Diseño.
 - c. Implementación.
- 6. Identifique la fase del desarrollo evolutivo a la que pertenece la acción que permite generar un prototipo con una simulación de la funcionalidad.**
 - a. Prototipos desechables.
 - b. Prototipos exploratorios.
 - c. Desarrollo exploratorio.
- 7. Identifique dos ventajas de los modelos de ciclo de vida**
 - a. El equipo de desarrollo está preparado por los cambios de requisitos; a través de las fases se trata vincular a los desarrolladores con los usuarios finales.
 - b. Se adapta a trabajar en proyectos exclusivamente pequeños; se puede realizar una distribución eficiente del personal.
 - c. Se puede detectar en etapas finales errores, que eviten pérdidas al inicio; entrega rápida de la solución funcional.

- 8. Identifique los valores del manifiesto ágil**
- a. Valorar a las personas y evitar las interacciones del equipo; desarrolla *software* que funcione; colaborar con el cliente; responder a los cambios.
 - b. Valorar a las personas y las interacciones del equipo; desarrolla *software* que funcione; colaborar con el cliente; responder de forma tardía a los cambios.
 - c. Valorar a las personas y las interacciones del equipo; desarrolla *software* que funcione; colaborar con el cliente; responder a los cambios.
- 9. Identifique las metodologías ágiles, de las siguientes opciones.**
- a. Metodología Scrum; metodología XP; metodología Kanban.
 - b. Metodología Scrum; metodología XP; metodología funcional y reactiva.
 - c. Metodología Scrum; metodología Open Source; metodología Kanban.
- 10. Identifique los grupos de procesos en los que está dividido la ISO/IEC 12207:2017**
- a. Procesos de acuerdo / procesos de gestión técnica / procesos técnicos.
 - b. Procesos de acuerdo / procesos organizacionales / procesos de gestión técnica / procesos técnicos.
 - c. Procesos de acuerdo / procesos organizacionales / procesos de gestión de *hardware*/ procesos técnicos.

¡Hemos concluido la semana de estudio número dos, continuemos!

[Ir a solucionario](#)

Resultado de aprendizaje 2

Comprende el alcance y las capas constitutivas de una solución completa de software y las tecnologías digitales que las soportan.

El resultado de aprendizaje busca formar al estudiante en conceptos relacionados con el desarrollo de soluciones completas; para ello, las siguientes semanas de estudio tendrán relación con temáticas que son empleadas en el desarrollo de software. Los contenidos relacionados a control de versiones y programación con base de datos relacionales y no relacionales; están estructurados desde el punto de vista teórico y práctico; para una mayor comprensión.

Contenidos, recursos y actividades de aprendizaje

Unidad 2. Desarrollo de aplicaciones con base de datos



Semana 3

2.1. Control de versiones

Estimado estudiante, para el estudio y comprensión de las temáticas de la semana, se hará uso del recurso educativo abierto denominado [Pro Git](#). El recurso posee amplia información que permitirá una comprensión de lo propuesto en la presente sección de contenido.

2.1.1. Fundamentos de control de versiones

En el [apartado 1.1 de recurso denominado Inicio - Sobre el Control de Versiones - Acerca del Control de Versiones](#), se parte de una pregunta muy importante ¿Qué es un control de versiones, y por qué debería ser importante? En la actualidad, generar aplicaciones de código y no usar control de versiones, implica poner en riesgo la productividad del equipo o grupo de desarrollo. Un control de versiones, se convierte en un mecanismo o proceso que permite registrar en el tiempo el conjunto de cambios que se

ha realizado para uno o varios archivos, lo anterior permite que se pueda recuperar de manera exacta la versión en un día y hora específica.

Entre las ventajas de un control de versiones se tiene las siguientes:

- Emplear versiones anteriores de los archivos.
- Ejecuta comparaciones de forma muy sencilla entre la versión actual de un archivo y una versión anterior.
- Identificar la persona(s) que modificó un archivo.

De acuerdo a lo descrito en el recurso educativo, se puede distinguir tres tipos de sistemas de control de versiones:

- **Sistema de control de versiones locales;** son sistemas que no se utilizan de manera frecuente en la actualidad. Se caracterizan por tener una base de datos local, quien era la encargada de llevar el control de los cambios realizados en los archivos. Una de las soluciones más usadas en este tipo de sistemas fue [Revision Control System \(RCS\)](#).
- **Sistema de control de versiones centralizados;** los sistemas centralizados han tenido a lo largo de la historia reciente grandes exponentes como: [Subversion](#) y [Perforce](#). La característica principal aquí es la de tener colaboradores de un proyecto en varios lugares. Existe un servidor único que mantiene los archivos de un proyecto, desde donde los miembros del proyecto pueden descargar los archivos. Se puede conocer, especialmente el administrador, el detalle del trabajo realizado y los niveles de acceso a cada parte del código. Entre la desventaja que se destaca, es la posible pérdida total de la información, a causa del fallo del servidor único; es por eso que en este tipo de sistemas se aconseja el uso de frecuentes respaldos.
- **Sistema de control de versiones distribuidos;** los mayores exponentes en este tipo de sistemas son: [Git](#), [Mercurial](#) y [Bazaar](#). La distribución, marca diferencia de los anteriores sistemas, aquí los colaboradores de un proyecto de software, tienen la posibilidad de descargar en sus entornos locales la versión más reciente del proyecto, además, se replica de forma total el repositorio. Lo mencionado permite replicar o restaurar un servidor en caso de algún daño, con la información de los repositorios usados por los clientes o colaboradores.

2.1.2. Conceptos de Git



Durante el estudio de la asignatura se ha decidido emplear un sistema de control de versiones distribuido, llamado Git. Se solicita estimado estudiante que pueda realizar una lectura comprensiva de la información [1.2 Inicio - Sobre el Control de Versiones - Una breve historia de Git](#), y [1.3 Inicio - Sobre el Control de Versiones - Fundamentos de Git](#); del recurso educativo

El creador de Git fue [Linus Torvalds](#), quien decidió efectuar una herramienta de control de versiones para manejar el código del núcleo de [Linux](#) en el 2005. Entre los principales objetivos que se impulsó con Git, fueron:

- Velocidad.
- Diseño sencillo.
- Soporte.
- Distribuido.
- Manejo de grandes proyectos.

En el recurso educativo se invita a entender el concepto de Git, dejar de lado los conceptos/ideas que se tiene otros sistemas de control de versiones no distribuidos como subversión, con el objetivo de no tener confusiones.

Git es un sistema que permite capturar el estado de todos los archivos del proyecto, al momento de realizar un cambio en cualquier archivo. La eficiencia radica en que, si un archivo no fue modificado, no es necesario generar una nueva instancia, si no usar la referencia que ya exista.

Cuando se usa Git, las operaciones que se realicen se las puede hacer de forma local. Por ejemplo: revisar el historial de cambios de los archivos, guardar las nuevas versiones de los archivos modificados. No es necesario conectarse al servidor de forma obligatoria como se requiere en otros sistemas de versiones.

Git posee un mecanismo de integridad de información, se hace uso de una suma de comprobación, lo que permite que el sistema de control de

versiones se mantenga informado de lo que sucede con los archivos en todo momento. La suma de comprobación se la conoce como hash SHA-1; cadena hexadecimal de una longitud de 40 caracteres.

Ejemplo de una cadena tipo hash SHA-1:

- 56745ee7053904d24b5a5ce7394b2bdac83c513e.

Los mecanismos que posee Git permiten tener la seguridad que los archivos en sus diferentes versiones, estarán disponibles, y es muy difícil que se puedan borrar o perder la información. Lo comentado, ayuda a los desarrolladores nuevos, pues se puede experimentar sin la preocupación de generar problemas.

En Git existen tres estados posibles para los archivos de los proyectos. Es importante que se tengan en consideración los mismos. A continuación, se lista los posibles estados:

- **Confirmado** (committed): en este estado los archivos con su información están almacenados de forma segura en la base de datos local.
- **Modificado** (modified): indica que los archivos están modificados, pero aún no forman parte de la base de datos local.
- **Preparado** (staged): cuando los archivos son marcados en su actual versión, dejando los mismos listos para la próxima confirmación.

Por otro lado, un proyecto de Git formalmente tiene tres secciones.

- Directorio de Git (Git directory).
- Directorio de trabajo (working directory).
- Área de preparación (staging área).

Con relación al Directorio de Git, se lo considera como la parte más relevante en el proceso, aquí se guardan los metadatos y la base de datos del proyecto. El Directorio de trabajo tiene formalmente una copia de una versión del proyecto; se los obtiene de la base de datos y se hacen accesibles desde el disco duro de la máquina en donde se trabaja. Finalmente, el área de preparación, contiene todos los datos o información que constará en el siguiente proceso de confirmación.

El flujo de trabajo (básico) de Git, se resume en los siguientes pasos:

- **Paso 1:** cuando se realizan modificaciones a los archivos en el directorio de trabajo.
- **Paso 2:** en este paso se agregan los archivos al área de preparación.
- **Paso 3:** se confirman los cambios, pasa los archivos en el estado que indica el área de preparación, almacenando una copia de forma permanente en el directorio.

2.1.3. Comandos de Git



Para este apartado, se solicita realizar una lectura comprensiva de las siguientes secciones del recurso educativo: [1.4 Inicio - Sobre el Control de Versiones - La Línea de Comandos](#), [1.5 Inicio - Sobre el Control de Versiones - Instalación de Git](#), [1.6 Inicio - Sobre el Control de Versiones - Configurando Git por primera vez](#).

En las etapas iniciales de formación académica, se recomienda el uso de la línea de comandos para trabajar con Git. Como primer paso para poder trabajar en las computadoras personales con este sistema de control de versiones, hay que realizar el proceso de instalación. De acuerdo al sistema operativo, se comparte los enlaces para realizar el proceso de instalación:

- [Instalación en sistema operativo Linux](#).
- [Instalación en sistema operativo Windows](#).
- [Instalación en sistema operativo Mac](#).

Es importante estimado estudiante que tenga instalado Git en los entornos locales, seguir los enlaces mencionados para aquello.

El siguiente paso luego de la instalación, es la configuración. Se lo realiza a través de la herramienta **git config**. Lo cual permite generar valores para situaciones como: identidad, editor, etc. El proceso para generar la información con relación al nombre de usuario y correo electrónico que se usa al momento de registrar un commit, son los siguientes:

- `git config --global user.name "Nombre de Usuario".`
- `git config --global user.email nombre@ejemplo.com.`

La bandera **-- global**, permite asignar dichos valores a la configuración, por una sola oportunidad.

A continuación, se listan algunos comandos para poder trabajar con Git desde la línea de comandos.

- `git -help.`
- `git config -list .`
- `git status.`
- `git add.`
- `git comit.`
- `git push.`
- `git pull.`
- `git log.`

2.1.4. Creación de repositorios de versiones

Para la presente temática, se solicita revisar la información que se detalle en el recurso educativo en la [sección 2.1 Fundamentos de Git - Obteniendo un repositorio Git](#)

Se asume que ya se tiene instalado Git o se ha tenido la experiencia en el proceso a través de lecturas previas.

Existen dos formas para empezar a trabajar bajo la filosofía de Git; una de ella es usar un directorio que tenga información, agregándole Git; la segunda es clonar un repositorio que puede estar en algún servidor público o privado.

La forma para trabajar con un repositorio en directorio que ya existe, se la detalla a continuación:

- Se usa el comando **git init**, dicho comando crea un subdirectorio importante y base llamado **.git**, el mismo contiene la estructura que necesita Git.

Usar la siguiente secuencia en su computador desde un terminal (sistema operativo Linux / Mac) o usando **git-bash** (Windows), desde el directorio del que se quiera trabajar.

```
mkdir demo  
cd demo  
git init  
ls -la
```

Lo anterior generará la estructura para trabajar con Git en un directorio.

La opción siguiente es utilizar el comando que permita realizar la clonación de un repositorio que esté alojado en un servidor. En este punto es importante mencionar que existen algunas opciones para tener repositorios en la nube, así por ejemplo: [Github](#), [Bitbucket](#), [GitLab](#). El comando a utilizar será, **git clone**, seguido de la dirección donde está alojado el servidor.

Se sugiere realizar la siguiente secuencia de comandos en su computador, desde un terminal o git-bash.

- Opción 1: `git clone https://github.com/ProgramacioAvanzada-Tec-Utpl/ejemplo01.git` .
 - El anterior genera una carpeta con el nombre ejemplo01.
- Opción 2: `git clone https://github.com/ProgramacioAvanzada-Tec-Utpl/ejemplo01.git ejemploDemo`.
 - Lo anterior generará una carpeta con el nombre ejemploDemo.



A medida que se empieza a trabajar en un proyecto, se agregan archivos a los directorios. La forma para empezar a darle seguimientos a los nuevos directorios o archivos, bajo la filosofía de Git es a través del comando **git add**.

- Se puede usar

- `git add .` Para agregar en el proceso de seguimiento a todos los nuevos archivos que son partes el directorio.
- `git add archivo.txt`. Permite solo dar seguimiento al archivo específico que consta luego del comando `git add`.

En cualquiera de los dos casos anteriores, se debe realizar la confirmación de los cambios, a través del comando **git commit**.

- La forma del comando es la siguiente:
- `git commit -m 'Cadena informativa de la confirmación'`

Con lo mencionado se puede tener un directorio con la filosofía de Git, y con archivos en seguimiento.



Semana 4

2.2. Control de versiones

Estimado estudiante, para el estudio y comprensión de las temáticas de la semana número dos, se hará uso del recurso educativo abierto denominado [Pro Git](#). El recurso posee amplia información que permitirá una comprensión de lo propuesto en la presente sección de contenido. Es importante indicar que se debe tener claros los conceptos de la semana uno y así continuar las temáticas siguientes.

2.2.1. Uso de branch en repositorios de versiones



Se solicita revisar de forma detallada y analítica la información del recurso ubicada en los puntos [3.1 Ramificaciones en Git - ¿Qué es una rama?](#) y [3.2 Ramificaciones en Git - Procedimientos Básicos para Ramificar y Fusionar](#).

En algunos recursos que se puede encontrar para el estudio, a un **branch** se lo conoce como **rama**. Importante, tomar en consideración, en términos generales usar ramas en un repositorio permite tomar como base la rama

por defecto que tiene un repositorio en Git y comenzar a trabajar de forma paralela e independiente.

Por defecto los proyectos bajo Git poseen una rama por llamada **main** o **master**. Se puede ejecutar la siguiente secuencia de comandos para verificar el nombre del branch por defecto que tiene un repositorio.

- `git clone https://github.com/ProgramacioAvanzada-Tec-Utpl/ ejemplo01.git .`
- `cd ejemplo01.`
- `git branch.`

Lo anterior genera como salida la lista de ramas/branchs que existen en el repositorio, en este punto solo existe uno, por tal razón se tiene como salida:

- `*main/master`; el * solo demuestra la rama en la que se está trabajando.

Para crear una nueva rama en un repositorio se ejecutará el siguiente comando: **git branch nuevarama**.

Al ejecutar en consola el comando **git branch** solo, deberá mostrar las dos ramas, aunque la rama que se sigue utilizando es la **main/master**.

Ejemplo 1: ahora se realiza un ejemplo donde se demuestra el proceso de creación de una rama y como varía la información dependiendo de la rama en la que se está trabajando.

Pasos a seguir en su computadora.

1. Crear un directorio, llamado demo.
2. Dentro del directorio, ejecutar **git init**.
3. Crear un archivo con el nombre index.html.
4. Ingresar en el archivo la siguiente información.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Demo</title>
    link
  </head>
  <body>
    body
  </body>
</html>
```

Agregar la información y guardar los cambios en el archivo.

5. Agregar el archivo para que sea manejado por Git, usar, **git add index.html**.
6. Confirmar los cambios del archivo, usar el comando, **git commit -m"ejemplo subido en rama master"**.
7. Listar los branch existentes, usar **git branch**.

La salida debe ser:

***master**

8. Agregar o crear un branch llamado nuevarama, usar el comando, **git branch nuevarama**.

Listar los branch existentes, usar **git branch**.

La salida debe ser:

***master**

nuevarama

9. Cambiar de rama/branch a la creada recientemente, usar el comando, **git checkout nuevarama**.
10. Listar los branch usar el comando, **git branch**.

La salida debe ser:

master

* nuevarama

11. Verificar la información del archivo index.html, debe ser igual al archivo de rama anterior.
12. Modificar la información del archivo index.html con la siguiente información.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Demo</title>
    link
  </head>
  <body>
    <ul>
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
    </ul>
  </body>
</html>
```

13. Agregar y confirmar los cambios del archivo en la nueva rama, usar el comando, **git commit -m"subiendo cambios en rama nueva rama"**.
14. Regresar a la rama anterior a través del comando: **git checkout master**.
15. Verificar el contenido del archivo index.html de la rama master. La información del archivo tendrá el mismo estado, que cuando fue creada la rama denominada **nuevarama**.

En la figura 3, se puede revisar el contenido de cómo deben quedar los archivos en cada rama.

Figura 3.

Uso de branch / ramas

rama: master

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <meta name="viewport" content="width=device-width" />
6     <title>Demo</title>
7     <link href="https://www.w3schools.com/html/html.css" rel="stylesheet">
8   </head>
9   <body>
10    <h1>Hello World</h1>
11    <ul>
12      <li>1</li>
13      <li>2</li>
14      <li>3</li>
15      <li>4</li>
16    </ul>
17  </body>
18</html>
```

rama: nuevarama

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <meta name="viewport" content="width=device-width" />
6     <title>Demo</title>
7     <link href="https://www.w3schools.com/html/html.css" rel="stylesheet">
8   </head>
9   <body>
10    <h1>Hello World</h1>
11    <ul>
12      <li>1</li>
13      <li>2</li>
14      <li>3</li>
15      <li>4</li>
16    </ul>
17  </body>
18</html>
```

Nota. Elizalde R, 2023

Algunos comandos que pueden ser importantes al momento de trabajar con las ramas.

- **git merge**
 - Ejemplo: Si está activo el trabajo en la rama **master** y se ejecuta el comando, **git merge nuevarama**. En ese momento Git realiza el proceso de unir o fusionar los archivos de la rama **nuevarama** con los archivos de la rama **master**.
- **git branch -d**
 - Cuando ya no se necesite trabajar en una rama, se puede usar el siguiente comando: **git branch -d nuevarama**.
 - Lo anterior elimina la rama con el nombre que se le pase como indicación.

2.2.2. Uso de GitHub



Para el mejor entendimiento de este apartado se solicita una revisión y lectura del siguiente recurso educativo denominado [¡Se puede entender cómo funcionan Git y GitHub!](#). En el recurso hace importantes precisiones en relación con los conceptos de Git y Github.

Actualmente, existen algunas plataformas que permiten manejar proyectos de repositorios de versiones, lideran las opciones: Github, Bitbucket y GitLab.

En la presente guía se va a comentar el uso de Github. Dicha plataforma permite almacenar proyectos para que sean mantenidos de forma colaborativa. Además, por cada repositorio que se crea en Github se puede agregar archivos, colaboradores, utilizar una *wiki* informativa, el [uso de pull-request de forma gráfica](#), entre otras características. Existen algunos planes que se pueden contratar para trabajar con Github, pero el plan sin costo, permite trabajar sin problemas.

Para hacer uso de Github, se debe crear un usuario para acceder y trabajar. Para ello, ingresar al proceso de [creación de usuario de Github](#), se recomienda usar este proceso si no se tiene un usuario en la plataforma. Luego de registrarse debe [ingresar a Github](#).

Los pasos para generar un repositorio en GitHub se detalla en el siguiente módulo didáctico.

[Creación de un proyecto en GitHub](#)

Luego de la creación del repositorio, se debe usar el comando *git clone* en la máquina donde se requiere trabajar. El proceso es el siguiente, desde una consola/cmd:

Crear un repositorio con su cuenta de Github, con el nombre ejemplo-002.

1. Ejecutar el comando *git clone* con la información de Github.

- Usar: *git clone https://github.com/[usuario]/ejemplo-002.git*.

2. Ingresar a la carpeta desde el terminal
 - Usar: cd ejemplo002.
3. Agregar los archivos a la dinámica de Github
 - git add .
4. Confirmar los cambios
 - git commit -m "cambios en archivo del repositorio ejemplo"
5. Finalmente, subir los cambios al repositorio en Github, se hace uso del comando git push. Dicho comando permite subir el conjunto de commits confirmados.
 - git push.

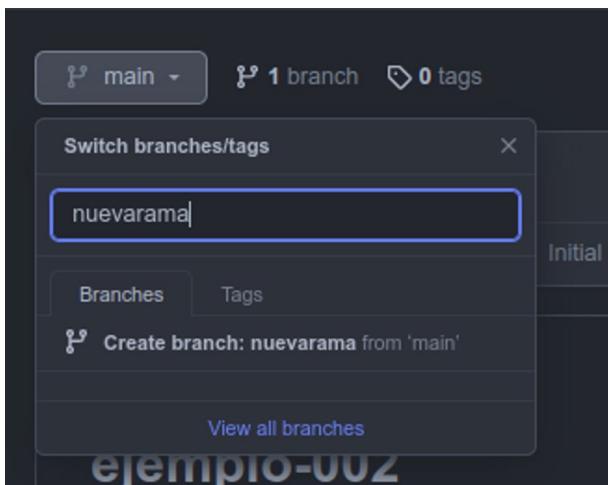


Importante mencionar, para poder ejecutar el comando git push, se necesita tener un [token de acceso personal](#) de Github. Dicho paso solo se lo debe realizar en una sola oportunidad.

Otra de las ventajas de usar Github, es la creación de branch de forma gráfica, desde la interfaz propia de la plataforma. En la figura 4, se indica la parte de la interfaz donde se puede agregar una rama/branch al repositorio en Github. Se digita el **nombre de la rama** y se pulsa el enlace **Create branch**.

Figura 4.

Crear branch desde Github



Nota. Elizalde R, 2023

Le invito a profundizar sus conocimientos acerca del uso y creación de GitHub Actions.

2.2.3. Uso y creación de GitHub Actions



La temática presente se desarrolla con base a la información del [Github Actions](#) de Github. Se solicita una revisión detallada. La información presentada resalta la importancia que puede prestar este componente de Github en procesos de integración y despliegue continuo ([IC/DC](#)).

Lo que busca Github Actions es crear flujos de trabajo que reaccionen ante cambios que se generen el repositorio. Como ventaja Github provee, en la capa gratuita, máquinas virtuales en algunos sistemas operativos: Linux, Windows, MacOS para ejecutar los flujos de trabajo; existe la posibilidad que las acciones trabajen en servidores propios.

En la documentación se detalla los siguientes componentes:

- **Workflows**, es un proceso automatizado definido en un archivo [YAML](#), que guiará la ejecución de un conjunto de trabajos (*jobs*).

- **Eventos**, se describen con las acciones que se realizan en los repositorios para que desencadenen el trabajo de los actions.
- **Trabajos**, conjunto de pasos que se ejecutarán en el flujo de trabajo. Los pasos pueden ser dependientes o independientes.
- **Acciones**, propio de la plataforma Github Actions. Genera proceso de complejidad alta, logrando evitar que el usuario genere código repetitivo.
- **Ejecutores**, la plataforma Github Actions proporciona ejecutores para diversos sistemas operativos.

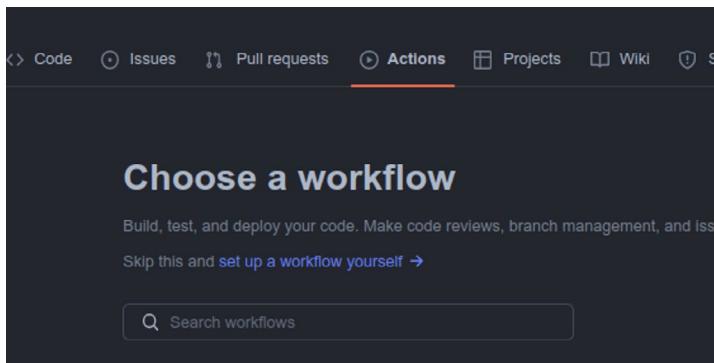
A continuación, se detalla un ejemplo usando la plataforma Github Actions. El [ejemplo está desarrollado en Github](#), se lo puede revisar. Pero como sugerencia, es importante que usted lo ejecute desde cero.

Paso a seguir:

1. Crear un repositorio en Github.
2. Clonar el repositorio en su máquina local.
3. Crear un archivo con el nombre ejemplo.py al repositorio. Se sugiere agregar en el archivo la información [detallada en el enlace](#).
4. Agregar y confirmar el archivo bajo la dinámica de Github.
5. Subir los cambios al repositorio de Github.
6. Desde la plataforma agregar un **Actions**. Dar click en la opción **set up a workflow yourself**. Ver figura 5.

Figura 5.

Crear un actions en Github

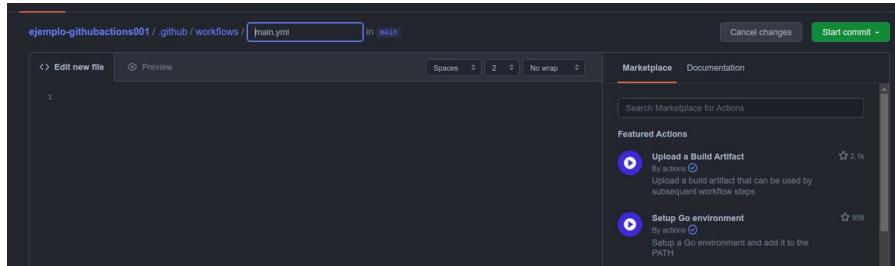


Nota. Elizalde R, 2023

7. Agregar información al archivo main.yml bajo las consideraciones de Github Actions.

Figura 6.

Agregar información al Github Actions

A screenshot of the GitHub Actions editor interface. At the top, it shows the repository path "ejemplo-githubactions001/.github/workflows/main.yml". Below this is a code editor window with the file content: "1". To the right of the code editor is a sidebar titled "Marketplace" which lists "Featured Actions". The first action is "Upload a Build Artifact" by actions, with a star rating of 2.1k. The second action is "Setup Go environment" by actions, with a star rating of 958.

Nota. Elizalde R, 2023

El código que se debe agregar es:

```
name: learn-github-actions
run-name: ${{ github.actor }} is learning GitHub Actions
on: [push]
jobs:
  prueba-python:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-python@v4
        with:
          python-version: '3.10'
      - run: python ejemplo.py
```

Existen algunas consideraciones:

- **Name**, es el nombre que se le dará al actions.
- **Run-name**, nombre que se mostrará en la lista de ejecuciones.
- **On**, define el evento que ejecutará la acción. Para el ejemplo, la acción se genera cuando el usuario realiza un *push*.
- **Jobs**, lista de trabajos que se van a realizar. La lista no tiene límite, a cada job, se le debe asignar un nombre. Para el ejemplo se llama *prueba-python*.

- Run-on, bajo el job con nombre asignado prueba-python se agrega run-on, donde se especifica el entorno donde se debe ejecutar la acción; en el ejemplo se usa ubuntu-latest.
 - Steps, los pasos que se deben seguir para que tenga éxito lo solicitado.
 - Uses, permite agregar **acciones predeterminadas por la plataforma**. En el ejemplo se está especificando con actions/checkout@v3, que el action pueda tomar el repositorio propio y clonarlo; actions/setup-python@v4, instala una versión de Python y lo agrega al PATH de la máquina virtual.
 - Con *with*, se agrega características particulares. En el ejemplo se define la versión de Python.
 - Luego, con run se ejecuta en consola comandos del entorno, de acuerdo a lo especificado. En el ejemplo de ejecuta el archivo de Python, detallado.
 - Finalmente, pulsar el botón, start commit.
8. Se regresa a la pestaña Actions, ver figura 7, ahora se observará el listado de la o las acciones creadas. En la parte central de la pantalla, se lista los trabajos ejecutados. Por cada trabajo se puede revisar los pasos a seguir por los actions.

Figura 7.

Vista de la ejecución de un actions

The screenshot shows the GitHub Actions interface. On the left, there's a sidebar with options like 'Summary', 'Jobs', 'Run details', 'Usage', and 'Workflow file'. The 'Jobs' section is expanded, showing a single job named 'prueba-python' which has succeeded 44 minutes ago. The job details show a series of steps: 'Set up job', 'Run actions/checkout@v3', 'Run actions/setup-python@v4', and 'Run python ejemplo.py'. The final step is expanded, showing the output of the Python script 'ejemplo.py' which prints 'Hola mundo' 10 times from 0 to 9. The output is as follows:

```
1 ► Run python ejemplo.py
11 0 - Hola mundo
12 1 - Hola mundo
13 2 - Hola mundo
14 3 - Hola mundo
15 4 - Hola mundo
16 5 - Hola mundo
17 6 - Hola mundo
18 7 - Hola mundo
19 8 - Hola mundo
20 9 - Hola mundo
21 10 - Hola mundo
```

Nota. Elizalde R, 2023

2.2.4. Uso de herramientas gráficas para control de versiones

Hasta ahora el trabajo realizado en Git ha sido mediante comandos de consola/terminal/cmd. Pero es importante mencionar que existen algunas alternativas gráficas.

En el recurso educativo existe una sección llamada [GUI Clients](#), en ella se explica una lista de clientes de acuerdo al sistema operativo que se necesite. Revisar la tabla 8.

Tabla 8.

Listado de clientes gráficos para Git.

Nombre del cliente	Windows	Linux	MacOs
GitHub Desktop.	X		X
SourceTree.	X		X
TortoiseGit.	X		
Git Extensions.	X	X	X
GitKraken.	X	X	X
Magit.	X	X	X
SmartGit.	X	X	X
LazyGit.	X	X	X

Nota. Elizalde R, 2023



Actividades de aprendizaje recomendadas

- **Actividad 1:** realizar una lectura comprensiva del recurso denominado [Introducción a Git y Github – Día 1](#). La información en el recurso está dividida en seis capítulos cortos, se abordan temas como: conceptos de los sistemas de control de versiones, aspectos básicos de git, uso básico y avanzado de git, manejo de ramas en git y explicación del flujo de trabajo con git. La sugerencia principal es la de efectuar una lectura completa del contenido propuesto, con el objetivo de relacionar con lo expuesto en la guía didáctica y reforzar las temáticas. Luego, es importante recrear los ejemplos y configuraciones en su entorno local de los capítulos que hacen referencia a la instalación, configuración, creación de proyectos y administrar ramas o branchs.

Actividad 2: resolver la autoevaluación 2; en la actividad se presentan un conjunto de preguntas basadas en las temáticas descritas en la unidad 2: desarrollo de aplicaciones con base de datos; sección 2.1 Control de versiones. En la parte introductoria de la actividad se listan los contenidos de la guía didáctica, que se debe revisar para contestar las interrogantes. Recuerde que la autoevaluación le permitirá identificar el nivel de aprendizaje y comprensión alcanzado; así mismo, determinar los ítems que se debe volver a revisar para lograr los resultados de aprendizaje propuestos. Adicionalmente, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado. Finalmente, después de realizar la autoevaluación, puede comprobar las respuestas correctas de las preguntas en la sección solucionario de la guía didáctica.



Autoevaluación 2

Estimado estudiante, es momento de realizar la autoevaluación 2 que permitirá identificar el nivel de entendimiento de algunos conceptos estudiados en la unidad 2: desarrollo de aplicaciones con base de datos punto 2.1 Control de versiones. Antes de realizar la autoevaluación, se requiere las siguientes acciones:

- Revisar todo el contenido de las temáticas indicadas.
- Leer los recursos educativos recomendados.
- Hacer organizadores gráficos para ordenar la información para su estudio.

La autoevaluación le permitirá reafirmar los conceptos estudiados. Los enunciados tienen una sola opción correcta, lea de forma atenta cada respuesta y seleccione la opción que usted considere correcta.

1. Identifique el concepto más apropiado para un control de versiones.

- a. Un control de versiones, se convierte en un mecanismo o proceso que permite registrar en el tiempo el conjunto de cambios que se ha efectuado para uno o varios archivos.
- b. Un control de versiones, se convierte en un mecanismo o proceso que permite registrar en el tiempo el conjunto de cambios que se ha realizado para uno o varios archivos de tipo txt.
- c. Un control de versiones, se convierte en un mecanismo o proceso que permite registrar, de manera particular al líder del proyecto, en el tiempo el conjunto de cambios que se han hecho para uno o varios archivos.

- 2. Identifique los tipos principales de sistemas de control de versiones.**
- a. Sistema de control de versiones en la nube; sistema de control de versiones centralizado; sistema de control de versiones distribuidas.
 - b. Sistema de control de versiones locales; sistema de control de versiones centralizado; sistema de control de versiones de software libre.
 - c. Sistema de control de versiones locales; sistema de control de versiones centralizado; sistema de control de versiones distribuidas.
- 3. ¿Cuáles son los máximos exponentes en los sistemas de control de versiones distribuidos?**
- a. Git; Mercurial; SVN.
 - b. Git; Mercurial; Bazaar .
 - c. Git; Gitlab; Bazaar.
- 4. Hablando de Git. ¿Quién es el creador y en qué año lo hizo?**
- a. Linus Torvalds / 2007.
 - b. Linus Torvalds / 2005.
 - c. Richard Stallman / 2005.
- 5. Identifique los comandos que permiten asignar información relacionada con el nombre de usuario y correo al momento de realizar un commit en Git.**
- git config --global user.name "Nombre de Usuario".
git config --global user.email nombre@ejemplo.com.
- git config global user.name "Nombre de Usuario".
git config global user.email nombre@ejemplo.com.
- git -global user.name "Nombre de Usuario".
git -global user.email nombre@ejemplo.com.

6. Seleccione la secuencia de comandos en Git, que permite crear la estructura correcta para trabajar.

mkdir demo.

cd demo.

git clone.

mkdir demo.

cd demo.

git init.

mkdir demo.

cd demo.

git ls.

7. ¿Cuál es el comando que se usa para empezar a darle seguimientos a los nuevos directorio o archivos, bajo la filosofía de Git?

a. git commit.

b. git add.

c. git init.

8. De las siguientes afirmaciones, ¿Cuál es la correcta para la definición de una rama o branch?

a. Un branch es un repositorio que permite tomar como base la rama por defecto que tienen un repositorio en Git y comenzar a trabajar de forma paralela e independiente.

b. Un branch es un repositorio que permite tomar como base la rama por defecto que tienen un repositorio en Git y comenzar a trabajar de forma paralela pero dependiente.

c. Un branch es un repositorio que permite tomar como base la rama la última rama que tiene un repositorio en Git y comenzar a trabajar de forma paralela e independiente.

9. ¿Cuál es el comando que permite crear una rama en un repositorio?

a. git branch nuevarama.

b. git "nuevarama" branch.

c. git branch add nuevarama.

10. ¿Cuál es el comando que permite acceder a una rama, previamente creada?

- a. git checkout pass mirama.
- b. git checkout branch mirama.
- c. git checkout mirama.

[Ir a solucionario](#)



2.3. Programación con bases de datos

Es momento de avanzar hacia el estudio de los conceptos y ejemplificaciones para entender cómo usar un lenguaje de programación de alto nivel, y poder generar información que se guarde y luego esté disponible en algún gestor de base de datos relacional o no relacional. Para el proceso de las temáticas relacionadas con lo comentado, se solicita a usted estimado estudiante que se realice una lectura comprensiva de las siguientes temáticas del recurso denominado [Gestión de Información Web en Python](#), que consta en la biblioteca virtual de la universidad.

2.3.1. Acceso base de datos relacionales desde lenguajes de programación

Para entender los conceptos y métodos a usar para manejar información desde un lenguaje de programación hacia un gestor de base de datos, es indispensable realizar una lectura del capítulo cuatro denominado [Acceso a bases de datos relacionales utilizando Python: MySQL y SQLite](#), del recurso educativo mencionado anteriormente.

Se utilizará el lenguaje de programación Python en conjunto con los conceptos de programación orientada a objetos. Se puede emplear la [documentación de Python](#) y la [página oficial de descargas del lenguaje](#), como proceso introductorio. Es fundamental que se tenga instalado Python en los computadores personales.

Con relación a la base de datos que se va a utilizar será SQLite, pero lo destacable es que los ejemplos pueden ser fácilmente replicables en otras bases de datos como: [MySQL](#), [MariaDB](#), [PostgreSQL](#), [Oracle](#). En relación con SQLite, se solicita que se haga una instalación simple, con base a la [página oficial](#), además se usará una interfaz de acceso a la información de las bases de datos SQLite, llamado [SQLite Browser](#).

Se usará una de las potencialidades de Python a través de la especificación denominada [Python Database API o DB-API](#).

A continuación, se detallará un ejemplo que permitirá: crear y agregar información, realizar consultas y eliminar información de la base de datos.

Revisar la [información del ejemplo subido en Github](#). Se recomienda seguir los pasos en su entorno local en una primera instancia.

1. Creación de la base de datos.

- Se importa la librería sqlite3 y se crea un objeto de tipo Connection, al cual se le envía como cadena el nombre de la base de datos. En este caso indicar que si aún no existe la base creada, al ejecutar el script, se crea inicialmente.

2. Creación de entidades

- Se utiliza el objeto Connection del script anterior, luego se accede al método cursor para poder realizar las acciones en la base de datos. Se usa la sentencia: cursor = conn.cursor()

Cabe indicar que a través de cursor se puede ejecutar comandos SQL mediante el método execute.

- Luego, para el ejemplo se crea una tabla denominada autor con las siguientes características: nombre, apellido, cédula, edad.

Se usa una cadena en Python y luego se ejecuta la sentencia.

```
cadena_sql = 'CREATE TABLE Autor (nombre TEXT, apellido TEXT, cedula  
TEXT, edad INTEGER)'  
cursor.execute(cadena_sql)
```

Luego de ejecutar la sentencia anterior, a través de SQLite Browser se puede verificar la creación de la entidad en la base de datos, ver figura 8.

Figura 8.

Base datos creada en SQLite, vista del SQLite Browser

Name	Type	Schema
Autor		CREATE TABLE Autor (nombre TEXT, apellido TEXT, cedula TEXT, edad INTEGER) 'nombre' TEXT 'apellido' TEXT 'cedula' TEXT 'edad' INTEGER

Nota. Elizalde R, 2023

Finalmente, se cierra el enlace a la base datos:

```
cursor.close()
```

3. Ingresar información en las entidades de la base datos.

- Se usa el objeto connection del script de la base de datos.
- Se usa la sentencia: cursor = conn.cursor() para trabajar en las acciones.
- Se crea una cadena que almacene la sentencia de ingreso de información, se recuerda los atributos: nombre, apellido, cédula, edad. Se utiliza las siguientes sentencias en Python, donde se crean variables de ayuda.

```
nombre = "Andrés Vinicio 2"  
apellido = "Jara Vinces"  
cedula = "1011019091"  
edad = 30  
cadena_sql = """INSERT INTO Autor (nombre, apellido, cedula, edad) \  
VALUES ('%s', '%s', '%s', %d);""" % (nombre, apellido, cedula, edad)  
cursor.execute(cadena_sql)
```

Posterior a la ejecución se debe confirmar los cambios a través de la sentencia que hace uso del método commit del objeto de tipo Connection.

```
conn.commit()
```

Finalmente, se puede comprobar que la información ha sido guardada en la base de datos. Utilizar la SQLite Browser, ver figura 9.

Figura 9.

Verificar ingreso de información en la base de datos

	nombre	apellido	cedula	edad
Filter	Filter	Filter	Filter	Filter
1	Andrés Vinicio	Jara Vinces	1011019091	30

Nota. Elizalde R, 2023

4. Consulta de información en la entidad de la base de datos.

- Se usa el objeto Connection del script de la base de datos.
- Se usa la sentencia: cursor = conn.cursor() para trabajar en las acciones.
- Para el ejemplo, se crean algunos registros adicionales, de acuerdo a lo explicando anteriormente.
- Se hace una consulta a la base de datos a través de código SQL, con el método execute.

```
cadena_consulta_sql = "SELECT * from Autor"  
cursor.execute(cadena_consulta_sql)
```

- Los registros obtenidos de la base de datos mediante el método fetchall; este proceso, devuelve una lista de información; en cada posición del arreglo existe una tupla de información, con los datos de cada registro.
- Finalmente, se realiza un proceso básico en Python para poder iterar la lista y presenta los datos, ver figura 10.

Figura 10.

Proceso iterativo de una lista de registros de una entidad de la base de datos

Proceso iterativo

```
48 # se realiza un ciclo repetitivo para recorrer la secuencia de información
49 # resultante
50 for d in informacion:
51     print("%s - %s - %s - %d" % (d[0], d[1], d[2], d[3]))
52
53 # cerrar el enlace a la base de datos (recomendado)
54 cursor.close()
```

salida

```
Andrés Víncio - Jara Víncos - 1011019091 - 30
Ana Salas - Jara Víncos - 1011019091 - 30
Marco Barcia - Jara Víncos - 1011019091 - 30
```

Nota. Elizalde R, 2023

5. Actualización de información de las entidades de la base de datos.

- Opcional, se puede borrar la base datos; para que la información no se repita.
- Se importa el objeto connection y se crea el cursor para las acciones.
- Se genera la cadena de actualización.

```
cadena = """UPDATE Autor SET apellido='%s' WHERE nombre='%s'"""
("Dávalos Lima", "Ana Salas")
cursor.execute(cadena)
conn.commit()
```

Se cambia el apellido con nuevo valor “Dávalos Lima” a todos los registros que tengan como valor la cadena “Ana Salas”.

- Luego de la ejecución, se puede verificar la información con SQLite Browser, ver figura 11.

Figura 11.

Actualización de la entidad de la base de datos

The screenshot shows the SQLite Browser interface with the 'Autor' table selected. The table has four columns: 'nombre', 'apellido', 'cedula', and 'edad'. The data is as follows:

	nombre	apellido	cedula	edad
1	Andrés Vinicio	Jara Vinces	1011019091	30
2	Ana Salas	Dávalos Lima	1011019091	30
3	Marco Barcia	Jara Vinces	1011019091	30

Nota. Elizalde R, 2023

6. Eliminación de información de las entidades de la base de datos.

- Opcional, se puede borrar la base datos; para que la información no se repita.
- Se importa el objeto connection y se genera el cursor para las acciones.
- Se genera la cadena y se ejecuta el proceso de eliminación.

```
cadena = """DELETE from Autor WHERE nombre='%s'"""\ncursor.execute(cadena)\nconn.commit()
```

Se elimina los registros la entidad de la base de datos que tenga el nombre de Marco Barcia.

Luego de la ejecución, se puede verificar la información con SQLite Browser, ver figura 12.

Figura 12.

Vista de información antes y después del proceso de eliminación

Vista de la base de datos antes de eliminar información

nombre	apellido	cedula	edad
Filter	Filter	Filter	Filter
1 Andrés Vinicio	Jara Vinces	1011019091	30
2 Ana Salas	Jara Vinces	1011019091	30
3 Andrés Vinicio 2	Jara Vinces	1011019091	30

Vista de la base de datos después de eliminar información

nombre	apellido	cedula	edad
Filter	Filter	Filter	Filter
1 Andrés Vinicio	Jara Vinces	1011019091	30
2 Ana Salas	Jara Vinces	1011019091	30

Nota. Elizalde R, 2023



Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.

Le invito a profundizar sus conocimientos acerca del uso de Object-relational mapping (ORM).

2.3.2. Uso de Object-relational mapping (ORM)

De acuerdo a Farias (2016), el uso de ORM en el desarrollo de aplicaciones permite mejorar el proceso de la persistencia de información entre los lenguajes de programación de alto nivel y los gestores de base de datos

Los ORM tienen algunas ventajas que las describimos a continuación:

- El empleo de los conceptos de programación orientada a objetos para ingresar y acceder a la información de una base de datos.
- Reutiliza código.

- Tratar la información con estructuras de datos que contienen objetos.
- Poder pasar de una base de datos relacional a otra, cambiando la cadena de conexión.
- Acelerar los procesos de desarrollo en los equipos de software.
- Usar los mecanismos propios de seguridad que incluyen los ORM para el tratamiento de la información.

Por otro lado, la desventaja que se le atribuye al uso de ORM es la posibilidad que las consultas generadas en las aplicaciones, no estén afinadas, causando que los procesos tarden, fuera del rango normal.

En la actualidad existen gran cantidad de librerías ORM, que varían según el lenguaje de programación. En Elizalde (2020), se muestra un listado de las principales librerías, ver tabla 9.

Tabla 9.

Lista de las principales librerías ORM según el lenguaje de programación.

Nombre de la librería	Lenguaje de programación	Enlace web
Doctrine.	PHP.	Documentación oficial de Doctrine.
Eloquent.	PHP.	Documentación oficial de Eloquent.
Zend-db.	PHP.	Documentación oficial de Zend-db.
Hibernate.	Java.	Documentación oficial de Hibernate.
Ebean.	Java/Kotlin.	Documentación oficial de Ebean.
Sequel.	Ruby.	Documentación oficial de Ruby.
SQLAlchemy.	Python.	Documentación oficial de SQLAlchemy.
Django-ORM.	Python.	Documentación oficial de Django-ORM.
Pony-ORM.	Python.	Documentación oficial de Pony-ORM.

Nota. Elizalde R, 2023

Para ejemplificar el uso básico de un ORM con un lenguaje de programación, se usa la librería SQLAlchemy de Python. Para realizar la exemplificación se requiere realizar la instalación de la librería, se sugiere ejecutar el siguiente comando:

- `pip install SqlAlchemy.`

Además, se recomienda revisar la [documentación de SQLAlchemy](#).

El motor de base de datos que se utiliza es SQLite para guardar la información. Se solicita, realizar la instalación de la [página oficial](#), además se utilizará una interfaz de acceso a la información de bases de datos SQLite, llamado [SQLite Browser](#).

1. Creación de una base de datos a través de SQLAlchemy en SQLite.

- Se crea el enlace a la base de datos, a través del método `create_engine` propio de SQLAlchemy. Este *script* será usado en los procesos de creación de las entidades y consulta de información.

```
from sqlalchemy import create_engine
# se genera en enlace al gestor de base de
# datos para el ejemplo se usa la base de datos
# sqlite
engine = create_engine('sqlite:///ejemplo001.db')
```

- Se crea el *script* para iniciar las entidades, aquí se resalta el hecho de usar características propias de SqlAlchemy.

```
# se crea la clase llamada Base que permite definir las clases bajo las
# características de SQLAlchemy
Base = declarative_base()

# Se crea la una entidad llamada Autor, que hereda
# de Base
class Autor(Base):
    __tablename__ = 'autor' # El nombre de la entidad en sqlite
    # Se definen los atributos
    id = Column(Integer, primary_key=True) # este atributo es entero y
                                            # se lo considera como llave
                                            # primaria
    nombre = Column(String) # atributo de tipo String
    apellido = Column(String)
    nacionalidad = Column(String)
    edad = Column(Integer)
    # Sentencia que permite crear o migrar las clases en Python al
    # gestor de base de datos, expresado en el engine.
    Base.metadata.create_all(engine)
```

Se puede verificar que existe la base de datos y que contiene una entidad/tabla llamada *autor*, las propiedades de la tabla son las definidas en la clase *autor*, ver figura13.

Figura 13.

Base de datos generada a través de SQLAlchemy, en SQLite

The screenshot shows the SQLite Browser interface. At the top, there are tabs: Database Structure, Browse Data, Edit Pragmas, and Execute SQL. Below these are buttons for Create Table, Create Index, Modify Table, and Delete Table. A sidebar on the left lists 'Tables (1)' containing 'autor'. Under 'autor', there are five columns: 'id' (INTEGER), 'nombre' (VARCHAR), 'apellido' (VARCHAR), 'nacionalidad' (VARCHAR), and 'edad' (INTEGER). To the right of the table structure is the corresponding SQL CREATE TABLE statement.

Name	Type	Schema
Tables (1)		
autor		CREATE TABLE autor (id INTEGER NOT NULL, nombre VARCHAR, apellido VARCHAR, `id` INTEGER NOT NULL, `nombre` VARCHAR, `apellido` VARCHAR, `nacionalidad` VARCHAR, `edad` INTEGER)
id	INTEGER	
nombre	VARCHAR	`nombre` VARCHAR
apellido	VARCHAR	`apellido` VARCHAR
nacionalidad	VARCHAR	`nacionalidad` VARCHAR
edad	INTEGER	`edad` INTEGER
Indices (0)		
Views (0)		
Triggers (0)		

Nota. Elizalde R, 2023

2. Agregar información a través de SQLAlchemy.

- Se crea un script que permite agregar registros a la base de datos, se realiza inserciones a la base de datos a través de lenguaje Python.

```
# se crea un objetos de tipo Autor
autor1 = Autor(nombre="José", apellido="Armijos", nacionalidad="ecuatoriana",
               edad=40)
autor2 = Autor(nombre="Sara", apellido="Benítez", nacionalidad="colombiana",
               edad=20)
autor3 = Autor(nombre="Pedro", apellido="Díaz", nacionalidad="peruana",
               edad=35)
autor4 = Autor(nombre="Mónica", apellido="Carrión",
               nacionalidad="ecuatoriana",
               edad=25)

# se agrega los objetos de tipo Autor a la sesión
# a la espera de un commit
session.add(autor1)
session.add(autor2)
session.add(autor3)
session.add(autor4)

# se necesita confirmar los cambios que existan en la sesión
# se usar commit
session.commit()
```

A través de SQLite Browser se puede verificar, ver figura 14, que la información ya está en la base de datos.

Figura 14.

Verificar que la información está almacenada en SQLite

The screenshot shows a database interface with a toolbar at the top containing 'Database Structure', 'Browse Data', 'Edit Pragmas', and 'Execute SQL'. Below the toolbar, a dropdown menu says 'Table: autor'. The main area displays a table with the following data:

	id	nombre	apellido	nacionalidad	edad
1	1	José	Armijos	ecuatoriana	40
2	2	Sara	Benitez	colombiana	20
3	3	Pedro	Díaz	peruana	35
4	4	Mónica	Carrión	ecuatoriana	25

Nota. Elizalde R, 2023

3. Consultar información a través de SQLAlchemy.

- Luego de revisar los procesos de creación de entidades e ingreso de información, es momento de revisar algunas consultas generales que se pueden realizar a través de SQLAlchemy.
- Consultar todos los registros de la entidad, ver la salida generada en la figura 15.

```
# Obtener todos los registros de la entidad Autor.  
# Se hace uso del método query.  
# all, significa que se obtiene todos los registros  
lista_autores = session.query(Autor).all()  
# La variable lista_autores, tendrá un listado de objetos de tipo  
Autor.  
  
# se realiza un proceso iterativo para presentar la información  
# de cada objeto.  
for l in lista_autores:  
    print(l)
```

Figura 15.

Salida de información al consultar todos los registros de la entidad a través de SQLAlchemy

```
José Armijos ecuatoriana 40
Sara Benítez colombiana 20
Pedro Díaz peruana 35
Mónica Carrión ecuatoriana 25
```

Nota. Elizalde R, 2023

- Consultar los registros que cumplan una condición. Ver figura 16.

```
# Obtener todos los registros de la entidad Autor con una(s)
# condición.
# Se hace uso del método query.
# filter, permite agregar condiciones a la búsqueda, con base
# a las propiedades de la entidad
lista_autores =
    session.query(Autor).filter(Autor.nacionalidad=="ecuatoriana")
# La variable lista_autores, tendrá un listado de objetos de tipo
Autor que
# tengan en la propiedad de nacionalidad el valor: ecuatoriana

# se realiza un proceso iterativo para presentar la información
# de cada objeto.
for l in lista_autores:
    print(l)
```

Figura 16.

Salida de información al consultar con una condición los registros de la entidad a través de SQLAlchemy

```
José Armijos ecuatoriana 40
Mónica Carrión ecuatoriana 25
```

Nota. Elizalde R, 2023

- Consultar los registros y ordenarlos en función de una propiedad. Ver figura 17.

```
# Obtener todos los registros de la entidad Autor.  
# Se hace uso del método query.  
# order_by, permite ordenar la búsqueda, con base  
# a las propiedades de la entidad  
lista_autores = session.query(Autor).order_by(Autor.edad)  
# La variable lista_autores, tendrá un listado de objetos de tipo  
Autor  
# ordenados por la propiedad edad de la entidad  
  
# se realiza un proceso iterativo para presentar la información  
# de cada objeto.  
for l in lista_autores:  
    print(l)
```

Figura 17.

Salida de información al consultar los registros de la entidad a través de SQLAlchemy, pero ordenados por una propiedad de la entidad

```
Sara Benítez colombiana 20  
Mónica Carrión ecuatoriana 25  
Pedro Díaz peruana 35  
José Armijos ecuatoriana 40
```

Nota. Elizalde R, 2023



Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.



2.4. Programación con bases de datos

2.4.1. Acceso base de datos no relacionales desde lenguajes de programación



Para entender los conceptos y métodos a usar para manejar información desde un lenguaje de programación hacia un gestor de base de datos, es indispensable realizar una lectura del capítulo quinto denominado [Acceso a bases de datos NoSQL utilizando Python: MongoDB con PyMongo](#) del recurso educativo mencionado anteriormente.

Con relación a la base de datos que se va a utilizar será MongoDB, que es una base de datos NoSQL que guarda la información en documentos, se debe tener conocimiento del uso de datos en formato [JSON](#) y la estructura de datos en Python llamada diccionarios.

Importante para este apartado que se tenga instalado en el computador personal el lenguaje de programación Python y la base de datos MongoDB. Se puede revisar el proceso de instalación de [MongoDB en Ubuntu](#); o, el [proceso en Windows](#).

La finalidad de esta temática es conocer los procesos para guardar, buscar y eliminar información en una base de datos NoSQL, desde un lenguaje de programación de alto nivel, como Python.

Para llevar el proceso adelante, se necesita preparar el entorno de trabajo. Adicional a la instalación de Python y MongoDB, es necesario hacer uso de una librería que permite desde Python conectarse a una base de datos en MongoDB. Para la instalación se sugiere realizar el siguiente proceso desde un terminal/CMD/consola del sistema operativo.

- `pip install pymongo`.

A continuación, se detalla el proceso para agregar, consulta y eliminar información usando Python y MongoDB.

1. Creación de una base de datos y una colección

- Este proceso se lo realizará desde una consola propia de MongoDB. El comando que se debe ejecutar es el siguiente:

mongosh.

Mongosh, permite acceder a una consola propia de MongoDB.

- Dentro de la consola se puede ejecutar comandos de MongoDB. Para crear la base de datos (no formalmente en MongoDB) se usará, dentro de mongosh:

use ejemploMongo001.

- Luego, se usa el comando `db.createCollection` para crear la colección, donde se ubicarán los documentos(registros) que se necesiten almacenar.

db.createCollection("autores").

Importante, lo comentado en este punto se lo debe efectuar en mongosh, ver figura 18.

Figura 18.

Proceso para crear una colección en MongoDB, desde mongosh

```
ejemploMongo001> use ejemploMongo001
already on db ejemploMongo001
ejemploMongo001> db.createCollection("autores")
{ ok: 1 }
ejemploMongo001> █
```

Nota. Elizalde R, 2023

2. Agregar información a una colección

- Para agregar información a la colección se lo realizará desde Python, a través de la librería Pymongo.

Se crea un script en Python, que inicializará cuando sea necesario un [enlace a la base de datos](#). Se usa las siguientes sentencias:

```
from pymongo import MongoClient  
client = MongoClient('mongodb://localhost:27010/')
```

La dirección de enlace usada para crear el objeto client, es con base al puerto (27010) donde se tiene instalado MongoDB. Es posible que el puerto cambie en cada computador. Importante, este *script* se lo usará en las demás acciones.

Luego se crea un [script que permite agregar información](#) a la colección previamente creada en mongosh.

```
# conjunto de datos a guardar en la colección  
# importante, aquí se usa la estructura de Python denominada  
# diccionario  
# El proceso que agrega un solo documento  
data_01 = {"nombre": "Luis", "apellido": "Valencia",  
"nacionalidad": "ecuatoriana", "numero_publicaciones": 100}  
colección.insert_one(data_01)
```

Se utiliza el método `insert_one`, que recibe como parámetro un diccionario.

Además, es posible emplear otro método para agregar una lista de diccionarios.

```
# proceso que agrega una lista de documentos  
lista = [  
{"nombre": "José", "apellido": "Medina",  
"nacionalidad": "ecuatoriana",  
"numero_publicaciones": 90},  
{"nombre": "María", "apellido": "Velez", "nacionalidad": "peruana",  
"numero_publicaciones": 80}  
]  
colección.insert_many(lista)
```

Luego de ejecutar el *script*, se puede regresar a mongosh y ejecutar el siguiente comando para verificar que se ha guardado la información: `db.autores.find()`. Ver figura 19

Figura 19.

Verificación de ingreso de información en mongosh

```
ejemploMongo001> db.autores.find()
[
  {
    _id: ObjectId("63ceffd72d2ca613e259824b"),
    nombre: 'Luis',
    apellido: 'Valencia',
    nacionalidad: 'ecuatoriana',
    numero_publicaciones: 100
  },
  {
    _id: ObjectId("63cf03cb8537d2ee9246a465"),
    nombre: 'José',
    apellido: 'Medina',
    nacionalidad: 'ecuatoriana',
    numero_publicaciones: 90
  },
  {
    _id: ObjectId("63cf03cb8537d2ee9246a466"),
    nombre: 'María',
    apellido: 'Velez',
    nacionalidad: 'peruana',
    numero_publicaciones: 80
  }
]
```

Nota. Elizalde R, 2023

3. Consulta información de una colección

- Se crea un *script* que genera las consultas a la colección desde Python.
- Proceso que muestra el último registro guardado.

```
# se usa método find_one, a partir de la colección
data_01 = colección.find_one()
print(data_01)
```

- Proceso que muestra todos los registros de la colección, ver figura 20.

```
# se usa método find, a partir de la colección
print("Muestra todos los documentos de la base de datos")
data_02 = colección.find()
for registro in data_02:
    print(registro)
```

Figura 20.

Verificación de la salida de información usando consultas en pymongo

```
Muestra un solo documento de la base de datos
{'_id': ObjectId('63ceffd72d2ca613e259824b'), 'nombre': 'Luis', 'apellido': 'Valencia', 'nacionalidad': 'ecuatoriana',
 '_numero_publicaciones': 100}
Muestra todos los documentos de la base de datos
[{'_id': ObjectId('63ceffd72d2ca613e259824b'), 'nombre': 'Luis', 'apellido': 'Valencia', 'nacionalidad': 'ecuatoriana',
 '_numero_publicaciones': 100},
 {'_id': ObjectId('63cf03cb8537d2ee9246a465'), 'nombre': 'José', 'apellido': 'Medina', 'nacionalidad': 'ecuatoriana',
 '_numero_publicaciones': 90},
 {'_id': ObjectId('63cf03cb8537d2ee9246a466'), 'nombre': 'María', 'apellido': 'Velez', 'nacionalidad': 'peruana', 'nu
mero_publicaciones': 80}]
```

Nota. Elizalde R, 2023

- Se puede realizar las consultas usando los **métodos find_one** y **find pero con parámetros**, con el objetivo de afinar las búsquedas de información.

```
# se usa método find_one con parámetros, a partir de la colección
print("Muestra un solo documento de la base de datos")
data_01 = colección.find_one({'nombre':'Luis'})
print(data_01)

# se usa método find con parámetros, a partir de la colección
print("Muestra todos los documentos de la base de datos que cumplan
con la \
condición")
data_02 = colección.find({'numero_publicaciones':{"$lt":100}})
for registro in data_02:
    print(registro)
```

Figura 21.

Verificación de la salida de información usando consultas en pymongo, usando parámetros

```
Muestra un solo documento de la base de datos
{'_id': ObjectId('63ceffd72d2ca613e259824b'), 'nombre': 'Luis', 'apellido': 'Valencia', 'nacionalidad': 'ecuatoriana',
 '_numero_publicaciones': 100}
Muestra todos los documentos de la base de datos que cumplan con la condición
[{'_id': ObjectId('63cf03cb8537d2ee9246a465'), 'nombre': 'José', 'apellido': 'Medina', 'nacionalidad': 'ecuatoriana',
 '_numero_publicaciones': 90},
 {'_id': ObjectId('63cf03cb8537d2ee9246a466'), 'nombre': 'María', 'apellido': 'Velez', 'nacionalidad': 'peruana', 'nu
mero_publicaciones': 80}]
```

Nota. Elizalde R, 2023

Indica que la consulta está buscando todos los registros que tiene en el valor de número_publicaciones menor a 100.

4. Eliminar información de una colección

- Se crea un *script* que permita eliminar información de las colecciones, desde Python.

- Se usa el método `delete_many`, a partir de la colección, indicar que el método recibe un diccionario como parámetro.

```
# se usa método delete_many con parámetros, a partir de la colección
# para eliminar un documento de la colección
print("Proceso para borrar un documento de una colección")
colección.delete_many({'numero_publicaciones':80})
```

Con lo anterior se elimina todos los documentos que tengan en el atributo `número_publicaciones` el valor igual a 80.



Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.

Le invito a profundizar sus conocimientos acerca de la generación de aplicaciones con base de datos.

2.4.2. Generación de aplicaciones con base de datos

Ejemplo 1: en el presente apartado se crea una aplicación mediante scripts de Python que permita registrar equipos de fútbol del campeonato ecuatoriano. Cada equipo de fútbol tiene características como: nombre, siglas, número de seguidores, número de campeonatos y nombre del estadio.

Se presenta tres soluciones:

1. Solución usando Python y SQLite ([acceso a código de la solución 1](#))

- Se crea la conexión a la base de datos

```
import sqlite3
conn = sqlite3.connect('base_solución1.db')
```

- Se crea la entidad en base de datos con SQL

```

from base_datos import conn

# se usa el objeto Connection y se accede al método cursor
# para poder realizar las acciones en la base de datos.

cursor = conn.cursor()

# a través de cursor su puede ejecutar comandos SQL mediante el método
# execute

# Crear una tabla denominada Equipo

cadena_sql = 'CREATE TABLE Equipo (nombre TEXT, siglas TEXT, \
seguidores INTEGER, campeonatos INTEGER, nombre_estadio TEXT)'

# ejecutar el SQL
cursor.execute(cadena_sql)

# cerrar el enlace a la base de datos (recomendado)
cursor.close()

from base_datos import conn

# se usa el objeto Connection y se accede al método cursor
# para poder realizar las acciones en la base de datos.

cursor = conn.cursor()

```

- Se ingresa, guarda y presenta información de la entidad equipo. Se usa estructuras propias de Python como: **list** y **tuple**.

```

# a través de cursor su puede ejecutar comandos SQL mediante el
método
# execute
# Se crea un proceso repetitivo para ingresar información por
teclado,
# de acuerdo a las características de la entidad Equipo
lista_equipos = []
bandera = True
while bandera:
    nombre = input("Ingrese el nombre del equipo: ")
    siglas = input("Ingrese las siglas del equipo: ")
    seguidores = int(input("Ingrese el número de seguidores: "))
    campeonatos = int(input("Ingrese el número de campeonatos: "))
    estadio = input("Ingrese el nombre del estadio: ")
    lista_equipos.append((nombre, siglas, seguidores, campeonatos,
estadio))
    salida = input("Desea salir del ciclo, ingrese la letra (f): ")
    if salida == 'f':
        bandera = False

```

```

print("fin de ingreso de información\n")
for l in lista_equipos:
    cadena_sql = """INSERT INTO Equipo (nombre, siglas, seguidores, \
    campeonatos, nombre_estadio) VALUES ('%s', '%s', %d, %d,
    '%s');""" % \
        (l[0], l[1], l[2], l[3], l[4])
    # ejecutar el SQL
    cursor.execute(cadena_sql)
    # confirmar los cambios
    conn.commit()

# hacer la consulta a la base de datos
cadena_consulta_sql = "SELECT * from Equipo"
cursor.execute(cadena_consulta_sql)
# la información resultante se la obtiene del método fetchall de
cursor.
informacion = cursor.fetchall()

# se realiza un ciclo repetitivo para recorrer la secuencia de
información
# resultante
print("Presentación de información de la base de datos")
for d in informacion:
    print("Nombre: %s - Siglas:%s - Seguidores: %d - Campeonatos: %d"
        "- Estadio: %s" % (d[0], d[1], d[2], d[3], d[4]))
# cerrar el enlace a la base de datos (recomendado)
cursor.close()

```

2. Solución usando Python, SQLAlchemy y SQLite ([acceso a código de la solución 2](#))

- Se crea la conexión a la base de datos.

```

from sqlalchemy import create_engine
# se genera en enlace al gestor de base de
# datos para el ejemplo se usa la base de datos
# sqlite
engine = create_engine('sqlite:///base_solucion2.db')

```

- Se crea la entidad en base de datos con SQLAlchemy

```

from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
from sqlalchemy import Column, Integer, String
# se importa el engine
from base_datos import engine

```

```

# se crea la clase llamada Base que permite definir las clases bajo las
# características de SQLAlchemy
Base = declarative_base()

# Se crea la una entidad llamada Equipo
class Equipo(Base):
    __tablename__ = 'mиеquipo' # El nombre de la entidad en sqlite
    # Se definen los atributos
    id = Column(Integer, primary_key=True) # este atributo es entero y
                                            # se lo considera como llave
                                            # primaria
    nombre = Column(String) # atributo de tipo String
    siglas = Column(String)
    seguidores = Column(Integer)
    campeonatos = Column(Integer)
    nombre_estadio = Column(String)

    def __str__(self):
        return ("Nombre: %s/ Siglas: %s/ Seguidores: %d/ Campeonatos:%d"
               "Estadio: %s") % (self.nombre, self.siglas, self.seguidores,
                                 self.campeonatos, self.nombre_estadio)

    # Sentencia que permite crear o migrar las clases en Python al
    # gestor de base de datos, expresado en el engine.
Base.metadata.create_all(engine)

```

- Se ingresa, guarda y presenta información de la entidad Equipo.

```

from sqlalchemy.orm import sessionmaker
# se importa la clase(s) del
# archivo crear_entidades
from crear_entidades import Equipo
# se importa el engine
from base_datos import engine

# Se crea una clase llamada Sesión,
# desde el generador de clases de SQLAlchemy
# sessionmaker.
Session = sessionmaker(bind=engine) # Se usa el engine
# Importante, se crea un objeto llamado session
# de tipo Session, que permite: permitir guardar, eliminar,
# actualizar y generar consultas a la base de datos.
session = Session()

```

```

lista_equipos = []
bandera = True
while bandera:
    nombre = input("Ingrese el nombre del equipo: ")
    siglas = input("Ingrese las siglas del equipo: ")
    seguidores = int(input("Ingrese el número de seguidores: "))
    campeonatos = int(input("Ingrese el número de campeonatos: "))
    estadio = input("Ingrese el nombre del estadio: ")
    # con la información ingresada por teclado
    # se crea un objeto de tipo equipo
    equipo = Equipo(nombre=nombre, siglas=siglas, seguidores=seguidores, \
campeonatos=campeonatos, nombre_estadio=estadio)
    # se lo agrega a la lista
    lista_equipos.append(equipo)
    salida = input("Desea salir del ciclo, ingrese la letra (f): ")
    if salida == 'f':
        bandera = False

print("fin de ingreso de información\n")

for l in lista_equipos:
    # cada objeto que se itera de la lista equipos
    # se lo agrega a la sesión
    session.add(l)

# se necesita confirmar los cambios que existan en la sesión
# se usa commit
session.commit()

# Obtener todos los registros de la entidad Equipo.
# Se hace uso del método query.
# all, significa que se obtiene todos los registros
informacion = session.query(Equipo).all()
# La variable información, tendrá un listado de objetos de tipo Equipo

# se realiza un proceso iterativo para presentar la información
# de cada objeto.
print("Presentación de información de la base de datos")
for l in informacion:
    print(l)

```

3. Solución usando Python y MongoDB (acceso a código de la solución 3)

- Se crea la colección en MongoDB con mongsh.

```

use solucion03
db.createCollection("equipos")

```

- Se crea la conexión a la base de datos; se usa la librería pymongo.

```
from pymongo import MongoClient
client = MongoClient('mongodb://localhost:27010/')
```

- Se ingresa, guarda y presenta información de la colección equipos.

```
from base_datos import client
# se obtiene la colección general (base de datos)
db = client.solucion03
# se obtiene la colección que almacenará la información, para el
ejemplo
# se llama equipos
colección = db.equipos

# conjunto de datos a guardar en la colección
# importante, aquí se usa la estructura de Python denominada
diccionario
# proceso que agrega un solo documento

lista_equipos = []
bandera = True
while bandera:
    nombre = input("Ingrese el nombre del equipo: ")
    siglas = input("Ingrese las siglas del equipo: ")
    seguidores = int(input("Ingrese el número de seguidores: "))
    campeonatos = int(input("Ingrese el número de campeonatos: "))
    estadio = input("Ingrese el nombre del estadio: ")
    # con la información ingresada por teclado
    # se crea un diccionario
    equipo = {"nombre": nombre, "siglas": siglas, "seguidores":
seguidores, \
              "campeonatos": campeonatos, "nombre_estadio": estadio}
    # se lo agrega a la lista
    lista_equipos.append(equipo)
    salida = input("Desea salir del ciclo, ingrese la letra (f): ")
```

```

if salida == 'f':
    bandera = False

print("fin de ingreso de información\n")
# proceso que agrega una lista de documentos
colección.insert_many(lista_equipos)

# se usa método find, a partir de la colección
print("Presentación de información de la base de datos")
información = colección.find()
for registro in información:
    print("Nombre: %s - Siglas:%s - Seguidores: %d - Campeonatos: %d"
        "- Estadio: %s" % (registro['nombre'], registro['siglas'],
        registro['seguidores'], registro['campeonatos'],
        registro['nombre_estadio']))

```

Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.



Actividades de aprendizaje recomendadas

- **Actividad 1:** las temáticas revisadas en la presente sección han sido las relacionadas a la generación de soluciones que involucran el uso de base de datos relacionales y no relaciones desde lenguaje Python. En esta actividad se sugiere verificar el contenido del repositorio denominado [Ejemplo de uso de la SQLAlchemy](#); donde se hace una explicación de cómo crear y poblar las entidades, y hacer consultas a los registros guardados en la base de datos, a través de la librería SQLAlchemy como ORM del lenguaje Python y la base de datos SQLite. Es importante que se sigan las indicaciones del repositorio, con el objetivo que se pueda recrear el ejemplo en sus entornos locales. Como recomendación final, usted debe leer el código de los archivos del repositorio de forma pausada, de tal forma que sea comprendido y pueda producir su propia entidad(es) con registros nuevos.

Actividad 2: resolver la autoevaluación 3; en la actividad se presentan un conjunto de preguntas basadas en las temáticas descritas en la unidad 2: desarrollo de aplicaciones con base de datos; sección 2.2. Programación con bases de datos. En la parte introductoria de la actividad se listan los contenidos de la guía didáctica, que se debe revisar para contestar las interrogantes. Recuerde que la autoevaluación le permitirá identificar el

nivel de aprendizaje y comprensión alcanzado; así mismo, determinar los ítems que se debe volver a revisar para lograr los resultados de aprendizaje propuestos. Adicionalmente, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado. Finalmente, después de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección solucionario de la guía didáctica.



Autoevaluación 3

Estimado estudiante, es momento de medir el entendimiento de algunos conceptos estudiados en la unidad 2: desarrollo de aplicaciones con base de datos punto 2.2. Programación con bases de datos. Se sugiere, antes de la autoevaluación, realizar las siguientes acciones:

- Revisar todo el contenido de las temáticas indicadas.
- Leer los recursos educativos recomendados.
- Hacer organizadores gráficos para ordenar la información para su estudio.

La autoevaluación le permitirá reafirmar los conceptos estudiados. Los enunciados tienen una sola opción correcta, lea de forma atenta cada respuesta y seleccione la opción que usted considere correcta.

1. ¿Cuál de las siguientes sentencias permite crear una conexión a una base de datos SQLite?

a.

```
import sqlite3  
conn = sqlite3.connect('base_ejemplo.db')
```

b.

```
from sqlite3 import sqlite3  
conn = sqlite3.connect('base_ejemplo.db')
```

c.

```
import sqlite3  
conn = sqlite3('base_ejemplo.db')
```

2. Identifique las líneas de código que permiten crear en un base de datos SQLite, una entidad llamada Ciudad, con atributo nombre y población. Se asume que la importación, permite el enlace la base de datos.

a.

```
from base_datos import conn  
cursor = conn.cursor()  
cadena_sql = 'CREATE TABLE Ciudad (nombre TEXT, poblacion  
INTEGER)'  
cursor.close()
```

b.

```
from base_datos import conn  
cadena_sql = 'CREATE TABLE Ciudad (nombre TEXT, poblacion  
INTEGER)'  
cursor.execute(cadena_sql)  
cursor.close()
```

c.

```
from base_datos import conn  
cursor = conn.cursor()  
cadena_sql = 'CREATE TABLE Ciudad (nombre TEXT, poblacion  
INTEGER)'  
cursor.execute(cadena_sql)  
cursor.close()
```

3. Dadas las siguientes sentencias, encuentre las que permiten insertar información en una entidad llamada persona.

a.

```
apellido = "José Lima"  
pasaporte = "10012391xa1"  
cadena_sql = """INSERT Persona (apellido, pasaporte) VALUES ("%s',  
'%s');""" % (apellido, pasaporte)  
cursor.execute(cadena_sql)
```

b.

```
apellido = "José Lima"  
pasaporte = "10012391xa1"  
cadena_sql = """INSERT INTO Persona (apellido, pasaporte) VALUES  
('%s', '%s');""" % (apellido, pasaporte)  
cursor.execute(cadena_sql)
```

c.

```
apellido = "José Lima"  
pasaporte = "10012391xa1"  
cadena_sql = """INSERT INTO Persona (apellido, pasaporte) ("%s',  
'%s');""" % (apellido, pasaporte)  
cursor.execute(cadena_sql)
```

4. Identifique una característica de los ORM.

- a. Tratar la información con estructuras de datos que contienen objetos.
- b. Uso exclusivo de un motor de base de datos.
- c. El uso exclusivo de los conceptos de programación secuencial.

5. Identifique librerías ORM para lenguajes PHP y Python

- a. Eloquent / Sequel.
- b. Eloquent / Pony ORM.
- c. Hibernate / Pony ORM.

- 6. ¿Cuál es la forma correcta de instalar las librerías ORM, SQLAlchemy?**
- a. pip install SQLAlchemy.
 - b. pip SqlAlchemy.
 - c. install pip SQLAlchemy.
- 7. Identifique la línea de código que permitirá agregar una atributo llamada id y que sea considerado como llave primaria, a un clase bajo la filosofía de SQLAlchemy.**
- a. id = Primary_Key(Integer, column=True).
 - b. id = Column(primary_key=True).
 - c. id = Column(Integer, primary_key=True).
- 8. ¿Cuál de las siguientes sentencias permite realizar una consulta de todos los registros de una entidad llamada pais, a través de SQLAlchemy ?**
- a.
`p = Pais().
session.query(p).all()`
 - b.
`session.filter(Pais).all()`
 - c.
`session.query(Pais).all()`

- 9. Seleccione las líneas de código que permiten realizar una conexión a la base de datos MongoDB desde Python.**

a.

```
from pymongo import Client.  
client = Client('mongodb://localhost:27010/').
```

b.

```
from pymongo import MongoClient.  
client = MongoClient('mongodb://localhost:27010/').
```

c.

```
from pymongo import Mongo.  
client = Mongo('mongodb://localhost:27010/').
```

- 10. Si se requiere insertar en una base de datos MongoDB la un registro. ¿Cuál de las siguientes, es una opción válida?**

a.

```
data_01 = {"nacionalidad":"ecuatoriana", "numero_publicaciones":  
100}  
coleccion.insert_one(data_01[0]).
```

b.

```
data_01 = {"nacionalidad":"ecuatoriana", "numero_publicaciones":  
100}  
coleccion.insert_many(data_01).
```

c.

```
data_01 = {"nacionalidad":"ecuatoriana", "numero_publicaciones":  
100}  
coleccion.insert_one(data_01).
```

[Ir a solucionario](#)



Semana 7

Estimado estudiante, durante la semana 7 de estudio, se recomienda volver a revisar, analizar y contestar las preguntas de las autoevaluaciones y las actividades recomendadas del primer bimestre.

A continuación se detalla un ejemplo que permite reforzar los contenidos estudiados en el bimestre.

Ejemplo 01

Se debe generar una aplicación en lenguaje Python que permita agregar información a una entidad llamada empleados. Cada empleado tiene características como: nombre, apellido, edad, sueldo. La información debe ser ingresada por teclado y guardada en una base de datos SQLite. Al momento de presentación de la información, se debe listar los registros de la base de datos y además debe presentar el promedio de edades y sueldos de los empleados ingresados.

Solución:

- Se crea la conexión a la base de datos.

```
import sqlite3  
conn = sqlite3.connect('base_solucion1.db')
```

- Se crea la entidad en base de datos con SQLite.

```
from base_datos import conn  
  
# se usa el objeto Connection y se accede al método cursor  
# para poder realizar las acciones en la base de datos.  
  
cursor = conn.cursor()  
  
# a través de cursor se puede ejecutar comandos SQL mediante el método
```

```

# execute

# Crear una tabla denominada Equipo

cadena_sql = 'CREATE TABLE Empleado (nombre TEXT, apellido TEXT, \
edad INTEGER, sueldo FLOAT)'

# ejecutar el SQL
cursor.execute(cadena_sql)

# cerrar la enlace a la base de datos (recomendado)
cursor.close()

```

- Se ingresa y guarda información de la entidad empleado. Se usa estructuras propias de Python como: **list** y **tuple**.

```

from base_datos import conn

# se usa el objeto Connection y se accede al método cursor
# para poder realizar las acciones en la base de datos.

cursor = conn.cursor()

# a través de cursor se puede ejecutar comandos SQL mediante el método
# execute
# Se crea un proceso repetitivo para ingresar información por teclado,
# de acuerdo a las características de la entidad Empleado

lista_empleados = []
bandera = True

while bandera:
    nombre = input("Ingrese nombre del empleado: ")
    apellido = input("Ingrese apellido del empleado: ")
    edad = int(input("Ingrese la edad del empleado: "))
    sueldo = float(input("Ingrese el sueldo del empleado:"))
    lista_empleados.append((nombre, apellido, edad, sueldo))
    salida = input("Desea salir del ciclo, ingrese la letra (f): ")
    if salida == 'f':
        bandera = False

print("fin de ingreso de información\n")
# proceso para guardar la información en la base de datos
for l in lista_empleados:
    cadena_sql = """INSERT INTO Empleado (nombre, apellido, \
edad, sueldo) VALUES ('%s', '%s', %d, %f);""" % \
(l[0], l[1], l[2], l[3])
    # ejecutar el SQL
    cursor.execute(cadena_sql)
    # confirmar los cambios
    conn.commit()

```

- Se presenta información de la entidad, según los requerimientos.

```

from base_datos import conn

# se usa el objeto Connection y se accede al método cursor
# para poder realizar las acciones en la base de datos.

cursor = conn.cursor()

# hacer la consulta a la base de datos
cadena_consulta_sql = "SELECT * from Empleado"
cursor.execute(cadena_consulta_sql)
# la información resultante se la obtiene del método fetchall de cursor.
informacion = cursor.fetchall()

# se realiza un ciclo repetitivo para recorrer la secuencia de información
# resultante
print("Presentación de información de la base de datos")
# variable acumuladoras
suma_edades = 0
suma_sueldos = 0
cadena_presentacion = ""

for d in informacion:
    cadena_presentacion = "%sNombre: %s - Apellido: %s - Edad: %d - Sueldo:
%.2f\n" % (cadena_presentacion, d[0], d[1], d[2], d[3])
    suma_edades = suma_edades + d[2]
    suma_sueldos = suma_sueldos + d[3]

promedio_edades = float(suma_edades) / len(informacion)
promedio_sueldos = float(suma_sueldos) / len(informacion)
cadena_presentacion = "%s\nPromedio Edades: %.2f\nPromedio Sueldos: %.2f\n"
% (cadena_presentacion, promedio_edades, promedio_sueldos)
print(cadena_presentacion)

# cerrar el enlace a la base de datos (recomendado)
cursor.close()

```



Estimado estudiante, el ejemplo completo se lo puede descargar del repositorio de Github y realizar los cambios para su estudio. Usted puede crear una solución para la misma problemática, usando SQLAlchemy.



Semana 8



Actividades finales del bimestre

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

Se propone para la presente semana el repaso de las siguientes temáticas:

- **Unidad 1:** ciclo de vida de software (ISO/IEC 12207:2017).
- **Unidad 2:** desarrollo de aplicaciones con base de datos.
 - 2.1 Control de versiones.
 - 2.2. Programación con bases de datos.

Las estrategias sugeridas son:

- Elaboración de resúmenes, cuadros sinópticos, mapas conceptuales, etc; de las temáticas del bimestre.
- Revisar las actividades recomendadas en las semanas de estudio. Cuando lo amerite, recree los ejercicios desde cero, con el objetivo de familiarizarse con las herramientas y librerías usadas.
- Revisar los recursos de aprendizaje recomendados en el plan docente de la asignatura.
- Plantearse ejercicios y problemáticas similares a las expuestas en los contenidos de la guía didáctica.
- Revisar las preguntas que conforman las autoevaluaciones.
- Revisar las preguntas que conforman los cuestionarios de repaso del bimestre planteados en el entorno virtual de aprendizaje.

¿Estamos listos?



Avancemos al segundo bimestre.



Segundo bimestre

Resultado de aprendizaje 3

- Aplica conocimiento funcional y capacidades para todos los aspectos involucrados en la construcción de una aplicación de software.

Para alcanzar el resultado de aprendizaje a nivel de conocimiento funcional y de capacidades en el desarrollo de aplicaciones, las siguientes semanas de estudio están divididas en temáticas que abordan el desarrollo en el ámbito de aplicaciones de escritorio y aplicaciones de ambiente web; además, se estudia temáticas relacionadas con la programación multihilo, donde se destacan conceptos y ejemplos. Lo anterior consolida en usted estimado estudiante las bases necesarias para comprender los procesos que pueden estar involucrados en la construcción de una aplicación de software.

Contenidos, recursos y actividades de aprendizaje



Semana 9

Unidad 3. Conceptos avanzados de programación

3.1. Programación en capas



Para las temáticas de este apartado se va a trabajar con el recurso denominado [Desarrollo web en entorno servidor](#). Se solicita leer detenidamente [el capítulo cuatro, donde se detallan conceptos relacionados la Generación dinámica de páginas web](#).

En el recurso se hace referencia a la ventaja que existe en la generación de páginas web dinámicas, logrando tener independencia entre la parte del diseño de páginas web y la lógica del negocio. Se abordan temas como manejo de HTML, hojas de estilo, JavaScript, de manera introductoria; con la finalidad de entender algunos conceptos en el marco de la programación en capas.

En el recurso se habla de los mecanismos de separación de la lógica de negocio. Al inicio del apartado se habló de la separación entre el diseño de una página web y su lógica. Lo anterior se lo asume con una decisión arquitectónica, que involucra aspectos como: el modelo físico y el modelo lógico.

- En modelos físicos arquitectónicos o arquitectura física, se detalla la forma como se distribuye la infraestructura o el *hardware*, que son los encargados de ejecutar los procesos del sistema. La mayoría de sistemas hablan de las arquitecturas multinivel, de las cuales se desprende la arquitectura cliente-servidor. Dicha arquitectura busca que los elementos que generen alguna actividad distinta sea representado en una capa diferente.

A nivel de aplicaciones web los diversos sistemas usan arquitectura de dos o más niveles; lo que significa que cada una de las tareas se dividen en equipos de cómputo distinto. En la figura 22 y la figura 23, se detalla un ejemplo de arquitectura de dos y tres niveles respectivamente.

Figura 22.

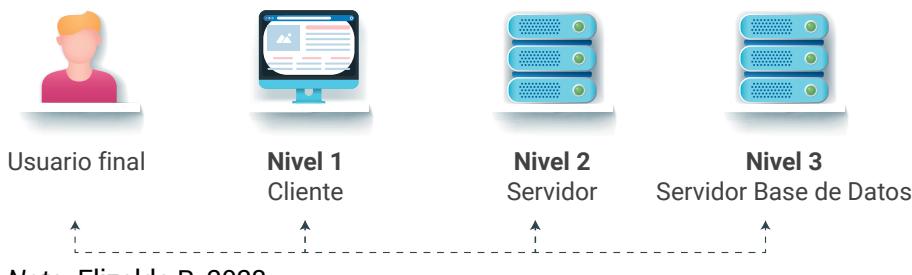
Ejemplo de arquitectura de dos niveles



Nota. Elizalde R, 2023

Figura 23.

Ejemplo de arquitectura de tres niveles



- Los modelos de arquitectura lógica del sistema, se refiere al proceso de separar o dividir el *software*, con el objetivo de obtener un mejor rendimiento. Se trata de organizar el código, se busca la organización y funcionalidad.

Entre las características se tiene:

- Generar desarrollos en paralelo.
- Generar aplicaciones más robustas.
- Mantenimientos posteriores sencillos.
- Soluciones flexibles y escalables.

Cuando se habla de arquitecturas en capas, se asocia al patrón arquitectónico de nombre: Modelo – vista – controlador (MVC).

Según manifiesta Elizalde (2020), al conceptualizar el MVC asociado a los *frameworks* de desarrollo, indica que la utilización del patrón permite la reutilización de código, facilitar los procesos de desarrollo y mejorar el mantenimiento a futuro de las aplicaciones. Además, menciona algunas características propias de los *frameworks*, tales como: rapidez en los tiempos de desarrollo a través del uso de propiedades de cada *framework*; mejorar el proceso de corrección de posibles errores en la construcción de la solución; uso de pruebas de desarrollo para garantizar que las funcionalidades cumplan con su objetivo; uso de componentes propios de los *framework*, asociados, por ejemplo, a la seguridad.

El recurso usado en la presente sección, explica que la utilización del esquema o patrón Modelo-Vista-Controlador (MVC) se lo realiza en aplicaciones *web* y aplicaciones *standalone*. En el MVC, se consideran los siguientes conceptos: Modelo, Vista y Controlador.

- **Modelo**, se encargan de construir la lógica de la aplicación, por lo general estos modelos trabajan directamente con una base de datos.
- **Vista**, tiene relación con las interfaces de usuario que se desarrollan. En una aplicación de ambiente web, es considerado como el conjunto de páginas web, donde los usuarios ingresan información.
- **Controlador**, su tarea es de administrar la interacción del usuario y la base de datos, luego, direccionar el flujo de datos a la vista o template correcto.

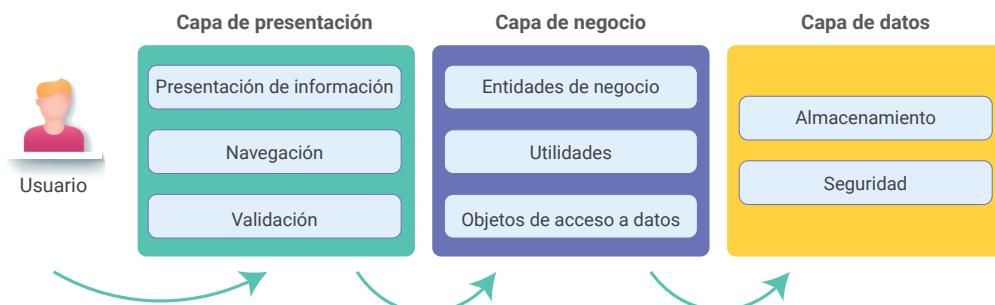
La implementación del patrón MVC, permite las siguientes relaciones:

- La lógica de la interfaz se relaciona con la vista.
- La lógica del negocio tiene relación con el modelo.
- La lógica de entradas tiene una relación el controlador.

En el marco de las arquitecturas multicapa para el desarrollo, existe la arquitectura 3 capas, que sigue siendo usado en los procesos de construcción de soluciones. Dicha arquitectura, realiza una división en tres niveles: capa de presentación, capa de la lógica y capa de datos o persistencia.

Se busca que los niveles o capas estén constituidos por diversos componentes, que buscan estar relacionados con las capas asociadas. Ver figura 24.

Figura 24.
Arquitectura tres capas en aplicaciones web



3.1.1. Capa de presentación

Las principales características en esta capa son:

- Se la conoce también como la capa de interfaz gráfica.
- Tiene comunicación con la capa lógica.
- Se presenta como la interfaz que visualiza el usuario.
- Recolecta la información que el usuario ingresa.

3.1.2. Capa lógica

Pose las siguientes características:

- En algunos entornos se la conoce como capa de negocio.
- Gestiona o administra las funcionalidades del sistema; se habla de las reglas del negocio.
- Recibe las diversas peticiones del usuario.
- Se procesa la información, y se devuelve la mejor respuesta posible. Se interrelaciona con la capa de datos en algunas oportunidades.
- Se debe gestionar que las reglas del negocio cumplan y satisfagan los requerimientos.
- Se puede desarrollar esta capa en el lado cliente, como en el servidor.
- Tareas puntuales: toma de decisiones en función de lo solicitado, cálculos, proceso de información.

3.1.3. Capa de datos

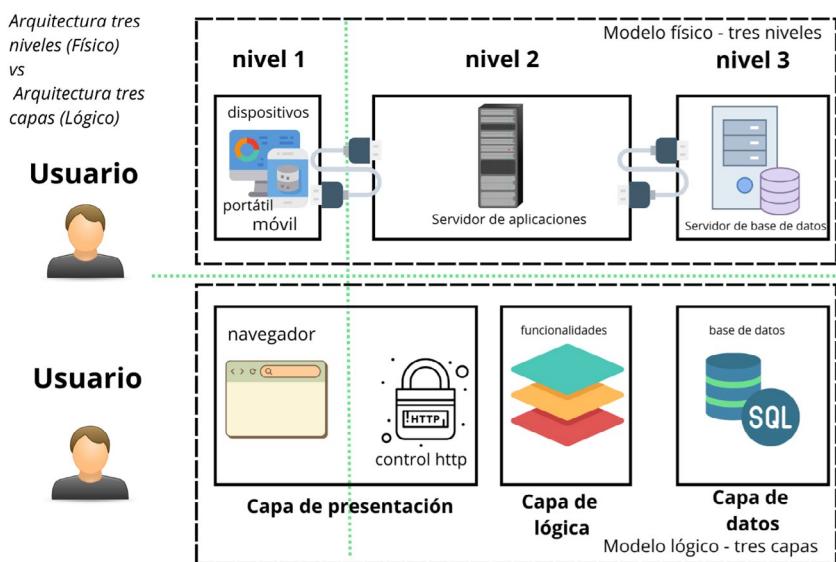
Sus principales características:

- Se encarga de administrar el acceso a los datos.
- Contiene los procesos para el manejo y almacenamiento de las entidades generadas.
- Está conformada por los enlaces a los gestores de base datos.
- Espera las peticiones de la capa de lógica, para almacenar, actualizar, y eliminar información.

En la figura 25, se describe la relación entre un modelo físico y lógico. Resaltar que un nivel del modelo físico, puede incluir varias capas de la arquitectura lógica; siempre de acuerdo a los contextos particulares de cada desarrollo.

Figura 25.

Arquitectura tres niveles (*Físico*) vs Arquitectura tres capas (*Lógico*)



Nota. Elizalde R, 2023



Semana 10

3.2. Programación en capas

3.2.1. Generación de aplicaciones usando capas



En la presente guía se usa el *framework* de ambiente web denominado Django para el proceso de construcción de aplicaciones usando capas. Se recomienda la lectura del recurso educativo: [La guía definitiva de Django: Desarrolla aplicaciones web de forma rápida](#)

y sencilla, en lo que respecta a los capítulos relacionados a los conceptos generales, plantillas, modelos, vistas y formularios.

Para la utilización del *framework* Django, se recomienda, inicialmente lo siguiente:

- Instalar el lenguaje de programación [Python](#).
- Instalar Django, se usa el proceso de instalación, a través del administrador de paquetes llamado [Pip](#).
- En un terminal/CMD/consola, usar: `pip install django`.
- Lectura de la información [oficial del framework Django](#).

Según Elizalde (2020), algunas de las características del *framework* son: ahorro de tiempo en la construcción de aplicaciones; uso de atajos para tareas frecuentes, uso de la filosofía Modelo-Vista-Template (similar al patrón MVC); acoplamiento débil en el desarrollo de componentes. Django está compuesto por un proyecto y una o muchas aplicaciones.

Los principales atajos que se usa en el desarrollo de proyectos y aplicaciones en Django son:

- Creación de un proyecto, a través de: `django-admin startproject [nombre-proyecto]`.
- Creación de las tablas iniciales y migraciones de cambios a la base datos: `python manage.py migrate`.
- Inicio del servidor web ligero, propio de Django: `python manage.py runserver`.
- Crear una aplicación para el proyecto de Django: `python manage.py startapp [nombre-aplicación]`.
- Uso de la consola de Django: `python manage.py shell`.

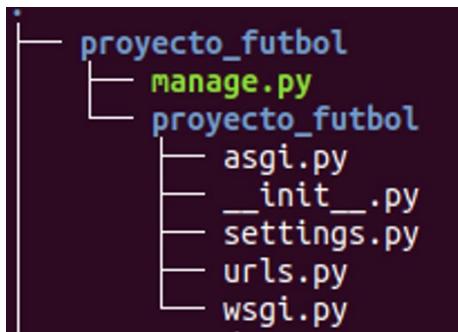
Ejemplo 1, a continuación, se detalla el proceso para la creación de un proyecto en Django que permita administrar equipos de fútbol. Las características de cada equipo son: nombre el equipo, siglas del equipo, número de seguidores, número de campeonatos y nombre de su estadio.

1. Crear el proyecto de Django, usar el nombre: `projeto_futbol`
 - `django-admin startproject projeto_futbol`.

Lo anterior, genera la estructura de carpetas y archivos que se indica en la figura 26.

Figura 26.

Estructura inicial de un proyecto de Django



Nota. Elizalde R, 2023

Una descripción de los archivos más importantes creados, se detalla a continuación:

- Manage.py; archivo que permite a través de la línea de comandos crear aplicaciones, acceder a una consola interactiva, actualizar la base de datos, crear usuarios administradores para el proyecto, entre otras tareas.
 - Carpeta proyecto_futbol; carpeta que contiene los archivos propios del proyecto.
 - __init__.py; indica a Python que el directorio debe ser considerado como paquete.
 - Settings.py; archivo de importancia alta, pues, contiene las configuraciones del proyecto de Django.
 - Urls.py; maneja las direcciones internas del proyecto de Django. Luego se asocia a las urls de la aplicación.
 - Wsgi.py y asgi.py; se los usa cuando se requiere ubicar el proyecto en un servidor web como [Apache](#) o [Nginx](#).
2. Ejecutar el proceso para inicializar las entidades en la base de datos. En el archivo settings.py, específicamente en la variable databases,

se puede indicar el gestor de base de datos para el proyecto. En el ejemplo planteado se usará la base de datos por defecto de los proyectos de Django, SQLite. Ver figuras 27, 28 y 29.

- python manage.py migrate.

Figura 27.

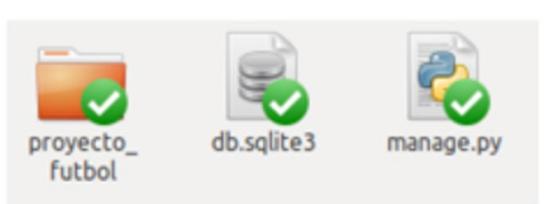
Uso y salida del comando migrate.

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

Nota. Elizalde R, 2023

Figura 28.

Se añade el archivo de la base de datos.



Nota. Elizalde R, 2023

Figura 29.

Verificar las tablas creadas en la base de datos.

The screenshot shows a database structure viewer with two panes. The left pane lists tables and indices:

- Tables (11): auth_group, auth_group_permissions, auth_permission, auth_user, auth_user_groups, auth_user_user_permissions, django_admin_log, django_content_type, django_migrations, django_session, sqlite_sequence.
- Indices (15): auth_group_permissions_group_id_312... (index), auth_group_permissions_group_id_per... (index), auth_group_permissions_permission_id_... (index), auth_permission_content_type_id_2f618e3b... (index), auth_permission_content_type_id_cade... (index), auth_user_groups_group_id_97501564 (index).

The right pane displays the SQL code for creating these tables and indices:

```
CREATE TABLE "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT);
CREATE TABLE "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY A
CREATE TABLE "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT);
CREATE TABLE "auth_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT);
CREATE TABLE "auth_user_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT);
CREATE TABLE "auth_user_user_permissions" ("id" integer NOT NULL PRIMARY KEY
CREATE TABLE "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT);
CREATE TABLE "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT);
CREATE TABLE "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT);
CREATE TABLE "django_session" ("session_key" varchar(40) NOT NULL PRIMARY KEY);
CREATE TABLE sqlite_sequence("name" text);
```

Nota. Elizalde R, 2023

3. Se crea una aplicación para el proyecto:

- python manage.py startapp campeonato.

Agregar en la variable de tipo lista, de nombre INSTALLED_APPS, del archivo settings.py, la aplicación creada, llamada: campeonato, ver figura 30.

Figura 30.

Agregar en la variable de tipo lista llamada INSTALLED_APPS, la aplicación creada

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'campeonato'
]
```

Nota. Elizalde R, 2023

Las carpetas y archivos generados en la aplicación son:

- Carpeta campeonato; contiene los archivos propios de la aplicación.

- *Models.py*; archivo donde se codifican las clases en Python, mismas que luego se convierten en tablas. Aquí se debe usar la documentación de Django para definir los atributos y tipos de datos de cada entidad de la base de datos.
 - *Admin.py*; ayuda a gestionar las clases o entidades que serán puestas en el proceso de administración del proyecto. Característica muy importante en Django.
 - *Views.py*; permite gestionar través de funciones, principalmente, la lógica de aplicación.
4. Agregar la entidad descrita en la problemática al archivo *models.py*. Tomar como referencia las siguientes líneas de código.

```
from django.db import models

class Equipo(models.Model):
    nombre = models.CharField(max_length=30)
    siglas = models.CharField(max_length=30)
    seguidores = models.IntegerField()
    campeonatos = models.IntegerField()
    estadio = models.CharField(max_length=100)

    def __str__(self):
        return "%s %s %s" % (self.nombre,
                             self.siglas,
                             self.seguidores,
                             self.campeonatos,
                             self.estadio)
```

La creación de la clase, sigue la filosofía de propia de Django, [a través de su documentación oficial](#).

Finalmente, se realiza la migración hacia la base de datos. Usar los siguientes comandos:

- *python manage.py makemigrations equipo*.
- *python manage.py migrate*.

Verificar en la base de datos, la creación de las entidades correspondientes a la aplicación. Ver figura 31 y 32, para una mejor comprensión.

Figura 31.

Salida del comando *makemigrations* y *migrate*.

Salida del comando:

```
python manage.py makemigrations campeonato
```

```
Migrations for 'campeonato':  
  campeonato/migrations/0001_initial.py  
    - Create model Equipo
```

Salida del comando:

```
python manage.py migrate
```

```
Operations to perform:  
  Apply all migrations: admin, auth, campeonato, contenttypes, sessions  
Running migrations:  
  Applying campeonato.0001_initial... OK
```

Nota. Elizalde R, 2023

Figura 32.

Verificar la creación de la entidad en la base de datos.

Tables (12)	
Name	Type
auth_group	
auth_group_permissions	
auth_permission	
auth_user	
auth_user_groups	
auth_user_user_permissions	
campeonato_equipo	
django_admin_log	
django_content_type	
django_migrations	
django_session	
sqlite_sequence	

Nota. Elizalde R, 2023

5. Crear un usuario administrador del proyecto. Usar el comando:

```
python manage.py createsuperuser.
```

Agregar la información que se solicite en cada pregunta.

6. Usar una característica propia y potente de Django; agregar la entidad del models.py al archivo admin.py, con el objetivo de acceder a un CRUD (proceso de creación, lectura, actualización y borrado) de los registros de la entidad; desde la interfaz administrativa de Django.

```
from django.contrib import admin

# Importar las clases del modelo
from campeonato.models import Equipo

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

Luego, ejecutar el comando:

```
python manage.py runserver
```

Posterior, ingresar la dirección <http://127.0.0.1:8000/admin> en un navegador. Finalmente, verificar que en la interfaz de administración exista la entidad equipo; luego, agregar registros y revisar que se haya guardado en la base de datos lo generado en la interfaz, para una mejor comprensión le invito a observar el siguiente vídeo presentación.

[Ingresar y verificar información en la entidad, a través de la interfaz de admin de Django](#)

7. Agregar una función en el archivo views.py, llamada *listar*. La misma debe permitir obtener de la base de datos, todos los registros de la entidad equipo. La información debe ser visualizada en un template. Le invito a revisar el documento [Pasos para agregar una función en el archivo views.py, llamada listar](#).

Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio en su entorno local.



Actividades de aprendizaje recomendadas

- **Actividad 1:** el siguiente recurso denominado [Curso de Django: Crear entorno virtual y Pip](#) le va a permitir reforzar su conocimiento, en lo referente a la instalación de librerías en lenguaje Python. El recurso hace uso del administrador de paquetes llamado Pip. A través del administrador de librerías se instala Django, luego se explica algunos comandos estudiados en la guía, como: django-admin. Importante, en el recurso se habla de los entornos virtuales en Python, el uso de los mismos está fuera del alcance de la guía didáctica, pero se cree conveniente conocer de manera general su uso en el entornos de desarrollo del lenguaje Python. Le invito a que pueda recrear el ejemplo del recurso en su entorno local.
- **Actividad 2:** la puesta en producción de una aplicación de ambiente web desarrollada en lenguaje Python es un tema importante y relevante. Por tal razón, con el objetivo de conocer el proceso de manera general, se solicita la lectura y análisis del siguiente repositorio denominado [Django en producción a través de gunicorn y nginx](#). En el recurso se explica la puesta en producción de una aplicación de Django, a través del uso de servidor web Nginx; los comandos y configuraciones son para el sistema operativo Ubuntu. Se solicita recrear el ejemplo en sus entornos locales, seguro será una buena experiencia que aportara en reforzar conceptos generales del ámbito del desarrollo de aplicaciones web.
- **Actividad 3:** resolver la autoevaluación 4; en la actividad se presentan un conjunto de preguntas basadas en las temáticas descritas en la unidad 3: conceptos avanzados de programación, sección: 3.1. Programación en capas. En la parte introductoria de la actividad se listan los contenidos de la guía didáctica, que se debe revisar para contestar las interrogantes. Recuerde que la autoevaluación le permitirá identificar el nivel de aprendizaje y comprensión alcanzado; así mismo, determinar los ítems que se debe volver a revisar para lograr los resultados de aprendizaje propuestos. Adicionalmente, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Finalmente, después de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección solucionario de la guía didáctica.



Autoevaluación 4

Estimado estudiante, es momento de medir el entendimiento de algunos conceptos estudiados en la unidad 3: conceptos avanzados de programación en el punto 3.1. Programación en capas. Considerar las siguientes acciones, antes de iniciar la autoevaluación.

- Revisar todo el contenido de las temáticas indicadas.
- Leer los recursos educativos recomendados.
- Hacer organizadores gráficos para la ordenar la información para su estudio.

La autoevaluación le permitirá reafirmar los conceptos estudiados. Los enunciados tienen una sola opción correcta, lea de forma atenta cada respuesta y seleccione la opción que usted considere correcta.

1. Seleccione la idea correcta que representa el concepto de los modelos físicos arquitectónicos.

- a. En los modelos físicos arquitectónicos se detalla la forma de cómo se distribuye la infraestructura o el *hardware* que son los encargados de ejecutar los procesos del sistema.
- b. En los modelos físicos arquitectónicos se detalla la forma de cómo se distribuye la infraestructura o el *software* que son los encargados de ejecutar los procesos del sistema.
- c. En los modelos físicos arquitectónicos se detalla la forma de cómo se distribuye la infraestructura o el *hardware* que son los encargados de ejecutar la base de datos exclusiva del sistema.

2. Identifique una característica de los modelos de arquitectura lógica del sistema.

- a. Genera aplicaciones con escalabilidad menor.
- b. Mantenimientos posteriores que requieren nuevos estudios.
- c. Soluciones flexibles y escalables.

- 3. Dentro de la arquitectura en capas; ¿Cuál es el nombre del patrón arquitectónico asociado?**
- a. Event-based pattern.
 - b. Modelo – Vista - Controlador.
 - c. Segregación de Responsabilidades de Comandos y Consulta.
- 4. En relación al patrón Modelo – Vista – Controlador. Del siguiente listado de ideas, indique la que se aproxime al modelo.**
- a. Su tarea es de administrar la interacción del usuario y la base de datos.
 - b. Se encarga de la lógica de la aplicación; trabajan directamente con una base de datos.
 - c. Son las interfaces donde los usuarios ingresan información.
- 5. Seleccione una característica de la capa lógica.**
- a. Recolecta la información que el usuario ingresa.
 - b. Contiene los procesos para el manejo y almacenamiento de las entidades generadas.
 - c. Gestiona o administra las funcionalidades del sistema; se habla de las reglas del negocio.
- 6. ¿Cuál es el comando para la creación de un proyecto en el framework Django?**
- a. django-admin startproject [nombre-proyecto].
 - b. django-admin project [nombre-proyecto].
 - c. django startproject [nombre-proyecto].
- 7. ¿Cuál es el comando que permite inicializar las entidades de la base de datos de un proyecto en Django?**
- a. python manage.py shell.
 - b. python manage.py migrate.
 - c. python manage.py admin.

- 8. Identifique el comando que se usa para generar una aplicación en un proyecto de Django.**
- a. python manage.py startapp [nombre de la app].
 - b. python startapp [nombre de la app].
 - c. python manage.py startapp init [nombre de la app].
- 9. Dadas las siguientes líneas de código, seleccione la línea que permite agregar el modelo *Jugador* al admin de Django. Se asume que se usa las líneas previas de código.**
- ```
from django.contrib import admin.
from campeonato.models import Jugador.
```
- a. admin.register(Jugador).
  - b. admin.site.register(Jugador).
  - c. admin.site(Jugador).
- 10. Dada una entidad llamada Jugador, en un modelo de una aplicación de Django. ¿Cuál es la forma para poder obtener todos los registros de la base de datos, a través del ORM de Django?**
- a. Jugador.objects.all().
  - b. Jugador.objects.filter().
  - c. Jugador.objects.count.all().

[Ir a solucionario](#)



### 3.3. Programación multihilo



Estimados estudiantes, para el estudio de la programación multihilo se solicita realizar una lectura comprensiva de la [sección de nombre Programación Paralela](#) del recurso educativo [Python Práctico](#), en el mismo se describen conceptos que ayudarán a la comprensión de las temáticas relacionadas a programación multihilo.

#### 3.3.1. Conceptos de programación multihilo

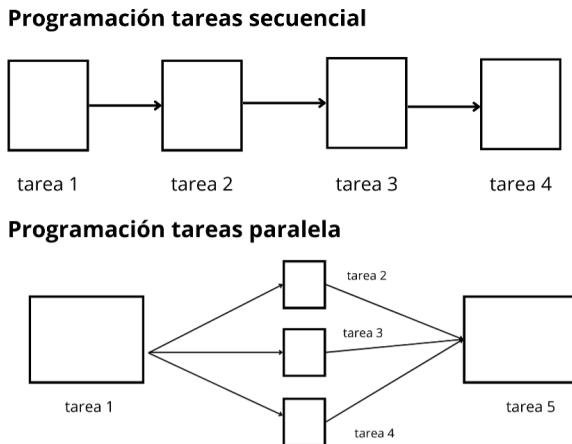
En la mayoría de campos de las tecnologías de la información, actualmente se reconoce el término del paralelismo; la exposición de grandes volúmenes de datos, por ejemplo, lleva consigo buscar mecanismos para acelerar procesos de transformación o carga.

El recurso menciona que el principal objetivo de la programación en paralelo es poder resolver los tiempos elevados que puedan existir al momento de ejecutar un proceso. La programación en paralelo se la puede definir como la estrategia para usar recursos computacionales, resolviendo problemas de forma simultánea.

En la figura 34, se puede apreciar la diferencia en la ejecución de procesos de forma secuencial y paralela.

**Figura 33.**

Ejecución de procesos secuencial y paralelo



Nota. Tomado de Moreno (2019).

En el recurso se menciona los siguientes pasos de forma general para poder resolver un problema de forma paralela:

- Intenta dividir el problema en subproblemas independientes, para resolver de forma simultánea.
- Generar una solución para cada subproblema encontrado.
- Ejecutar cada subproblema en un procesador.
- Mantener el control de los subproblemas, haciendo uso de mecanismos de supervisión general o global.

A continuación, se plantea una clasificación de los tipos de paralelismo:

1. En función del método:

- A nivel de bit, tiene relación con tamaño de la cadena de bits, que indican que tipo de instrucciones puede realizar un procesador, para generar el proceso de forma simultánea.
- A nivel de instrucción, agrupar las tareas para que el proceso de ejecución en paralelo obtenga el mismo resultado.
- A nivel de datos, dividir en varios procesos los datos que se necesita ejecutar.

- A nivel de tareas, separar las tareas independientes entre el conjunto de procesos existentes.
2. Indicando el código y cómo se realiza el proceso de paralelismo:
- Paralelismo automático, aquí el compilador del código fuente, es capaz de identificar por sí solo el código que puede ejecutarse en forma paralela.
  - Paralelismo natural, el usuario a través de comandos, decide que parte del código ejecutar de forma paralela.

En relación a las ventajas, se plantean las siguientes:

- Mejorar el tiempo de respuesta en situaciones que el proceso secuencial tiene un alto valor de demora.
- Ejecución de código de forma ágil.
- Trabajar de forma simultánea con tareas independientes.

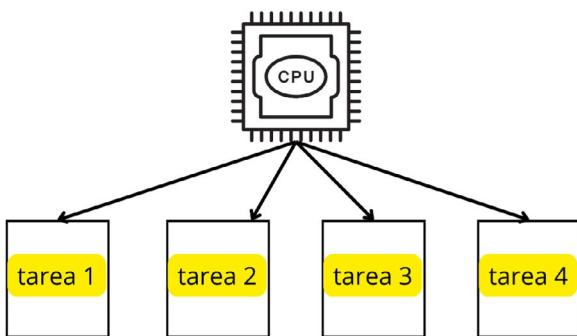
Por otro lado, algunas desventajas, se las lista a continuación:

- Existe mayor complejidad para generar las soluciones.
- Mayor consumo de recursos.
- El manejo de la sincronización puede ser un problema si no se lo realiza de forma adecuada.

Es importante diferenciar dos términos muy usados al momento de estudiar y practicar en la programación con hilos. Cuando se habla de paralelo y recurrente, se tiende a confundir los conceptos, pero existe la siguiente diferencia:

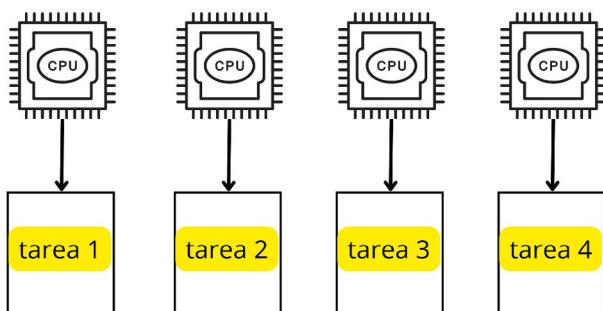
- Concurrente, es la capacidad de ejecutar un conjunto de tareas al mismo tiempo desde el mismo procesador; ver figura 35.
- Paralelo, las tareas se ejecutan al mismo tiempo en cada procesador de forma independiente; ver figura 36.

**Figura 34.**  
*Proceso Concurrente*



Nota. Tomado de Moreno (2019).

**Figura 35.**  
*Proceso Paralelo*



Nota. Tomado de Moreno (2019).

### 3.3.2. Gestión y administración de hilos

Al considerar un proceso como cualquier programa en ejecución; un hilo es conceptualizado como un conjunto de instrucciones que están dentro de un determinado proceso de forma independiente. Indicar también que, un proceso puede estar constituido por uno o más hilos, dando lugar a un proceso multihilo.

El autor del recurso educativo, menciona las siguientes ideas, en relación a algunos parámetros, desde el punto de vista de proceso e hilo, ver tabla 10.

**Tabla 10.**

Procesos e hilos.

| Parámetro             | Proceso                                                                          | Hilo                                                                    |
|-----------------------|----------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| <b>Independencia.</b> | Los procesos son independientes.                                                 | Los hilos no son independientes.                                        |
| <b>Creación.</b>      | Se consume más recursos para la creación de nuevos procesos.                     | No existe excesivo consumo de recursos para la creación de hilos.       |
| <b>Estados.</b>       | Posibles estados:<br>• Listo.<br>• Bloqueado.<br>• En ejecución.<br>• Terminado. |                                                                         |
| <b>Memoria.</b>       | No se comparte memoria entre los procesos.                                       | Existe memoria compartida al ejecutar varios hilos en un mismo proceso. |

Nota. Elizalde R, 2023



Semana 12

### 3.4. Programación multihilo

#### 3.4.1. Ejemplos de aplicaciones mediante multihilo

En el presente apartado se va a trabajar con Python como lenguaje de programación para la ejecución de ejemplos usando hilos. Para el estudio y ejecución de los ejemplos, se solicita instalar Python en los entornos locales. La librería para trabajar con hilos en lenguaje de programación, se denomina: [threading](#).



Recalcarse que, la librería se la debe importar para la exemplificación en cada script generado. Importante mencionar, no hay que instalar la librería, pues, viene incluida en la librería estándar del lenguaje de programación Python.

Algunas situaciones iniciales para el trabajo con la librería *threading*:

- Crear un hilo, ejemplo:
  - `hilo1 = threading.Thread(target=[nombre de la función])`
- Iniciar un hilo, a través del método start:
  - `hilo.start()`
- Obtener el nombre del hilo que está en ejecución:
  - `threading.current_thread().getName()`
- Obtener el identificador del hilo que está en ejecución:
  - `threading.current_thread().ident`

### Ejemplo 1:

- Generar un archivo en Python que use la librería *threading* para crear 4 hilos y exista un tiempo de espera 10 segundos para la finalización.

La solución a la problemática es la siguiente:

```
import time
se importa la librería
import threading

se genera la función
def worker(nombre):
 print("Iniciando %s %s" % (threading.current_thread().getName(),
 nombre))
 # tiempo de espera
 time.sleep(10)
 print("Finalizando %s" % (threading.current_thread().getName()))

lista para nombrar los hilos
lista = ["Ecuador", "Perú", "Colombia", "Venezuela"]
for i in lista:
 # se crea el hilo
 # el nombre del hilo, será el valor de i en la iteración
 # con target se llama a la función (def worker)
 # args, tiene los argumentos de la función; en el ejemplo
 # se envía el valor de i
 t = threading.Thread(name="paises", target=worker, args=(i,))
 # se inicia el hilo
 t.start()
```

## Ejemplo 2:

- Generar un archivo en Python que use la librería *threading* para generar 2 hilos y exista un tiempo de espera 10 segundos para la finalización. En cada llamada a la función se debe usar la librería *subprocess* para abrir un navegador (para el ejemplo se usa [chromium-browser](#)). La url que debe abrir el comando mediante el navegador, llegará como parámetro. La solución a la problemática es la siguiente:

```
import time
import threading
librería de python que permite ejecutar comandos
import subprocess

def worker(url):
 print("Iniciando %s %s" % (threading.current_thread().getName(), url))
 # se arma el comando
 # para el ejemplo se usa chromium-browser
 # comando debe variar según el sistema operativo que se use
 comando = "chromium-browser %s" % (url)
 # se ejecuta el comando, a través Popen
 # el comando debe levantar un navegador con la url, dada como
 # argumento.
 subprocess.Popen(comando, shell=True)
 time.sleep(10)
 print("Finalizando %s" % (threading.current_thread().getName()))

Se crea los hilos
args, tiene el argumento, será la url para abrir con el comando
en la función
hilo1 = threading.Thread(name='navegando', target=worker,
args=("www.youtube.com",))
hilo2 = threading.Thread(name='navegando', target=worker,
args=("www.utpl.edu.ec",))

inicio de cada hilo
hilo1.start()
hilo2.start()
```

## Ejemplo 3:

- Dado un archivo con la siguiente estructura, ver figura 36.

## Figura 36.

Estructura del archivo para trabajar con hilos

```
1|python con base de datos
2|python con nosql
3|sqlalchemy y mysql
4|postgres y python
5|git y github
6|python y Django
7|metodologias de desarrollo
8|Ecuador turistico
9|Loja capital cultural del Ecuador
10|provincias del Ecuador
```

Nota. Elizalde R, 2023

- Generar un archivo en Python que permita, mediante hilos, abrir  $n$  número de navegadores con la url de Youtube. Además, la búsqueda para el inicio de información en Youtube, está dada en cada línea del archivo.

```
import time
import csv
import threading
librería de python que permite ejecutar comandos
import subprocess
```

```

def obtener_data():
 lista = []
 # del archivo se necesita obtener la cadena de búsqueda para cuando
 # se busque en youtube
 # ejemplo:
 # De la linea: '7|metodologias de desarrollo', se necesita solo la
 # cadena metodologias de desarrollo
 #
 with open("datos/informacion.csv") as archivo:
 lineas = csv.reader(archivo, quotechar="|")
 for row in lineas:
 # row es una lista
 # con esta estructura ['7|metodologías de desarrollo']
 # row[0] tendrá: '7|metodologías de desarrollo'
 # row[0].split("|"), será: ["7", "metodologías de desarrollo"]
 # row[0].split("|")[1], será: "metodologías de desarrollo"
 cadena = row[0].split("|")[1]
 # A la cadena se le reemplaza los espacios vacios con el
 simbolo
 # +, para que quede listo para la búsqueda
 cadena = cadena.replace(" ", "+")
 lista.append(cadena)
 # se retorna la lista con la información que se necesita
 return lista

def worker(url, cadena_busqueda):
 print("Iniciando %s %s" % (threading.current_thread().getName(), url))
 # se arma el comando
 # para el ejemplo se usa google-chrome
 # comando debe variar según el sistema operativo que se use
 # se debe formar la siguiente cadena de búsqueda
 # https://www.youtube.com/results?search_query=loja+ciudad+cultural
 comando = "google-chrome %s/results?search_query=%s" % (url,
 cadena_busqueda)
 # se ejecuta el comando, a través Popen
 # el comando debe levantar un navegador con la url y la cadena de
 búsqueda,
 # dada como argumento.
 subprocess.Popen(comando, shell=True)
 time.sleep(10)
 print("Finalizando %s" % (threading.current_thread().getName()))

inicio del ciclo para crear los hilos
for c in obtener_data():
 # Se crea los hilos
 # args, tiene el argumento, será la url para abrir con el comando y
 # la cadena de búsqueda
 # en la función
 hilol = threading.Thread(name='navegando2',
 target=worker,
 args=("www.youtube.com", c))
 hilol.start()

```



Estimado estudiante, los ejemplos completos se lo puede [descargar del repositorio de Github](#) y realizar los cambios para su estudio.



## Actividades de aprendizaje recomendadas

- **Actividad 1:** para consolidar los conceptos relacionados al uso de hilos en aplicaciones en Python, se solicita la revisión del siguiente recurso denominado [Threading | Threading and Multi Processing in Python Part 1](#). En el mismo se explica el concepto de hilos, cuando se debería usar aplicaciones con hilos, finalmente se realizan algunos ejemplos, que se sugiere, recrearlos en sus entornos locales.
- **Actividad 2:** resolver la autoevaluación 5; en la actividad se presentan un conjunto de preguntas basadas en las temáticas descritas en la unidad 3: conceptos avanzados de programación, sección: 3.2. Programación multihilo. En la parte introductoria de la actividad se listan los contenidos de la guía didáctica, que se debe revisar para contestar las interrogantes. Recuerde que la autoevaluación le permitirá identificar el nivel de aprendizaje y comprensión alcanzado; así mismo, determinar los ítems que se debe volver a revisar para lograr los resultados de aprendizaje propuestos. Adicionalmente, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado.

Finalmente, después de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección solucionario de la guía didáctica.



## Autoevaluación 5

Estimado estudiante, es momento de medir el entendimiento de algunos conceptos estudiados en la unidad 3: conceptos avanzados de programación en el punto 3.2. Programación multihilo. Antes de realizar la autoevaluación, se requiere las siguientes acciones:

- Revisar todo el contenido de las temáticas indicadas.
- Leer los recursos educativos recomendados.
- Hacer organizadores gráficos para la ordenar la información para su estudio.

La autoevaluación le permitirá reafirmar los conceptos estudiados. Los enunciados tienen una sola opción correcta, lea de forma atenta cada respuesta y seleccione la opción que usted considere correcta.

### 1. ¿Cuál es el objetivo de la programación en paralelo?

- a. El principal objetivo de la programación en paralelo es la de poder resolver los tiempos elevados que puedan existir al momento de ejecutar un proceso.
- b. El principal objetivo de la programación en paralelo es la de poder incrementar los tiempos elevados que puedan existir al momento de ejecutar un proceso.
- c. El principal objetivo de la programación en paralelo es la de poder desconocer los tiempos elevados que puedan existir al momento de ejecutar un proceso.

- 2. Seleccione los pasos recomendados para poder resolver un problema de forma paralela.**
- Intenta la dividir el problema en subproblemas independientes / generar una solución para cada subproblema.
  - Generar una solución para cada subproblema / mantener el control de los subproblemas/ ejecutar cada subproblema en un procesador.
  - Intenta la dividir el problema en subproblemas independientes / generar una solución para cada subproblema / ejecutar cada subproblema en un procesador / Mantener el control de los subproblemas.
- 3. ¿Cuál es la subdivisión correcta para el tipo de paralelismo en función del método?**
- A nivel de bit / a nivel de datos / a nivel de tareas.
  - A nivel de bit / a nivel de instrucción / a nivel de datos.
  - A nivel de bit / a nivel de instrucción / a nivel de datos / a nivel de tareas.
- 4. Identifique una de las desventajas de la programación en paralelo.**
- Mayor consumo de recursos.
  - Ejecución de código de forma ágil.
  - Trabajar de forma simultánea con tareas independientes.
- 5. ¿Cuál de los siguientes es el concepto más apropiado para un hilo?**
- Un hilo es conceptualizado como un conjunto de instrucciones que están dentro de un determinado proceso de forma independiente.
  - Un hilo es conceptualizado como un conjunto de instrucciones que están dentro de un determinado proceso de forma dependiente.
  - Un hilo es conceptualizado como un conjunto de instrucciones que están fuera de un determinado proceso de forma independiente.

**6. ¿Cómo se denomina la librería para trabajar en Python con hilos?**

- a. threding.
- b. treading.
- c. threading.

**7. ¿Cuál es la forma correcta para crear un hilo en Python?**

a.

```
import threading.
hilo1 = threading.Thread()
```

b.

```
import threading.
hilo1 = threading.Thread(target=[nombre de la función]).
```

c.

```
import threading.
hilo1 = Thread(target=[nombre de la función]).
```

**8. ¿Cuál es la sentencia que permite obtener el nombre del hilo que está en ejecución?**

- a. threading.current\_thread().getName()
- b. threading.current\_thread().Name()
- c. threading.current\_thread().setName()

**9. ¿Cuál es la sentencia que permite obtener el identificador del hilo que está en ejecución?**

- a. threading.current().ident.
- b. threading.currentthread().ident.
- c. threading.current\_thread().ident.

**10. ¿Cuál es la forma correcta de instalar la librería threading en lenguaje Python?**

- a. pip install threading.
- b. easy\_install threading.
- c. La librería viene por defecto, no se necesita instalar nada.

[Ir a solucionario](#)



### 3.5. Programación de interfaces e interactiva

#### 3.5.1. Conceptos



Para la presente sección solicitamos realizar una lectura comprensiva de lo contenido en el recurso denominado [Curso de creación de GUIs con Qt. Capítulo 01: Qt, versiones y bindings.](#)

El recurso menciona a Qt como un *framework* multiplataforma para crear aplicaciones gráficas (GUIs -Interfaces Gráficas de Usuario). La idea es generar interfaces gráficas de manera ágil, en cualquier sistema operativo. La plataforma está escrita en C++, para usar desde lenguaje Python se debe manejar las librerías que permiten la comunicación. Existen algunas versiones de Qt que se utilizan en algunos proyectos: Qt4 y Qt5.

En el recurso se menciona las algunas librerías. En la tabla 11, se listan las librerías y se muestra la relación con las versiones de Qt.

**Tabla 11.**

*Librerías y su relación con plataforma Qt*

| Librería | Versión de Python   | Versión de Qt | Licencia                  |
|----------|---------------------|---------------|---------------------------|
| PyQt4.   | Python 3.5 o mayor. | 4 y 5.        | GPLv3<br>Comercial.       |
| PySide.  | Python 3.4 o menor. | 4.8.          | GPL.                      |
| PyQt5.   | Python 3.5 o mayor. | 5.6.          | GPLv3<br>Comercial.       |
| PySide2. | Python 3.5 o mayor. | 5.6.          | GPL, LGPL o<br>comercial. |

Nota. Elizalde R, 2023

Para el diseño de las interfaces se usa Qt Designer, que permite generar interfaces de forma rápida. Se recomienda realizar el proceso de

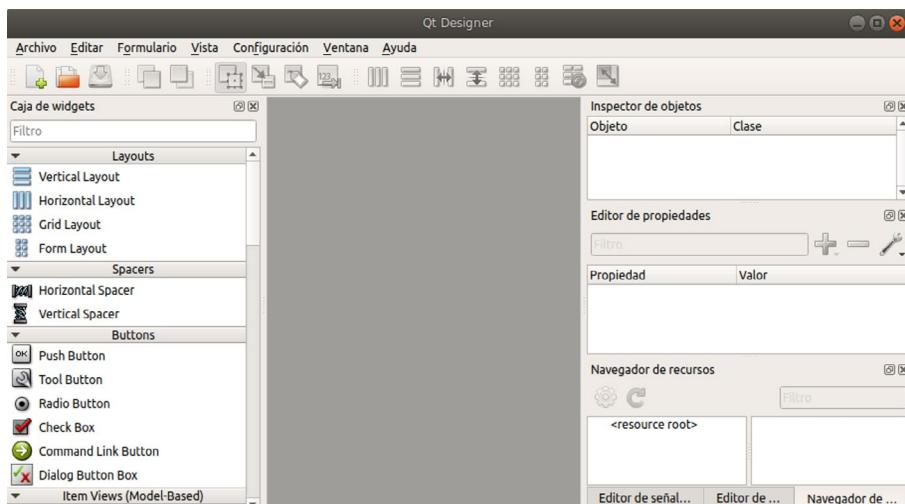
instalación en los entornos locales. Usar las indicaciones dadas en la página oficial de [Qt Designer](#).

Las principales características de Qt Designer son:

- Permite generar una interface mediante múltiples componentes, como: Layouts, spacer, buttons, ítem views, ítem widgets, etc.
- Por cada componente se puede personalizar la información, principalmente: nombre, tamaño, tipo de letra, ubicación, etc. Ver figura 37.

**Figura 37.**

Vista principal de Qt Designer



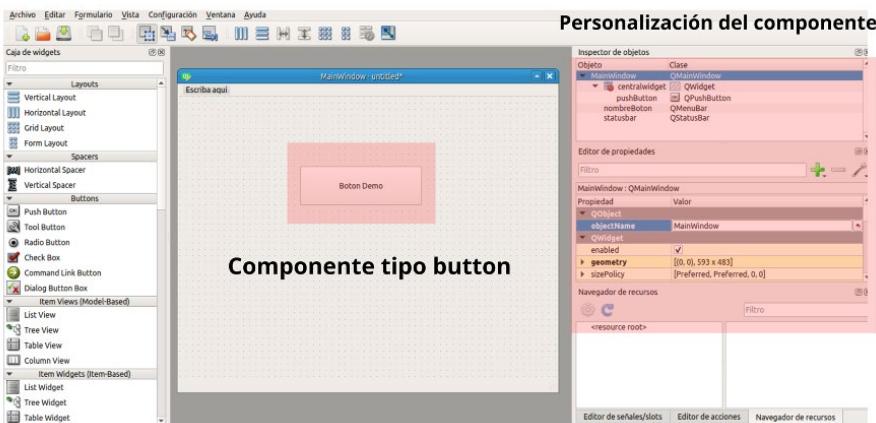
Nota. Elizalde R, 2023

Importante mencionar que la extensión con la que se guardan los archivos de Qt Designer es **.ui**; se lo debe recordar para la siguiente sección, cuando se use el proceso de transformación a código Python.

En la figura 37, se trabaja con un componente de tipo *button* y en la sección señalada se puede personalizar información del componente. Dicha información se usará dese lenguaje Python en lo posterior, para agregar acciones.

**Figura 38.**

Personalización del componente tipo Button en Qt Designer



Nota. Elizalde R, 2023

Es importante que se pueda experimentar con la generación de ejemplos en su entorno local. Le ánimo a crear la pantalla de la figura 39 a través de Qt Designer.

**Figura 39.**

Ejemplo para recrear con Qt Designer



Nota. Elizalde R, 2023

### 3.5.2. Herramientas

En lenguaje Python existen algunas librerías que permiten trabajar con interfaces gráficas. Entre las más usadas se tiene las siguientes:

- Tkinter.
- PyQt5.

En el estudio de las siguientes secciones se usará la librería PyQt5 para el desarrollo de ejercicios y ejemplos en la construcción de interfaces. La librería PyQt5 permite construir interfaces a través de la plataforma Qt, de forma rápida y sencilla. Es posible crear las interfaces a través de Qt Designer y luego mediante la librería PyQt5, personalizar o agregar acciones a cada componente que se requiera en código Python.

El proceso de instalación requiere el uso del instalador de paquetes de Python, denominado pip; los comandos de instalación son:

- *pip install PyQt5.*
- *pip install pyqt5-tools.*

Es importante verificar que esté instalada la librería, ver figura 40.

**Figura 40.**

*Instalación y verificación de PyQt5-tools*

**pip install PyQt5**

```
Collecting PyQt5
 Downloading PyQt5-5.15.9-cp37-abi3-manylinux2_17_x86_64.whl (8.4 MB)
 0% |██████████| 0.00 MB 0.00 MB/s eta 0:00:00
Collecting PyQt5-sip<13,>=12.11
 Using cached PyQt5_sip-12.11.1-cp39-cp39-manylinux_2_3_x86_64.manylinux1_x86_64.whl (1.3 MB)
Collecting PyQt5-Qt5<5.15.2
 Using cached PyQt5_Qt5-5.15.2-py3-none-manylinux2014_x86_64.whl (59.9 kB)
Installing collected packages: PyQt5, PyQt5-sip, PyQt5-Qt5
Successfully installed PyQt5-5.15.9 PyQt5-Qt5-5.15.2 PyQt5-sip-12.11.1
```

**pip install pyqt5-tools**

```
Collecting Pyqt5-tools
 Using cached pyqt5_tools-5.15.4.3.2-py3-none-any.whl (29 kB)
Collecting pyqt=5.15.4
 Using cached Pyqt5-5.15.4-cp36-cp37-cp38-cp39-abi3-manylinux2014_x86_64.whl (8.3 kB)
Collecting click
 Using cached click-8.1.3-py3-none-any.whl (56 kB)
Collecting python-dotenv
 Using cached python-dotenv-0.21.1-py3-none-any.whl (19 kB)
Collecting pyqt5-plugins<5.15.4>
 Using cached pyqt5_plugins-5.15.4.2-py3-manylinux2014_x86_64.whl (69 kB)
```

**Verificar la instalación**

```
In [1]: import PyQt5
In [2]:
```

Nota. Elizalde R, 2023



**Semana 14**

## 3.6. Programación de interfaces e Interactiva

### 3.6.1. Construcción de Interfaces

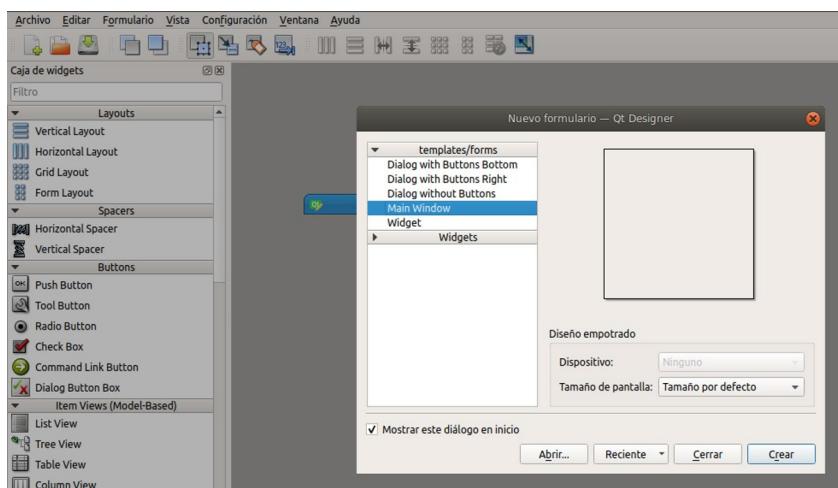
Para la construcción de interfaces a través de la librería de Python PyQt5 se pueden seguir varios procesos. En este apartado se genera un

ejemplo, para la ejecución del ejercicio en su entorno local, se necesita los siguientes elementos:

- Instalar Python y las librerías PyQt5 y pyqt5-tools; como se lo describió en la sección anterior. Además de la librería para el diseño gráfico de las interfaces Qt Designer.
- Se ejecuta la interfaz Qt Designer.
- Se agregan los elementos gráficos necesarios. Para el ejemplo se crea una interfaz usando el template **Main Windows**, ver figura 41.

**Figura 41.**

Creación de la interface a través de un template en Qt

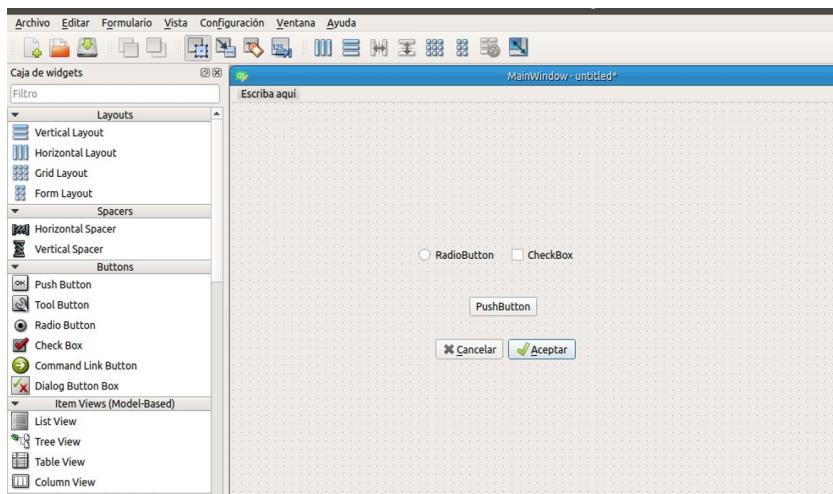


Nota. Elizalde R, 2023

- Se agregan los componentes que se necesiten y se lo ubica en la parte del *layout* que se requiera.

**Figura 42.**

Agregar componentes de la interface



Nota. Elizalde R, 2023

- Finalmente se guarda el diseño generado con la extensión .ui.
- Recalcar que lo generado solo es una interface con componentes que pueden estar personalizados; pero no tiene una lógica.



Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.

### 3.6.2. Uso de widgets

Luego del uso de la plataforma Qt Designer para la creación de la interfaz, a continuación se explica el proceso para pasar/convertir un archivo .ui (Qt) a un archivo de .py (Python) bajo las directrices propias de PyQt5.

En el directorio donde se guardó el archivo generado por Qt, se debe ejecutar el siguiente comando, a través de [pyuic5](#), desde el CMD/Terminal/Consola del sistema operativo:

- `pyuic5 -x ejemplo01.ui -o ejemplo01.py`.

Donde, ejemplo01.ui es el archivo creado a través de Qt; y, la salida del comando permitirá la creación de un nuevo archivo, con extensión .py.

En este punto, indicar que es posible no usar el programa Qt, y se puede generar el código con la librería PyQt5, línea por línea. Se va explicar el código generado con el comando de conversión pyuic5; donde se detalla algunos *widgets* propios de la librería.

En PyQt5 se pueden usar algunos *widgets*, a través de [QtWidgets](#). Los mismos permiten crear elementos propios de la interface para aplicaciones de tipo escritorio.

En el siguiente código:

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
 def setupUi(self, MainWindow):
 MainWindow.setObjectName("MainWindow")
 MainWindow.resize(574, 466)
 self.centralwidget = QtWidgets.QWidget(MainWindow)
 self.centralwidget.setObjectName("centralwidget")
 self.pushButton = QtWidgets.QPushButton(self.centralwidget)
 self.pushButton.setGeometry(QtCore.QRect(230, 200, 80, 23))
 self.pushButton.setObjectName("pushButton")
 MainWindow.setCentralWidget(self.centralwidget)
 self.menubar = QtWidgets.QMenuBar(MainWindow)
 self.menubar.setGeometry(QtCore.QRect(0, 0, 574, 20))
 self.menubar.setObjectName("menubar")
 MainWindow.setMenuBar(self.menubar)
 self.statusbar = QtWidgets.QStatusBar(MainWindow)
 self.statusbar.setObjectName("statusbar")
 MainWindow.setStatusBar(self.statusbar)

 self.retranslateUi(MainWindow)
 QtCore.QMetaObject.connectSlotsByName(MainWindow)

 def retranslateUi(self, MainWindow):
 _translate = QtCore.QCoreApplication.translate
```

```
MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
self.pushButton.setText(_translate("MainWindow", "Saludo"))

if __name__ == "__main__":
 import sys
 app = QtWidgets.QApplication(sys.argv)
 MainWindow = QtWidgets.QMainWindow()
 ui = Ui_MainWindow()
 ui.setupUi(MainWindow)
 MainWindow.show()
 sys.exit(app.exec_())
```

Se usan algunos *widgets*:

- QApplication, que permite administrar el flujo de información de la aplicación. Es necesario crear un objeto de tipo *widget*; es posible que acepte argumentos externos, por ejemplo, desde la consola.
- QMainWindow, es un *widget* que proporciona la pantalla principal de la aplicación. Cuando se usa Qt para la generación de la interfaz, existen algunas opciones como templates para la creación de pantalla principal: *MainWindow*, *Dialog without Buttons*, *Dialog with Buttons*, principalmente. Se debe crear un objeto del tipo QMainWindow y luego usar el método show, para que inicie la aplicación.



Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.

En el recurso educativo denominado [PyQt5 Widgets](#) se hace la explicación de un conjunto de *widgets* que se pueden utilizar en la generación de aplicaciones con PyQt5: *QCheckbox*, *QComboBox*, *QLabel*, *QPushButton*, *QradioButton*, etc. Usted puede revisar la información y experimentar con ellos. A continuación, algunos ejemplos tomados del recurso educativo.

- En el siguiente ejemplo se hace uso del Widget QLabel.

```
import sys
from PyQt5.QtWidgets import (
 QMainWindow, QApplication,
 QLabel, QCheckBox, QComboBox, QListWidget, QLineEdit,
 QLineEdit, QSpinBox, QDoubleSpinBox, QSlider
)
from PyQt5.QtCore import Qt

class MainWindow(QMainWindow):

 def __init__(self):
 super(MainWindow, self).__init__()

 self.setWindowTitle("My App")
 widget = QLabel("Hello") # Se crea el Widget
 font = widget.font() # se crea una fuente (tipo de letra)
 font.setPointSize(30)
 widget.setFont(font) # se agrega al Widget el tipo de letra
 widget.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter) #

 self.setCentralWidget(widget)

if __name__ == "__main__":
 import sys
 app = QApplication(sys.argv)
 mainWindow = MainWindow()
 mainWindow.show()
 sys.exit(app.exec_())
```

En el siguiente ejemplo se hace uso del *widget* QListWidget.

```
import sys
from PyQt5.QtWidgets import (
 QMainWindow, QApplication,
 QLabel, QCheckBox, QComboBox, QListWidget, QLineEdit,
 QLineEdit, QSpinBox, QDoubleSpinBox, QSlider
)
from PyQt5.QtCore import Qt

class MainWindow(QMainWindow):

 def __init__(self):
 super(MainWindow, self).__init__()

 self.setWindowTitle("My App")
 widget2 = QListWidget() # se crea el objeto del Widget
 widget2.addItems(["Loja", "Quito", "Guayaquil"]) # Se agrega datos al
 Widget

 widget2.currentItemChanged.connect(self.index_changed)
 widget2.currentTextChanged.connect(self.text_changed)

 self.setCentralWidget(widget2)

 def index_changed(self, i):
 print(i.text())

 def text_changed(self, s):
 print(s)

if __name__ == "__main__":
 import sys
 app = QApplication(sys.argv)
 mainWindow = MainWindow()
 mainWindow.show()
 sys.exit(app.exec_())
```



Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.

Le invito a profundizar sus conocimientos sobre el uso de eventos.

### 3.6.3. Uso de eventos



Es necesario realizar una lectura comprensiva de lo indicado en el recurso [PyQt5 Signals, Slots & Events](#); el mismo explica la forma como se debe manejar los eventos (signals) en la librerías PyQt5.

Se resalta que un signal es la notificación generada por un *widget* al momento que sucede alguna acción, por ejemplo, al presionar un botón, al cambiar un texto, etc. Los eventos por lo general son inicializados por los usuarios, pero no se descarta que exista otras formas de ingreso de datos.

**Ejemplo 1:** en el siguiente ejemplo, la acción click a un botón, genera una llamada a un proceso (función en Python), que realiza un requerimiento; para el ejemplo, la acción es muy simple, imprime algo en la consola. Ver figura 43.

```
import sys
from PyQt5.QtWidgets import (
 QMainWindow, QApplication, QPushButton
)
from PyQt5.QtCore import Qt

class MainWindow(QMainWindow):

 def __init__(self):
 super(MainWindow, self).__init__()

 self.setWindowTitle("Probando los signals")
 button = QPushButton("Acepta")
 button.setCheckable(True)
 # se llama a la acción (proceso conocido como slot)
 # la acción es accion_en_boton
 button.clicked.connect(self.accion_en_boton)

 # se ubica el botón en la pantalla principal
 self.setCentralWidget(button)

 def accion_en_boton(self):
 print("Hemos dado click en un botón")

if __name__ == "__main__":
 import sys
 app = QApplication(sys.argv)
 mainWindow = MainWindow()
 mainWindow.show()
 sys.exit(app.exec_())
```

## Figura 43.

Uso de Signal en una aplicación de PyQt5



Nota. Elizalde R, 2023

**Ejemplo 2:** en el siguiente ejemplo se realiza dos llamadas a procesos o funciones con la misma acción de un botón de tipo **clicked**. Las acciones son paralelas y tienen procesos (lógica) diferentes; el segundo proceso recibe un argumento propio de los botones, de tipo booleano. Ver figura 44.

```
import sys
from PyQt5.QtWidgets import (
 QMainWindow, QApplication, QPushButton
)
from PyQt5.QtCore import Qt

class MainWindow(QMainWindow):

 def __init__(self):
 super(MainWindow, self).__init__()

 self.setWindowTitle("Probando los signals")
 button = QPushButton("Acepta")
 button.setCheckable(True)
 # se llama a la acción (proceso conocido como slot)
 # la acción es accion_en_boton
 button.clicked.connect(self.accion_en_boton)
 button.clicked.connect(self.accion_en_boton_parametro)

 # se ubica el botón en la pantalla principal
 self.setCentralWidget(button)

 def accion_en_boton(self):
 print("Hemos dado click en un botón")

 def accion_en_boton_parametro(self, check):
 print("Chekeado en %s" % (check))

if __name__ == "__main__":
 import sys
 app = QApplication(sys.argv)
 mainWindow = MainWindow()
 mainWindow.show()
 sys.exit(app.exec_())
```

## Figura 44.

Uso de dos signal en una aplicación PyQt5

```
Hemos dado click en un botón
Chekeado en True
Hemos dado click en un botón
Chekeado en False
Hemos dado click en un botón
Chekeado en True
Hemos dado click en un botón
Chekeado en False
Hemos dado click en un botón
Chekeado en True
Hemos dado click en un botón
Chekeado en False
Hemos dado click en un botón
Chekeado en True
```



Nota. Elizalde R, 2023



Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.

### 3.6.4. Ejemplos

**Ejemplo 1:** generar una solución usando la librería PyQt5 que permita ingresar información de empleados a través de un formulario; y que permita visualizar los registros almacenados en una base de datos.

Los pasos a seguir son los siguientes:

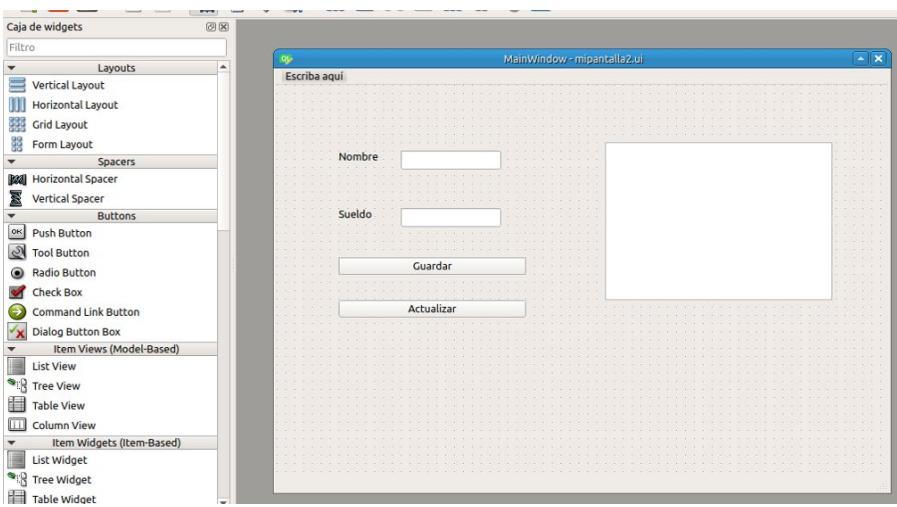
#### Paso 1:

- Instalar las librerías necesarias, de acuerdo a lo estudiado.
- La base de datos que se usa es SQLite.

#### Paso 2:

- Generar el diseño de la interface, a través de Qt. figura

**Figura 45.**  
*Diseño de la interface, a través de Qt 46*



Nota. Elizalde R, 2023

### Paso 3:

- Transformar el archivo generado en Qt a lenguaje Python.
  - Usar la siguiente sentencia:

*pyuic5 -x mipantalla.ui -o mipantalla.py.*

El archivo en Python queda de la siguiente forma:

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
 def setupUi(self, MainWindow):
 MainWindow.setObjectName("MainWindow")
 MainWindow.resize(689, 514)
 self.centralwidget = QtWidgets.QWidget(MainWindow)
 self.centralwidget.setObjectName("centralwidget")
 self.nombre = QtWidgets.QLineEdit(self.centralwidget)
 self.nombre.setGeometry(QtCore.QRect(140, 80, 113, 23))
 self.nombre.setObjectName("nombre")
 self.label = QtWidgets.QLabel(self.centralwidget)
```

```

 self.label.setGeometry(QtCore.QRect(70, 80, 54, 15))
 self.label.setObjectName("label")
 self.sueldo = QtWidgets.QLineEdit(self.centralwidget)
 self.sueldo.setGeometry(QtCore.QRect(140, 150, 113, 23))
 self.sueldo.setText("")
 self.sueldo.setObjectName("sueldo")
 self.label_2 = QtWidgets.QLabel(self.centralwidget)
 self.label_2.setGeometry(QtCore.QRect(70, 150, 54, 15))
 self.label_2.setObjectName("label_2")
 self.guardar = QtWidgets.QPushButton(self.centralwidget)
 self.guardar.setGeometry(QtCore.QRect(70, 210, 211, 21))
 self.guardar.setObjectName("guardar")
 self.actualizar = QtWidgets.QPushButton(self.centralwidget)
 self.actualizar.setGeometry(QtCore.QRect(70, 262, 211, 21))
 self.actualizar.setObjectName("actualizar")
 self.listaEmpleados = QtWidgets.QTableWidget(self.centralwidget)
 self.listaEmpleados.setGeometry(QtCore.QRect(370, 70, 256, 192))
 self.listaEmpleados.setObjectName("listaEmpleados")
 self.listaEmpleados.setColumnCount(0)
 self.listaEmpleados.setRowCount(0)
 MainWindow.setCentralWidget(self.centralwidget)
 self.menuubar = QtWidgets.QMenuBar(MainWindow)
 self.menuubar.setGeometry(QtCore.QRect(0, 0, 689, 20))
 self.menuubar.setObjectName("menuubar")
 MainWindow.setMenuBar(self.menuubar)
 self.statusbar = QtWidgets.QStatusBar(MainWindow)
 self.statusbar.setObjectName("statusbar")
 MainWindow.setStatusBar(self.statusbar)

 self.retranslateUi(MainWindow)
 QtCore.QMetaObject.connectSlotsByName(MainWindow)

 def retranslateUi(self, MainWindow):
 _translate = QtCore.QCoreApplication.translate
 MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
 self.label.setText(_translate("MainWindow", "Nombre"))
 self.label_2.setText(_translate("MainWindow", "Sueldo"))
 self.guardar.setText(_translate("MainWindow", "Guardar"))
 self.actualizar.setText(_translate("MainWindow", "Actualizar"))

 if __name__ == "__main__":
 import sys
 app = QtWidgets.QApplication(sys.argv)
 MainWindow = QtWidgets.QMainWindow()
 ui = Ui_MainWindow()
 ui.setupUi(MainWindow)
 MainWindow.show()
 sys.exit(app.exec_())

```

## Paso 4

- Agregar las funciones necesarias para los botones generados en la interface.
- Llamar a las funciones creadas en el método `__init__py` de la clase.

```
guardar
self.crear_base()
self.guardar.clicked.connect(self.guardar_informacion)
self.actualizar.clicked.connect(self.obtener_informacion)
```

- Generar la función que permitirá crear la entidad de la base de datos.

```
def crear_base(self):
 cursor = conn.cursor()
 cadena_sql = 'CREATE TABLE Empleado (nombre TEXT, sueldo INTEGER)'
 cursor.execute(cadena_sql)
 cursor.close()
```

- Generar la función que ayuda en el proceso de guardar información.

```
def guardar_informacion(self):
 cursor = conn.cursor()
 nombre = str(self.nombre.text())
 sueldo = int(self.sueldo.text())
 cadena_sql = """INSERT INTO Empleado (nombre, sueldo) VALUES ('%s',
%d);""" % \
 (nombre, sueldo)
 # ejecutar el SQL
 cursor.execute(cadena_sql)
 # confirmar los cambios
 conn.commit()
 cursor.close()
```

- Crear la función para listar la información de la base de datos.

```

def obtener_informacion(self):
 cursor = conn.cursor()
 cadena_consulta_sql = "SELECT * from Empleado"
 cursor.execute(cadena_consulta_sql)
 informacion = cursor.fetchall()
 database_table_column_count = 2
 self.listaEmpleados.setColumnCount(database_table_column_count)
 numero_filas = len(informacion)
 self.listaEmpleados.setRowCount(numero_filas)
 for j in range(numero_filas):
 valor = informacion[j]
 for i in range(0, len(valor)):
 elemento = valor[i]
 elemento = str(elemento)
 nuevo_registro = QtWidgets.QTableWidgetItem(elemento)
 self.listaEmpleados.setItem(j, i, nuevo_registro)

```

## Paso 5

- Ejecutar desde consola el archivo de Python y verificar el funcionamiento de la interfaz generada, ver figura 47.

**Figura 46.**

Proceso para guarda y visualizar información en PyQt5 / SqLite

**1** Interface



**2** Agregar información



Visualizar información

**3**



Nota. Elizalde R, 2023



Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.



## Actividades de aprendizaje recomendadas

- **Actividad 1:** revisar el recurso denominado [Calculadora Básica con Interfaz Gráfica de Usuario \(GUI\) con PyQt5](#), en el mismo se hace la explicación de creación de un ejemplo bajo la librería PyQt5. Se parte de bosquejo de pantalla, luego se pasa a generar la interface y finalmente se genera la lógica que permite obtener administrar la información que el usuario ingresa por teclado. Es muy importante que se pueda recrear el ejercicio en sus entornos locales, para reafirmar los conceptos y ejemplos revisados en la guía didáctica.
- **Actividad 2:** resolver la autoevaluación 6; en la actividad se presentan un conjunto de preguntas basadas en las temáticas descritas en la unidad 3: conceptos avanzados de programación, sección: 3.3. Programación de interfaces e interactiva. En la parte introductoria de la actividad se listan los contenidos de la guía didáctica, que se debe revisar para contestar las interrogantes.

Recuerde que la autoevaluación le permitirá identificar el nivel de aprendizaje y comprensión alcanzado; así mismo, determinar los ítems que se debe volver a revisar para lograr los resultados de aprendizaje propuestos. Adicionalmente, usted podrá plantearse interrogantes o ejercicios similares para reafirmar lo analizado y estudiado. Finalmente, después de realizar la autoevaluación, puede revisar las respuestas correctas de las preguntas en la sección solucionario de la guía didáctica.



## Autoevaluación 6

Es momento de medir el entendimiento de algunos conceptos estudiados en la unidad 3: conceptos avanzados de programación en el punto 3.3. Programación de interfaces e interactiva. Se sugiere revisar realizar las siguientes acciones, previo al inicio de la autoevaluación.

- Revisar todo el contenido de las temáticas indicadas.
- Leer los recursos educativos recomendados.
- Hacer organizadores gráficos para ordenar la información para su estudio.

La autoevaluación le permitirá reafirmar los conceptos estudiados. Los enunciados tienen una sola opción correcta, lea de forma atenta cada respuesta y seleccione la opción que usted considere correcta.

**1. De las siguientes descripciones ¿Cuál es la mejor forma para conceptualizar a la plataforma Qt?**

- a. Qt como es un *framework* multiplataforma para crea aplicaciones gráficas (GUIs -Interfaces Gráficas de Usuario).
- b. Qt como es un *framework* de Linux exclusivo para crea aplicaciones gráficas (GUIs -Interfaces Gráficas de Usuario).
- c. Qt como es un *framework* de Windows exclusivo para crea aplicaciones gráficas (GUIs -Interfaces Gráficas de Usuario).

**2. ¿En qué lenguaje de programación está escrita la plataforma Qt?**

- a. Python.
- b. C.
- c. C++.

- 3. Seleccione una de las características de QT Designer.**
  - a. Permite generar una interface mediante múltiples componentes: *layouts*, *spacer*, *buttons*, ítem views, ítem widgets, en ambiente web.
  - b. Permite generar una interface mediante múltiples componentes: *layouts*, *spacer*, *buttons*, ítem views, ítem widgets, etc.
  - c. Permite generar una interfaz mediante múltiples componentes: *layouts*, *spacer*, *buttons*, ítem views, ítem widgets con extensión .py.
- 4. Seleccione la extensión de los archivos generados por la plataforma Qt Designer**
  - a. Extensión .ui.
  - b. Extensión .py.
  - c. Extensión .pybn.
- 5. ¿Cuáles son las librerías que se deben instalar en Python para trabajar con Qt?**
  - a. PyQt5 / PyQt5-tools.
  - b. PyQt5 / PyQt5-browser.
  - c. Qt5 / Qt5-tools.
- 6. Dado un archivo generado por la plataforma Qt Designer. Seleccione el comando que se debe usar para hacer la transformación a lenguaje Python.**
  - a. pyuic5 -x ejemplo.py -o ejemplo.ui .
  - b. pyuic5 -x ejemplo.ui -o ejemplo.py.
  - c. python -x ejemplo.py -o ejemplo.ui.

- 7. ¿Cuál es la función del *widget* QMainWindow de la librería PyQt5?**
- a. QMainWindow, es un *widget* que proporciona el acceso a la base de datos.
  - b. QMainWindow, es un *widget* que proporciona la pantalla principal de la aplicación donde se despliegan los componentes.
  - c. QMainWindow, es un *widget* que proporciona la pantalla principal con el botón de salida, únicamente.
- 8. ¿Qué es un signal?**
- a. Un signal, es la notificación generada por un *widget* al momento que sucede alguna acción, por ejemplo, al presionar un botón.
  - b. Un signal, es la notificación generada por una base de datos al momento que sucede alguna acción, por ejemplo, al presionar un botón.
  - c. Un signal, es la notificación generada por Qt Designer al momento de que sucede alguna acción, por ejemplo, al presionar un botón.
- 9. Cuando se realiza una acción en un componente de PyQt5; se puede llamar a varios procesos con la misma acción al mismo tiempo?**
- a. Solo se puede llamar a un solo proceso.
  - b. Se puede realizar, ya que las acciones son paralelas y tienen procesos (lógica) diferentes.
  - c. Se puede llamar, pero en tiempos diferentes.

**10. Dado un elemento *widget* en PyQt5 de tipo QLineEdit. ¿Cuál es la forma de recuperar el valor ingresado por pantalla en dicho componente?**

- a. Se usa el nombre del componente generado y se llama al método text().
- b. Se usa el nombre del componente generado y se llama al método texto().
- c. Se usa el nombre del componente generado y se llama al método get\_text().

[Ir a solucionario](#)



## Semana 15

---

Estimado estudiante, durante la semana 15 de estudio, se recomienda volver a revisar, analizar y contestar las preguntas de las autoevaluaciones y las actividades recomendadas del primer bimestre.

A continuación se detalla un ejemplo que permite reforzar los contenidos estudiados en el bimestre.

### **Ejemplo 01**

Se debe generar una aplicación en el *framework* Django que permita agregar información a una entidad llamada empleados. Cada empleado tiene características como: nombre, apellido, edad, sueldo. Se de usar la interfaz de administración de Django para ingresar la información; y a través del uso de vistas en Django, se debe listar los empleados registrados.

Los paso para la solución se los lista a continuación:

1. Crear el proyecto de Django, usar el nombre: proyecto\_empleados  
`django-admin startproject proyecto_empleados.`
2. Ejecutar el proceso para inicializar las entidades en la base de datos  
`python manage.py migrate.`
3. Se crea una aplicación para el proyecto:  
`python manage.py startapp manejo_empleados.`

**Se agrega la aplicación al proyecto de Django**

```
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'manejo_empleados',
]
```

4. Generar el modelo de la aplicación, con base a los requerimientos dados. Se recuerda que el modelo se lo generar a través de clases en Python, en el archivo models.py.

```
from django.db import models

Create your models here.

class Empleado(models.Model):
 nombre = models.CharField(max_length=30)
 apellido = models.CharField(max_length=30)
 edad = models.IntegerField()
 sueldo = models.FloatField()

 def __str__(self):
 return """Nombre: %s -
 Apellido: %s \n
 Edad: %d\n
 Sueldo: %.2f\n
 """ % (self.nombre,
 self.apellido,
 self.edad,
 self.sueldo)
```

5. Se realiza la migración hacia la base de datos, con los siguientes comandos:

```
python manage.py makemigrations manejo_empleados.
python manage.py migrate.
```

6. Crear un usuario administrador del proyecto.

```
python manage.py createsuperuser.
```

7. Agregar la entidad del modelo creado en el archivo admin.py de la aplicación. Con el objetivo de administrar registros de tipo empleado desde el admin propio de Django.

```
from django.contrib import admin

Importar las clases del modelo
from manejo_empleados.models import Empleado

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

8. Iniciar el servidor propio de Django e ingresar la siguiente dirección en un navegador.
  - python manage.py runserver.
  - Ingresar a http://127.0.0.1:8000/admin.
9. Verificar que en la interfaz del admin de Django este disponible la entidad agregada. Luego, agregar registros de dicha entidad.
10. Agregar la información de las urls nuevas en los archivos urls.py, del proyecto y de la aplicación.

El archivo urls.py del proyecto queda la siguiente manera:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
 path('admin/', admin.site.urls),
 path('', include('manejo_empleados.urls')),
]
```

El archivo urls.py de la aplicación tiene el siguiente aspecto:

```
"""
Manejo de urls para la aplicación
"""

from django.urls import path
se importa las vistas de la aplicación
from . import views

urlpatterns = [
 path('listado/empleado', views.listar_empleados,
 name='listar'),
 path('', views.listar_empleados,
 name='listar'),
]
```

11. Agregar una función en el archivo views.py de la aplicación. La lógica que se necesita en esta función, es sencilla, obtener los registros de la base de datos.

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

importar las clases de models.py
from manejo_empleados.models import *

Create your views here.

def listar_empleados(request):
 """
 Listar los registros del modelo Empleado,
 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
 # empleados
 empleados = Empleado.objects.all()
 # se obtiene el número de elementos de la lista
 numero_empleados = len(empleados)
 # en la variable tipo diccionario llamada informacion_template
 # se agregará la información que estará disponible
 # en el template
 informacion_template = {'empleados': empleados,
 'numero_empleados': numero_empleados}
 return render(request, 'listar_empleados.html', informacion_template)
```

12. Crear una carpeta llamada templates, en la misma se agregan los siguientes archivos.

El archivo que será el template padre, del cual heredan características los templates hijos.

```

<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width" />
 {%- load static %}
 <link rel="stylesheet" type="text/css" href="{% static
'css/miestilo.css' %}">

 <link rel="stylesheet" type="text/css"
href="https://cdn.datatables.net/1.13.1/css/jquery.dataTables.min.css">
 <script
 src="https://code.jquery.com/jquery-3.6.3.min.js"
 integrity="sha256-
pvPw+upLPUjgMXY0G+800xUf+/Im1MZjXxxgOcBQBXU="
 crossorigin="anonymous"></script>
 <script type="text/javascript"
src="https://cdn.datatables.net/1.13.1/js/jquery.dataTables.min.js"></script>
 <t>
 <title>
 Proyecto Ejemplo Repaso
 </title>
 </head>
 <body>
 <h1>Aplicación para Empleados </h1>
 <hr>
 <section>
 {%- block contenido %}
 {%- endblock %}
 </section>

 <footer>
 <p>Loja - Ecuador</p>
 <p>{%- now "j F Y H:i" %}</p>
 </footer>
 </body>
</html>

```

```

{% extends "principal.html" %}
{% block contenido %}

<h3>Listado de Empleados</h3>
<p>Número de Empleados: {{numero_empleados}} </p>
<table id="tabla_datos">
 <thead>
 <tr>
 <th>Nombre</th>
 <th>Apellido</th>
 <th>Edad</th>
 <th>Sueldo</th>
 </tr>
 </thead>
 <tbody>
 {% for e in empleados %}
 <tr>
 <td>{{e.nombre}}</td>
 <td>{{e.apellido}}</td>
 <td>{{e.edad}}</td>
 <td>{{e.sueldo}}</td>
 </tr>
 {% endfor %}
 </tbody>
 <script type="text/javascript">
 $(document).ready(function () {
 $('#tabla_datos').DataTable();
 });
 </script>
{% endblock %}

```

- Finalmente reiniciar el servidor levantado e ingresar a la dirección <http://localhost:8000/>; en la misma se debe observar el listado de empleados creados.



Estimado estudiante, el ejemplo completo se lo puede descargar del [repositorio de Github](#) y realizar los cambios para su estudio.



### Actividades finales del bimestre

La semana 16, es un período de tiempo importante destinado para que usted estimado estudiante, pueda volver a revisar temáticas del segundo bimestre de estudio que aún están pendientes de comprender, con el objetivo de prepararse para rendir la evaluación presencial del bimestre.

Se propone para la presente semana el repaso de las siguientes temáticas:

- **Unidad 3:** conceptos avanzados de programación
  - 3.1. Programación en capas.
  - 3.2. Programación multihilo.
  - 3.3. Programación de interfaces e Interactiva.

Las estrategias sugeridas son:

- Elaboración de resúmenes, cuadros sinópticos, mapas conceptuales, etc; de las temáticas del bimestre.
- Revisar las actividades recomendadas en las semanas de estudio. Cuando lo amerite, recree los ejercicios desde cero, con el objetivo de familiarizarse con las herramientas y librerías usadas.
- Revisar los recursos de aprendizaje recomendados en el plan docente de la asignatura.
- Plantearse ejercicios y problemáticas similares a las expuestas en los contenidos de la guía didáctica.
- Revisar las preguntas que conforman las autoevaluaciones.
- Revisar las preguntas que conforman los cuestionarios de repaso del bimestre planteados en el entorno virtual de aprendizaje.



¡Hemos concluido el estudio del componente  
académico!



### 3. Solucionario

| Autoevaluación 1 |           |                                                                                                                                                                                                                                                                                                                                                  |
|------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pregunta         | Respuesta | Retroalimentación                                                                                                                                                                                                                                                                                                                                |
| 1                | a         | Las actividades que se realizan en el proceso de desarrollo de software son: Especificaciones de requerimientos, diseño, codificación, validación y mantenimiento                                                                                                                                                                                |
| 2                | c         | La otra forma de denominar a los modelos del ciclo de vida del software es paradigma del proceso                                                                                                                                                                                                                                                 |
| 3                | c         | Existen tres principales tipos de modelos del ciclo de vida: Modelo en cascada, Modelos de desarrollo evolutivo, Modelos de componentes reutilizables                                                                                                                                                                                            |
| 4                | a         | Las fases del modelo en cascada son: Especificación (Análisis y definición de requerimientos); Implementación (Diseño, Codificación, Validación), Mantenimiento.                                                                                                                                                                                 |
| 5                | b         | La acción que aborda la realización de la primera aproximación de la arquitectura del sistema, está considerada como objetivo en la fase de diseño.                                                                                                                                                                                              |
| 6                | a         | La fase del desarrollo evolutivo a la que pertenece la acción que permite generar un prototipo con una simulación de la funcionalidad es la denominada prototipos desecharables                                                                                                                                                                  |
| 7                | a         | Las ventajas son: El equipo de desarrollo está preparado por los cambios de requisitos; a través de las fases se trata vincular a los desarrolladores con los usuarios finales. En las otras respuestas, al menos una no es ventaja, por tal razón no se considera. Ejemplo: se puede detectar en finales errores, que eviten pérdidas al inicio |
| 8                | c         | Los valores establecidos en el manifiesto ágil son: Valorar a las personas y las interacciones del equipo; desarrollar software que funcione; colaborar con el cliente; responder a los cambios.                                                                                                                                                 |
| 9                | a         | Las metodologías ágiles presentadas en las opciones de respuesta son: Metodología Scrum; metodología XP; Metodología Kanban.                                                                                                                                                                                                                     |

| Autoevaluación 1 |           |                                                                                                                                                                                       |
|------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pregunta         | Respuesta | Retroalimentación                                                                                                                                                                     |
| 10               | b         | Procesos de acuerdo / procesos organizacionales / procesos de gestión técnica / procesos técnicos; son considerados como los grupos de procesos establecidos en la ISO/IEC 12207:2017 |

[Ir a la  
autoevaluación](#)

| Autoevaluación 2 |           |                                                                                                                                                                                                                                                                                   |
|------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pregunta         | Respuesta | Retroalimentación                                                                                                                                                                                                                                                                 |
| 1                | a         | El concepto que se acerca a la definición de un control de versiones es: un control de versiones, se convierte en un mecanismo o proceso que permite registrar en el tiempo el conjunto de cambios que se ha realizado para uno o varios archivos                                 |
| 2                | c         | Sistema de control de versiones locales; Sistema de control de versiones centralizados; Sistema de control de versiones distribuidos; son los tres tipos principales de sistemas de control de versiones.                                                                         |
| 3                | b         | Los máximos exponentes en los sistemas de control de versiones distribuidos son: Git; Mercurial; Bazaar                                                                                                                                                                           |
| 4                | b         | El creador de Git es Linus Torvalds en el año 2005; quien decidió realizar una herramienta de control de versiones para manejar el código del núcleo de Linux.                                                                                                                    |
| 5                | a         | <p>Los comandos:</p> <pre>git config --global user.name "Nombre de Usuario" git config --global user.email nombre@ejemplo.com</pre> <p>Son los que generan la información en relación al nombre de usuario y correo electrónico que se usa al momento de registrar un commit.</p> |
| 6                | b         | <p>Se usa el comando git init, dicho comando crea un subdirectorio importante y base llamado .git, el mismo contiene la estructura que necesita Git. La secuencia de comandos que puede generar lo indicado es:</p> <pre>mkdir demo cd demo git init</pre>                        |
| 7                | b         | El comando que se usa para empezar a darle seguimientos a los nuevos directorios o archivos, bajo la filosofía de Git es: git add                                                                                                                                                 |
| 8                | a         | La finalidad de un branch en un repositorio es la de permitir tomar como base la rama por defecto que tienen un repositorio en Git y comenzar a trabajar de forma paralela e independiente.                                                                                       |
| 9                | a         | El comando que permite crear una rama en un repositorio: git branch nuevarama                                                                                                                                                                                                     |

## Autoevaluación 2

| Pregunta | Respuesta | Retroalimentación                                                                             |
|----------|-----------|-----------------------------------------------------------------------------------------------|
| 10       | c         | El comando que permite acceder a una rama, previamente creada es: git checkout branch mirama. |

Ir a la  
autoevaluación

| Autoevaluación 3 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pregunta         | Respuesta | Retroalimentación                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 1                | a         | <p>Las sentencias que permiten realizar una conexión a una base de datos SQLite son las siguientes:</p> <pre>import sqlite3 conn = sqlite3.connect('base_ejemplo.db')</pre> <p>Si se ubica las líneas anteriores en un archivo .py; desde otro archivo .py se puede importar la variable conn.</p>                                                                                                                                  |
| 2                | c         | <p>Las sentencias que permiten la creación de una entidad en una base de datos SQLite desde python son:</p> <pre>from base_datos import conn cursor = conn.cursor() cadena_sql = 'CREATE TABLE Ciudad (nombre TEXT, poblacion INTEGER)' cursor.execute(cadena_sql) cursor.close()</pre> <p>En las demás opciones el proceso no está completo; no se crea el cursor o no se hace la ejecución de la cadena, a través de execute.</p> |
| 3                | b         | <p>Las sentencias que permiten ingresar un nuevo registro a la base de datos son:</p> <pre>apellido = "José Lima" pasaporte = "10012391xa1" cadena_sql = """INSERT INTO Persona (apellido, pasaporte) VALUES ('%s', '%s');""" % (apellido, pasaporte) cursor.execute(cadena_sql)</pre>                                                                                                                                              |
| 4                | a         | <p>De las opciones solo la frase, tratar la información con estructuras de datos que contienen objetos; es una característica de los ORM.</p>                                                                                                                                                                                                                                                                                       |
| 5                | b         | <p>Eloquent es una librería ORM para PHP</p> <p>Pony ORM es una librerías ORM para Python</p>                                                                                                                                                                                                                                                                                                                                       |

| Autoevaluación 3 |           |                                                                                                                                                                                                               |
|------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pregunta         | Respuesta | Retroalimentación                                                                                                                                                                                             |
| 6                | a         | La forma correcta de instalar una librería en Python es a través de comando pip.                                                                                                                              |
| 7                | c         | La forma correcta para agregar un atributo tipo INTEGER y que sea considerada como llave primaria en SQLAlchemy es:<br><br>id = Column(Integer, primary_key=True)                                             |
| 8                | c         | La sentencia que permite realizar una consulta de todos los registros de una entidad llamada País, a través de SQLAlchemy:<br>session.query(Pais).all()                                                       |
| 9                | b         | Las sentencias que permite realizar una conexión a la base de datos MongoDB desde Python son:<br><br>from pymongo import MongoClient<br><br>client = MongoClient('mongodb://localhost:27010/')                |
| 10               | c         | Las sentencias que permite agregar información a una colección desde Python a MongoDB son:<br><br>data_01 = {"nacionalidad": "ecuatoriana", "numero_publicaciones": 100}<br><br>colección.insert_one(data_01) |

Ir a la  
autoevaluación

| Autoevaluación 4 |           |                                                                                                                                                                                  |
|------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pregunta         | Respuesta | Retroalimentación                                                                                                                                                                |
| 1                | a         | Los modelos físicos arquitectónicos, permiten detallar la forma como se distribuye la infraestructura o el hardware que son los encargados de ejecutar los procesos del sistema. |
| 2                | c         | El poder genera soluciones flexibles y escalables es una característica de los modelos de arquitectura lógica.                                                                   |
| 3                | b         | Modelo – Vista – Controlador es modelo asociado a la patrón asociado a la arquitectura en capas.                                                                                 |
| 4                | b         | En relación al modelo; se encargan de construir la lógica de la aplicación, por lo general estos modelos trabajan directamente con una base de datos.                            |
| 5                | c         | Gestiona o administra las funcionalidades del sistema; se habla de las reglas del negocio. Es una de las características de la capa lógica.                                      |
| 6                | a         | El comando para la creación de un proyecto en el framework Django tiene la siguiente estructura:<br><br>django-admin startproject [nombre-proyecto]                              |
| 7                | b         | El comando que permite inicializar las entidades en la base de datos, de un proyecto en Django es python manage.py migrate                                                       |
| 8                | a         | El comando que se usa para generar una aplicación en un proyecto de Django es:<br><br>python manage.py startapp [nombre de la app]                                               |
| 9                | b         | La forma correcta de agregar una clase del modelo al admin de Django es:<br><br>admin.site.register(Jugador)                                                                     |
| 10               | a         | La forma para poder obtener todos los registros de la base de datos, a través del ORM de Django de una entidad llamada Jugador es:<br><br>Jugador.objects.all()                  |

Ir a la  
autoevaluación

| Autoevaluación 5 |           |                                                                                                                                                                                                                                                                              |
|------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pregunta         | Respuesta | Retroalimentación                                                                                                                                                                                                                                                            |
| 1                | a         | El principal objetivo de la programación en paralelo es la poder resolver los tiempos elevados que puedan existir al momento de ejecutar un proceso.                                                                                                                         |
| 2                | c         | Los pasos recomendados para poder resolver un problema forma paralela son: Intenta la dividir el problema en subproblemas independientes / Generar una solución para cada subproblema / Ejecutar cada subproblema en un procesador / Mantener el control de los subproblemas |
| 3                | a         | <p>La división en función del método es:</p> <p>A nivel de bit</p> <p>A nivel de instrucción</p> <p>A nivel de datos</p> <p>A nivel de tareas</p>                                                                                                                            |
| 4                | a         | El mayor consumo de recursos es una de las desventajas del uso de la programación en paralelo.                                                                                                                                                                               |
| 5                | a         | Un hilo es conceptualizado como un conjunto de instrucciones que están dentro de un determinado proceso de forma independiente.                                                                                                                                              |
| 6                | c         | La librería para manejar hilos en Python se denomina threading                                                                                                                                                                                                               |
| 7                | b         | <p>La forma correcta para crear un hilo en Python es de la siguiente forma:</p> <pre>import threading hilo1 = threading.Thread(target=[nombre de la función])</pre>                                                                                                          |
| 8                | a         | La sentencia que permite obtener el nombre del hilo que está en ejecución es: <code>threading.current_thread().getName()</code>                                                                                                                                              |
| 9                | c         | La sentencia que permite obtener el identificador del hilo que está en ejecución es:                                                                                                                                                                                         |
| 10               | c         | La librería threading viene por defecto con la instalación de Python                                                                                                                                                                                                         |

[Ir a la autoevaluación](#)

| Autoevaluación 6 |           |                                                                                                                                                                     |
|------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pregunta         | Respuesta | Retroalimentación                                                                                                                                                   |
| 1                | a         | De acuerdo a lo estudiado, Qt como es un framework multiplataforma para crea aplicaciones gráficas (GUIs -Interfaces Gráficas de Usuario)                           |
| 2                | c         | La plataforma Qt está escrita en lenguaje C++                                                                                                                       |
| 3                | b         | Permite generar una interface mediante múltiples componentes: Layouts, Spacer, Buttons, Item Views, Item Widgets, etc; es una de las características de QT Designer |
| 4                | a         | Le extensión con la que se genera los archivos desde la plataforma Qt Designer es: .ui                                                                              |
| 5                | a         | Las librerías necesarias para trabajar con Qt desde Python son: PyQt5 / PyQt5-tools                                                                                 |
| 6                | b         | La forma correcta para transformar un archivo.ui en .py es:<br>pyuic5 -x ejemplo.ui -o ejemplo.py                                                                   |
| 7                | b         | QMainWindows, es un Widget que proporciona la pantalla principal de la aplicación donde se despliegan los componentes.                                              |
| 8                | a         | Un signal, es la notificación generada por un Widget al momento de que sucede alguna acción, por ejemplo, al presionar un botón.                                    |
| 9                | b         | En PyQt5 si se puede realizar, ya que las acciones son paralelas y tienen procesos (lógica) diferentes.                                                             |
| 10               | a         | Se usa el nombre del componente generado y se llama al método text()                                                                                                |

Ir a la  
autoevaluación



---

## 4. Referencias bibliográficas

---

- Farias, J. J. S., Frías, R. T., González, L. A. L., & Vázquez, J. I. C. (2016). Persistencia de datos con ActiveJDBC ORM. *Pistas Educativas*, 38(p. 122).
- Elizalde, R (2020). Guía didáctica de Desarrollo Basado en Plataformas Web. Loja. *Universidad Técnica Particular de Loja*.
- Ojeda, J. C., & Fuentes, M. D. C. G. (2012). Taxonomía de los modelos y metodologías de desarrollo de software más utilizados. *Universidades*, (52), pp. 37-47.
- Martínez, P. J. S., Montequín, V. R., Balsera, J. M. V., & Cuiñas, M. C. (2014). Selección de modelos y metodologías agiles en proyectos software. In *Proceedings from the 18th International Congress on Project Management and Engineering (Alcañiz, July 2014)* (pp. 1862-1873). Asociación española de ingeniería de proyectos (AEIPRO).
- Hernández Bejarno, M. (2020). *Ciclo de vida de desarrollo ágil de software seguro..* Fundación Universitaria Los Libertadores. <https://elibro.net/es/lc/bibliotecatpl/titulos/197008>.
- Vara Mesa, J. M. Verde Marín, J. & López Sanz, M. (2015). *Desarrollo web en entorno servidor..* RA-MA Editorial. <https://elibro.net/es/lc/bibliotecatpl/titulos/62489>.
- Moreno Muñoz, A., & Córcoles Córcoles, S. (2019). *Python Práctico*. Rama Editorial. <https://www.digitaliapublishing.com/a/110158>.
- Gómez Checa, J. (2018). Entorno de Soporte a la Mejora y Certificación de Procesos Software basado en estándares ISO/IEC 33000 e ISO/IEC 12207.
- Vázquez-Ingelmo, A., & García-Peñalvo, F. J. (2019). Proceso.



## 5. Anexos

### Pasos para agregar una función en el archivo views.py, llamada listar

- a. Agregar un archivo *urls.py* en la aplicación. Permite manipular las direcciones de la aplicación. Relacionarlo con el archivo *urls.py* del proyecto.

```
urls de la aplicación
from django.urls import path
se importa las vistas de la aplicación
from . import views

urlpatterns = [
 path('listar', views.listar, name='listar'),
 path("", views.listar, name='listar'),
]

urls del proyecto
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
 path('admin/', admin.site.urls),
 path('ligaprof/', include('campeonato.urls')),
]
```

- b. Incluir en la función las siguientes líneas, que permiten:

- Obtener la información de la base de datos.
- Crear una variable que traslade información desde la vista al template.

```
def listar(request):
```

~~om~~

        Listar los registros del modelo Equipo,

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
equipos
equipos = Equipo.objects.all()
se obtiene el número de elementos de la lista
numero_equipos = len(equipos)
en la variable tipo diccionario llamada informacion_template
se agregará la información que estará disponible
en el template
informacion_template = {'equipos': equipos, 'numero_equipos': numero_
equipos}
return render(request, 'listar_equipo.html', informacion_template)
```

- c. Crear una carpeta llamada *templates* en el directorio de la aplicación. Agregar un archivo que se llame, *listar\_equipo.html*. Además, se genera un archivo llamado *principal.html*; en este archivo se originará el esqueleto base o plantilla de los templates.

```
principal.html
<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width" />
 {% load static %}
 <link rel="stylesheet" type="text/css" href="{% static 'css/
miestilo.css' %}">
 <link rel="stylesheet" type="text/css" href="https://cdn.
datatables.net/1.13.1/css/jquery.dataTables.min.css">

<script
 src="https://code.jquery.com/jquery-3.6.3.min.js"
 integrity="sha256-pvPw+upLPUjgMXY0G+800xUf+/
 Im1MZjXxxg0cBQBXU="
 crossorigin="anonymous"></script>
```

```

<script type="text/javascript" src="https://cdn.datatables.net/1.13.1/
js/jquery.dataTables.min.js"></script>
<title>
 Proyecto Ejemplo
</title>
</head>
<body>
 <h1>Aplicación Equipos </h1>
 <hr>
 <section>
 {% block contenido %}
 {% endblock %}
 </section>

 <footer>
 <p>Loja-Ecuador</p>
 <p>{% now "j F Y H:i" %}</p>
 </footer>
</body>
</html>

```

- d. Incluir en el archivo *listar\_equipo.html*, las siguientes líneas de código, que permiten visualizar la lista de objetos, que vienen desde la vista.

```

listar_equipo.html
 {% extends "principal.html" %}
 {% block contenido %}
 <h3>Lista de Equipos</h3>
 <p>Número de equipos: {{numero_equipos}} </p>
 <table id="tabla_datos">
 <thead>
 <tr>
 <th>Nombre</th>
 <th>Siglas</th>
 <th>Estadio</th>
 <th>Seguidores</th>
 <th>Campeonatos</th>
 </tr>
 </thead>
 <tbody>

```

```

{%
 for equipo in equipos %
}
<tr>
 <td>{{equipo.nombre}}</td>
 <td>{{equipo.siglas}}</td>
 <td>{{equipo.estadio}}</td>
 <td>{{equipo.seguidores}}</td>
 <td>{{equipo.campeonatos}}</td>
</tr>
{%
 endfor %
}
</table>
</tbody>
<script type="text/javascript">
$(document).ready(function () {
 $('#tabla_datos').DataTable();
});
</script>
{%
 endblock %
}

```

- e. Ingresar la siguiente dirección: <http://localhost:8000/ligapro/listar>.

Se debe visualizar los registros de la entidad equipo.

**Figura 47.**

*Listado de objetos a través de una función del views y un template*

#### Aplicación Equipos

| Lista de Equipos                                                         |        |              |            |             |  |
|--------------------------------------------------------------------------|--------|--------------|------------|-------------|--|
| Número de equipos: (2)                                                   |        |              |            |             |  |
| Show <select>10</select> entries <input type="text" value="Search: 17"/> |        |              |            |             |  |
| Nombre                                                                   | Siglas | Estadio      | Seguidores | Campeonatos |  |
| Barcelona                                                                | BSC    | Isidro Carbo | 30000      | 17          |  |

Showing 1 to 1 of 1 entries (filtered from 2 total entries)

Loja-Ecuador  
30 January 2023 07:44

Previous  Next

Nota.