Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DEL LABORATORIO DE PROYECTOS DE COMPUTACIÓN APLICADOS A INGENIERÍA ELECTRÓNICA PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Jonathan Mardoqueo Lorenzo Lopez
Asesorado por el Ing. Kenneth Issur Estrada Ruíz

Guatemala, febrero de 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DEL LABORATORIO DE PROYECTOS DE COMPUTACIÓN APLICADOS A INGENIERÍA ELECTRÓNICA PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JONATHAN MARDOQUEO LORENZO LOPEZ
ASESORADO POR EL ING. KENNETH ISSUR ESTRADA RUÍZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, FEBRERO DE 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. José Francisco Gómez Rivera (a.i.)
VOCAL II	Ing. Mario Renato Escobedo Martinez
VOCAL III	Ing. José Milton De León Bran
VOCAL IV	Ing. Kevin Vladimir Cruz Lorente
VOCAL V	Ing. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. José Francisco Gómez Rivera (a.i.)
EXAMINADOR	Ing. Armando Alonso Rivera Carrillo
EXAMINADOR	Ing. Kenneth Issur Estrada Ruiz
EXAMINADOR	Ing. Natanael Jonathan Requena Gómez
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DEL LABORATORIO DE PROYECTOS DE COMPUTACIÓN APLICADOS A INGENIERÍA ELECTRÓNICA PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 18 de febrero de 2022.

Jonathan Mardoqueo Lorenzo Lopez

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Unidad de EPS

Guatemala, 28 de agosto de 2023.
REF.EPS.DOC.362.08.2023.

Ing. Oscar Argueta Hernández
Director Unidad de EPS
Facultad de Ingeniería
Presente

Estimado Ingeniero Argueta Hernández.

Por este medio atentamente le informo que como Supervisor de la Práctica del Ejercicio Profesional Supervisado (E.P.S.), del estudiante universitario **Jonathan Mardoqueo Lorenzo Lopez** de la Carrera de Ingeniería Electrónica, Registro Académico **No. 201709532 y CUI 3000 22379 0101,** procedí a revisar el informe final, cuyo título es **"DISEÑO DEL LABORATORIO DE PROYECTOS DE COMPUTACIÓN APLICADOS A INGENIERIÍA ELECTRÓNICA PARA LA ESCUELA DE INGENIERIÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA".**

En tal virtud, **LO DOY POR APROBADO**, solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,

"Id y Enseñad a Todos"

Universidad de San Carlos de Guatemala
Ing. Kenneth Issur Estrada Ruiz
ASESOR - SUPERVISOR DE EPS
Área de Ingeniería Eléctrica
Unidad de EPS y Práctica Profesional Supervisada
Facultad de Ingeniería

Ing. Kenneth Issur Estrada Ruiz
Asesor-Supervisor de EPS
Área de Ingeniería Eléctrica

c.c. Archivo
KIER/ra

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Unidad de EPS

Guatemala 29 de agosto de 2023.
REF.EPS.D.273.08.2023.

Ing. Armando Alonso Rivera Carrillo
Director Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería
Presente

Estimado Ingeniero Rivera Carrillo.

Por este medio atentamente le envío el informe final correspondiente a la práctica del Ejercicio Profesional Supervisado, (E.P.S) titulado **"DISEÑO DEL LABORATORIO DE PROYECTOS DE COMPUTACIÓN APLICADOS A INGENIERIÍA ELECTRÓNICA PARA LA ESCUELA DE INGENIERIÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA"** que fue desarrollado por el estudiante universitario, **Jonathan Mardoqueo Lorenzo Lopez,** quien fue debidamente asesorado y supervisado por el Ing. Kenneth Issur Estrada Ruiz.

Por lo que habiendo cumplido con los objetivos y requisitos de ley del referido trabajo y existiendo la aprobación del mismo por parte del Asesor - Supervisor de EPS, en mi calidad de Director apruebo su contenido solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,
"Id y Enseñad a Todos"

Universidad de San Carlos de Guatemala
DIRECCIÓN
Unidad de Prácticas de Ingeniería y EPS
Facultad de Ingeniería

Ing. Oscar Argueta Hernández
Director Unidad de EPS

/ra

ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

REF. EIME 51.2023.

El director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del director de EPS, del asesor, con el Visto Bueno del coordinador de área, al Informe final de EPS del estudiante Jonathan Mardoqueo Lorenzo Lopez: "**DISEÑO DEL LABORATORIO DE PROYECTOS DE COMPUTACIÓN APLICADOS A INGENIERÍA ELECTRÓNICA PARA LA ESCUELA DE INGENIERIÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA** ", procede a la autorización correspondiente.

Ing. Armando Alonso Rivera Carrillo

Guatemala, 27 de septiembre de 2023.

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.092.2024

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **DISEÑO DEL LABORATORIO DE PROYECTOS DE COMPUTACIÓN APLICADOS A INGENIERÍA ELECTRÓNICA PARA LA ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA DE LA FACULTAD DE INGENIERÍA, UNIVERSIDAD DE SAN CARLOS DE GUATEMALA,** presentado por: **Jonathan Mardoqueo Lorenzo Lopez,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO a.i.
Facultad de Ingeniería

Ing. José Francisco Gómez Rivera

Decano a.i.

Guatemala, febrero de 2024

JFGR/gaoc

ACTO QUE DEDICO A:

Dios Por ser un pilar muy importante durante todo el proceso de mi carrera, dándome perseverancia todos los días para poder concluir mi carrera.

Mis padres Mardoqueo Lorenzo Coronado y Margarita Lopez Quina, por su apoyo incondicional durante todo este proceso, apoyándome en situaciones difíciles, y dándome el ejemplo de perseverar para poder cumplir todas las metas que me proponga.

Mis hermanos Kevin Alexander Lorenzo Lopez y Keila Lucia Lorenzo Lopez, por su apoyo y por siempre estar presentes ante cualquier necesidad.

Mis amigos José Carlos Sagastume Ovalle y Angel Meoli Ajuchán Méndez por darme siempre ánimos para poder concluir mi carrera, y por todo el apoyo brindado durante toda la carrera.

Los ingenieros Mario René De León García, Glenda Patricia García Soria, Mario Gustavo López Hernández, Ingrid Rodríguez De Loukota, Kenneth Issur Estrada Ruíz, Luis Eduardo Durán Córdova,

Danilo Escobar, Lic. Ricardo Enrique Contreras Folgar, Dr. Juan Carlos Córdova, por ser un pilar muy importante en mi carrera, por mostrarme que ser catedrático no es solo impartir el contenido del programa del curso, sino que implica muchos más aspectos.

AGRADECIMIENTOS A:

La Universidad de San Carlos de Guatemala	Por ser la casa de estudios que me permitió culminar mi carrera universitaria, y por ser una institución que imparte la educación superior a muchas personas, logrando un desarrollo en el país.
Facultad de Ingeniería	Por brindarme todos los conocimientos necesarios para poder culminar mi carrera, entre otras cosas.
Población de Guatemala	Por brindarme la oportunidad de poder tener el acceso a la educación superior.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
I	Corriente
m	Metro
R	Resistencia
s	Segundo
V	Voltaje

GLOSARIO

Analógico	Señal continua en el tiempo.
Antena	Dispositivo que permite enviar y recibir señales.
Bash	Es un intérprete de distintos comandos nativo de sistemas basados en el Kernel Linux.
BIOS	Es el programa integrado en el procesador que sirve para iniciar el sistema.
Código	Conjunto de instrucciones que ejecutan una acción en un computador.
Digital	Señal no continua en el tiempo.
Driver	Componente que permite a distintos dispositivos comunicarse.
Hardware	Dispositivos físicos que confirman un ordenador.
Kernel	Es el núcleo de un sistema operativo.
Login	Es una etapa de identificación para poder ingresar a un sistema.

Multiusuario	Sistema que permite ser utilizado por múltiples usuarios al mismo tiempo.
Root	Es una cuenta de usuario que no tienen ninguna restricción, ya que posee absolutamente todos los derechos que un usuario puede tener.
Script	Bloque de código que realiza una acción dentro de un programa.
Software	Programas que permiten realizar tareas.

RESUMEN

El Ejercicio Profesional Supervisado propuesto consiste en diseñar la estructura y las prácticas del curso de proyectos de computación aplicados a la ingeniería electrónica, para la Escuela de Ingeniería Mecánica Eléctrica, con el fin de poder tener un complemento para las clases teóricas de este curso.

Los estudiantes de la carrera de Ingeniería Electrónica adquirirán conocimientos amplios sobre temas de programación, bases de datos y herramientas de visualización de datos. El aprendizaje de estos temas les dará a los estudiantes más herramientas para que puedan desenvolverse sin ningún problema en cualquier ámbito de tecnología, ya que podrán combinar todos sus conocimientos de electrónica con los conocimientos de programación y así podrán afrontar y solucionar de una forma eficaz los problemas que se les presenten.

OBJETIVOS

General

Diseñar el laboratorio de proyectos de computación aplicados a ingeniería electrónica para la escuela de ingeniería mecánica eléctrica de la Facultad de Ingeniería, Universidad de San Carlos de Guatemala.

Específicos

1. Diseñar las prácticas del laboratorio apegadas al contenido teórico de la clase magistral para mejorar el proceso de enseñanza aprendizaje en la formación académica de los estudiantes de la carrera de ingeniería electrónica.

2. Brindar una guía didáctica de orientación para el desarrollo del contenido teórico-práctico, al catedrático que impartirá el laboratorio.

3. Presentar fundamentos teóricos de Linux, Python, lenguaje SQL y *software* de visualización de datos, así como ejemplos prácticos para el laboratorio.

INTRODUCCIÓN

Actualmente en la carrera de Ingeniería Electrónica de la Escuela de Ingeniería Mecánica Eléctrica se tiene cierta deficiencia en ciertos cursos en la formación de ingenieros electrónicos. Específicamente hablando del curso de proyectos de computación aplicados a la ingeniería electrónica, si bien existe una clase magistral que imparte los fundamentos teóricos necesarios, se reconoce la falta de un espacio práctico que permita a los estudiantes aplicar y profundizar su comprensión en los temas del curso.

El estudiante de ingeniería electrónica ya posee conocimientos de programación, por ello se plantea el diseño del laboratorio para poder darle a los estudiantes de ingeniería electrónica nuevas herramientas que podrán utilizar en el área laboral de sus vidas profesionales.

Por esta razón, se realizó el proceso de diseño del laboratorio. Este espacio proporcionará a los estudiantes las herramientas y el entorno adecuado para experimentar y aplicar conceptos de la clase magistral. Este laboratorio no solo cerrará la brecha entre la teoría y la práctica, sino que también equipará a los futuros ingenieros electrónicos con habilidades concretas y experiencia práctica que enriquecerán su perfil profesional y les dará una ventaja competitiva en el entorno laboral.

1. ANTECEDENTES GENERALES

1.1. USAC

A continuación, se presenta la información general de la Universidad de San Carlos de Guatemala.

1.1.1. Misión

En su carácter de única universidad estatal le corresponde con exclusividad dirigir, organizar y desarrollar la educación superior del Estado y la educación estatal, así como la difusión de la cultura en todas sus manifestaciones. Promoverá por todos los medios a su alcance la investigación en todas las esferas del saber humano y cooperará al estudio y solución de los problemas nacionales. (USAC, s.f., párr. 1)

1.1.2. Visión

La Universidad de San Carlos de Guatemala es la institución de educación superior estatal, autónoma, con cultura democrática, con enfoque multi e intercultural, vinculada y comprometida con el desarrollo científico, social, social, humanista y ambiental con una gestión actualizada, dinámica, efectiva y con recursos óptimamente utilizados para alcanzar sus fines y

objetivos, formadora de profesionales con principios éticos y excelencia académica. (USAC, s.f., párr. 2)

1.2. Facultad de Ingeniería

A continuación, se presenta la información general de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala.

1.2.1. Historia

Desde 1676, en sus primeras épocas, la Universidad de San Carlos graduaba teólogos y abogados; posteriormente, a médicos. En 1769 se crearon cursos de física y geometría, lo que marcó el inicio de la enseñanza de las ciencias exactas en Guatemala.

En 1834, cuando el jefe de Estado de Guatemala era Mariano Gálvez, se creó la Academia de Ciencias, sucesora de la Universidad de San Carlos se implantó la enseñanza de álgebra, geometría, trigonometría y física, además, de otorgar títulos de agrimensores. Francisco Colmenares, Felipe Molina, Patricio de León y José Batres Montúfar fueron los primeros graduados.

La Academia de ciencia funcionó hasta 1840, hasta que, en el gobierno de Rafael Carrera volvió a transformarse la universidad,

mediante los cuales exigían que para obtener el título de agrimensor era necesario poseer el título de bachiller en filosofía, tener un año de práctica y aprobar el examen correspondiente.

En 1879 se estableció la Escuela de Ingeniería en la Universidad de San Carlos de Guatemala; por decreto del Gobierno, pero en 1882, se tituló como Facultad dentro de la institución y se separó de la Escuela Politécnica.

En 1959 se creó el Centro de Investigaciones de Ingeniería, para fomentar y coordinar la investigación científica con participación de varias instituciones públicas y privadas.

En 1966 en la Facultad de Ingeniería se estableció el primer programa regional (centroamericano) de estudios de Postgrado, mediante la creación de la Escuela Regional de Ingeniería Sanitaria y la maestría en ingeniería sanitaria. Estos estudios son reconocidos internacionalmente. Después, ese programa se amplió con la maestría en recursos hidráulicos.

Durante el período comprendido de 2001 a 2005 se iniciaron las maestrías de ciencias de ingeniería vial, gestión industrial, desarrollo

municipal y mantenimiento industrial. Y en 2007 se creó la carrera de ingeniería ambiental, con grado de licenciatura.

En los años siguientes se establecieron convenios con universidades europeas como la de Cádiz, de Almería y la Tecnológica de Madrid, para la realización de intercambios estudiantiles. En ese año concluyó el proceso que le otorgó la acreditación a la carrera de ingeniería química. Además, en ese período se inició el proceso en busca de la acreditación de la carrera de ingeniería civil. (FIUSAC, s.f., párr. 1-8)

1.2.2. Misión

Formar profesionales en las distintas áreas de la ingeniería que, a través de la aplicación de la ciencia y la tecnología conscientes de la realidad nacional y regional, y comprometidos con nuestras sociedades, sean capaces de generar soluciones que se adapten a los desafíos del desarrollo sostenible y los retos del contexto global. (FIUSAC, s.f., párr. 24)

1.2.3. Visión

Ser una institución académica con incidencia en la solución de la problemática nacional; formamos profesionales en las distintas áreas de la ingeniería, con sólidos conceptos científicos, tecnológicos, éticos y

sociales, fundamentados en la investigación y promoción de procesos innovadores orientados hacia la excelencia profesional. (FIUSAC, s.f., párr. 25)

1.3. Escuela de Ingeniería Mecánica Eléctrica

La Escuela de Ingeniería Mecánica Eléctrica forma parte de la Facultad de Ingeniería en la Universidad de San Carlos de Guatemala. Es la encargada de formar profesionales a nivel de licenciatura en las áreas de eléctrica, mecánica eléctrica y electrónica. Cuenta con una trayectoria de más de 50 años. Las funciones docentes y administrativas toman lugar en el edificio T-3 y T-1 del campus central de la universidad.

1.3.1. Historia

La Escuela de Ingeniería Mecánica Eléctrica se creó en 1968; a su cargo quedaron las carreras de ingeniería eléctrica y la combinada de ingeniería mecánica eléctrica.

1.3.2. Misión

Formar profesionales competentes, con principios éticos y conciencia social, en los campos de las Ingenierías Mecánica Eléctrica, Eléctrica y Electrónica, mediante técnicas de enseñanza actualizadas y fundamentados en la investigación, comprometidos con la sociedad, con el fin de contribuir al bien común y al desarrollo sostenible del país y de la región. (Picén, 2012, p.10)

1.3.3. Visión

Ser la institución académica líder a nivel nacional y regional, con incidencia en la problemática nacional, en la formación de profesionales de calidad, en los campos de las Ingenierías Mecánica Eléctrica, Eléctrica y Electrónica, emprendedores, con sólidos conocimientos científicos, tecnológicos, éticos, sociales, fundamentados en la investigación, orientados hacia la excelencia, reconocidos internacionalmente y comprometidos con el desarrollo sostenible de Guatemala y de la región. (Picén, 2012, p.10)

2. DIAGNÓSTICO DE LA SITUACIÓN ACTUAL

2.1. Red de estudios de la carrera de Ingeniería Electrónica

La carrera de ingeniería electrónica actualmente consta de 10 semestres divididos en 5 áreas que son: Ciencias básicas y complementarias, Electrotecnia, Electrónica, Potencia y EPS. Los estudiantes deben cursar y aprobar todas las materias de carácter obligatorio y completar los 300 créditos CLAR para poder terminar la carrera.

2.1.1. Ciencias básicas y complementarias

Esta área incluye todos los cursos básicos para todas las carreras de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala. Estos cursos tienen el objetivo de brindarle conocimientos básicos al estudiante en diferentes campos de la ciencia como matemática, física, química y ciencias sociales. Con estos cursos se espera que el estudiante pueda comprender conceptos básicos en las distintas ciencias, por ejemplo: el tiempo que en el sistema internacional se mide en segundos (s), la distancia (m) y la velocidad (m/s), donde estas cantidades son de vital importancia durante toda la carrera. Estos cursos son impartidos entre el primer y quinto semestre de la carrera de Ingeniería Electrónica, y muchos de estos cursos son prerrequisito para los cursos del área profesional. En la tabla 1 se muestra a detalle la información de los cursos que conforman esta área.

Tabla 1.

Cursos del área de Ciencias Básicas y Complementarias

Semestre	Código del curso	Nombre del curso	Cantidad de créditos
1er.	0017	Área Social Humanística 1	3
	0069	Área Técnica Complementaria 1	4
	0101	Área Matemática Básica 1	9
	0348	Química General 1	7
	0006	Idioma Técnico 1 *	3
	0039	Deportes 1 *	2
2do.	0005	Técnicas de Estudio e Investigación	3
	0019	Área Social Humanística 2	3
	0103	Área Matemática Básica 2	9
	0147	Física Básica	5
	0008	Idioma Técnico 2 *	3
	0040	Deportes 2 *	2
3er.	0107	Área Matemática Intermedia 1	9
	0150	Física 1	5
	0018	Filosofía de la Ciencia	1
	0009	Idioma Técnico 3 *	3
4to.	0732	Estadística 1	5
	0112	Área Matemática Intermedia 2	6
	0114	Área Matemática Intermedia 3	6
	0152	Física 2	6
	0010	Lógica	1
	0011	Idioma Técnico 4 *	3
5to.	0123	Matemática Aplicada 5	5
	0118	Matemática Aplicada 1	5
	0154	Física 3	6
	0734	Estadística 2 *	5
6to.	0736	Análisis Probabilístico	5
	0156	Física 4	6
	0022	Psicología Industrial *	3
	0120	Matemática Aplicada 2 *	5
	0662	Legislación 1 *	3
	0653	Ingeniería de Costos *	4

Continuación tabla 1.

Semestre	Código del curso	Nombre del curso	Cantidad de créditos
	0001	**Ética Profesional**	**2**
7mo.	0116	Matemática Aplicada 3 *	5
	0659	Administración de Recursos Humanos *	6
	0122	Matemática Aplicada 4 *	5
8vo.	0700	Ingeniería Económica 1 *	4
	0672	Fundamentos de Administración *	4
9no.	0706	Preparación y Evaluación de Proyectos 1 *	5
	0601	Investigación de Operaciones 1 *	6
10mo.	0288	Introducción al Estudio de Impacto Ambiental *	4

Nota. Pensum de estudios de la carrera de Ingeniería Electrónica. Elaboración propia, realizado con Microsoft Excel.

2.1.2. Electrotecnia

El área de Electrotecnia contiene cursos que presentan los fundamentos sobre electricidad y dispositivos electrónicos. El objetivo de estos cursos es darles el primer acercamiento a los estudiantes sobre temas de electricidad y electrónica, para que tengan las bases y los fundamentos suficientes para poder abordar con los conocimientos suficientes los cursos posteriores a estos. En estos cursos en donde se le presentan a los estudiantes los conceptos básicos de la electricidad, que son: corriente (I), voltaje (V) y resistencia (R), los cuales son de vital importancia durante toda la carrera.

Tabla 2.

Cursos del área de Electrotecnia

Semestre	Código del curso	Nombre del curso	Cantidad de créditos
5to.	0204	Circuitos Eléctricos 1	7
	0462	Electricidad y Electrónica Básica	6
6to.	0206	Circuitos Eléctricos 2	7
	0210	Teoría Electromagnética 1	6
7mo.	0230	Instrumentación Eléctrica	7
9no.	7913	Seminario de Investigación Electrónica	3

Nota. Cursos de la carrera de Ingeniería Electrónica. Elaboración propia, realizado con Microsoft Excel.

2.1.3. Electrónica

El área de Electrónica son todos aquellos cursos prácticos y teóricos que tienen como objetivo que el estudiante pueda comprender todas las áreas que conforman la electrónica. En la electrónica se ven involucrados distintos aspectos, como la electrónica analógica, electrónica digital, desarrollo de *software*, robótica, distintos tipos de comunicaciones y diseño de antenas.

Tabla 3.

Cursos del área de Electrónica

Semestre	Código del curso	Nombre del curso	Cantidad de créditos
2do.	0769	Introducción a la Programación de Computadoras	5
3er.	0991	Lenguajes de Programación Aplicados a la Ingeniería Eléctrica	5
6to.	0232	Electrónica 1	6

Continuación tabla 3.

Semestre	Código del curso	Nombre del curso	Cantidad de créditos
7mo.	0242	Comunicaciones 1	6
	0246	Electrónica 3	6
	0240	Electrónica 2	6
8vo.	0211	Teoría Electromagnética 2	5
	0980	Proyectos de Computación Aplicados a Ingeniería Electrónica	5
	0244	Comunicaciones 2	6
	0248	Electrónica 5	6
	0234	Electrónica 4	6
9no.	0245	Comunicaciones 3	6
	0241	Radiocomunicaciones Terrestres	5
	0233	Electrónica Aplicada 1	4
	0249	Electrónica 6	6
	0236	Sistemas de Control 1	5
10mo.	0969	Telecomunicaciones y Redes Locales	6
	0243	Comunicaciones 4	6
	0235	Robótica	7
	0239	Electrónica Aplicada 2	4
	0209	Instalación de Equipos Electrónicos	6
	0237	Sistemas de Control 2 *	5

Nota. Cursos de la carrera de Ingeniería Electrónica. Elaboración propia, realizado con Microsoft Excel.

2.1.4. Potencia

El área de Potencia son todos aquellos cursos prácticos y teóricos que tienen como objetivo que el estudiante analice y comprenda todos los aspectos de la electricidad. La electricidad difiere de la electrónica en que se manipulan señales de alto voltaje o amperaje, y por ello es importante conocer a detalle todas las características de este tipo de señales.

Tabla 4.

Cursos del área de Potencia

Semestre	Código del curso	Nombre del curso	Cantidad de créditos
7mo.	0212	Conversión de Energía Electromecánica 1 *	7
8vo.	0218	Líneas de Transmisión	5
9no.	0214	Maquinas Eléctricas	7
10mo.	0238	Automatización Industrial	6

Nota. Cursos de la carrera de Ingeniería Electrónica. Elaboración propia, realizado con Microsoft Excel.

2.1.5. EPS

El área de EPS, siglas que corresponden a Ejercicio Profesional Supervisado, y contiene cursos prácticos y teóricos cuyo objetivo es que los estudiantes puedan aplicar los conocimientos adquiridos en toda la carrera para poder diseñar e implementar soluciones a distintos problemas de ingeniería.

Dentro de estos cursos se encuentran los cursos de prácticas, donde el estudiante aplica sus conocimientos en ámbitos fuera de lo académico, en donde para cursar dichos cursos se deben aprobar requisitos previos, como cumplir con cierta cantidad de créditos y tener aprobados ciertos cursos. Estos cursos están complementados con el curso de seminario, en donde adicionalmente de la práctica se le enseñan al estudiante todos los procesos que intervienen en la realización del trabajo de graduación y de la graduación en la carrera para optar a merecer el título de Ingeniero Electrónico.

Tabla 5.

Cursos del área de EPS

Semestre	Código del curso	Nombre del curso	Cantidad de créditos
3er.	2025	Prácticas Iniciales	-
7mo.	2036	Prácticas Intermedias	-
9no.	2013	Prácticas Finales Ingeniería Electrónica	-
	7988	Seminario de Investigación EPS Electrónica	3

Nota. Cursos de la carrera de Ingeniería Electrónica. Elaboración propia, realizado con Microsoft Excel.

2.1.6. Perfil de egreso de los estudiantes

El egresado deberá poseer los siguientes atributos y características: en el campo cognitivo en el área de formación general, teniendo así incidencia en la problemática social, así como también, un amplio conocimiento en al menos un segundo idioma. Tendrá un amplio conocimiento en el área fundamental de la electrónica, las telecomunicaciones, automatización industrial, sistemas de control, circuitos digitales, así como también una base en los principios de robótica.

Poseerá formación en ciencia, ingeniería, además de conocimientos multidisciplinarios, dándole capacidad e iniciativa para impulsar nuevas fuentes de trabajo en el campo de su competencia, además de poder conocer y utilizar tecnologías de punta para su implementación en el ámbito laboral.

Trabajará en el área comercial e industrial, especialmente en los sistemas automatizados de control de procesos industriales. También en el desarrollo de

las telecomunicaciones, incluyendo la planificación, el desarrollo y la supervisión de proyectos de transmisión de señales de radio, UHF, conducción por fibra óptica o cable a altas velocidades, y sistema y servicios de telefonía, radio y televisión.

3. MARCO TEÓRICO

3.1. Sistema Operativo GNU/LINUX

El acrónimo GNU significa recursivamente GNU's Not Unix. Se seleccionó este nombre porque GNU fue modelado a partir de Unix y conserva su arquitectura similar a Unix, aunque se diferencia de Unix en dos formas importantes. En primer lugar, GNU es *software* libre y, en segundo lugar, GNU no posee ningún código Unix.

3.1.1. Introducción

GNU/LINUX comúnmente conocido como Linux, actualmente es uno de los sistemas operativos que existen más confiable y que ofrece un buen rendimiento. GNU/LINUX ha demostrado tener muy buenas características, y un muy buen funcionamiento lo que hace que este sistema operativo esté al mismo nivel que los otros sistemas operativos existentes. Es un sistema operativo de código abierto basado en el núcleo (kernel) Linux y en las herramientas del proyecto GNU. Este sistema operativo es conocido por su estabilidad, flexibilidad, seguridad y capacidad de personalización. El término GNU/LINUX se utiliza para enfatizar la colaboración entre el núcleo Linux y las herramientas y utilidades desarrolladas por el proyecto GNU.

3.1.1.1. GNU

El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre y de código abierto, basado en

conceptos similares a Unix. Las herramientas desarrolladas por GNU incluyen el Shell (interfaz de línea de comandos), compiladores (como GCC), bibliotecas y utilidades diversas (como el editor de texto Emacs).

A principios de los años setenta las compañías no se centraban en el *software* de los dispositivos, ya que la gran mayoría de todas estas compañías fabricaban ordenadores de donde obtenían la mayor parte de los ingresos en la venta y distribución de máquinas, en donde a estas máquinas para su funcionamiento tenían incluido un sistema operativo junto con aplicaciones. En esta época instituciones como las universidades podían utilizar el código fuente de los sistemas operativos para estudios y para fines de enseñanza. Todos estos usuarios podían solicitar el código fuente de drivers y distintos programas para poder realizar modificaciones y así estos quedaran con un funcionamiento específico para las necesidades que tenían en ese momento. El *software* por sí solo no tenía ningún valor, ya que tenía que estar junto al *hardware* para que en conjunto tuviesen el funcionamiento esperado, y en este entorno los laboratorios Bell (AT&T) fueron los que lograron diseñar un sistema operativo funcional llamado UNIX, que tenía como características principales su optimización de recursos, al igual que era un sistema operativo bastante estable y que era compatible con *hardware* de distintos fabricantes.

Estas características de este sistema operativo lograron que las empresas más grandes de esa época empezaran a darle el valor que merece el *software*, ya que se tenían maquinas con características muy buenas, pero no tenían *software* para optimizar los recursos de estas máquinas. En el año de 1965 IBM dejó de dar el código fuente de su sistema operativo, y en los siguientes años otras empresas empezaron a vender sus propios sistemas operativos. Esto causó que todas las compañías empezaran a sacar ventaja del *software* de sus máquinas y empezaron a poner restricciones en sus equipos y a vender distintos

programas que no venían incluidos en el *hardware*. En medio de todos estos hechos Richard Stallman, indignado al ver que conseguir el código fuente de los programas era muy complicado, empezó un proyecto para poder lograr que el código fuente fuera otra vez abierto, aunque ya que lograr que las compañías aceptaran esto era muy complicado, decidió crear su propio sistema operativo en conjunto con su propio paquete de aplicaciones que llamó GNU. Stallman explicó a toda la comunidad todo con respecto a su proyecto y el porqué de este. El empezó a describir el concepto de *software* libre, y explicó las razones de por qué era una necesidad que distintos programadores y desarrolladores pudieran empezar a contribuir con este proyecto. En ocasiones se confunden los términos de *software* libre y *software* gratuito.

Se debe entender que el *software* libre no tiene que ser gratuito, sino que se debe entender que el *software* libre son programas de los cuales se puede conseguir el código fuente, en donde este se puede analizar, modificar y también se puede redistribuir sin que estemos obligados a pagar por ello.

El concepto de código abierto implica que el código fuente del *software* está disponible públicamente. Esto permite a los usuarios y desarrolladores examinar el código, corregir errores, realizar mejoras y distribuir versiones modificadas. La licencia más comúnmente asociada con el *software* GNU/LINUX es la licencia pública general de GNU (GPL), que garantiza la libertad para usar, estudiar, modificar y compartir el *software*.

La naturaleza de código abierto del sistema operativo ha fomentado una comunidad global de colaboradores y desarrolladores voluntarios. Esto ha llevado a la creación de distribuciones, herramientas y aplicaciones diversas que abarcan desde el *software* de servidor hasta las interfaces gráficas de usuario y las aplicaciones de productividad.

3.1.1.2. Núcleo LINUX

El núcleo de Linux es el corazón del sistema operativo GNU/LINUX. Actúa como intermediario entre el *hardware* y el *software*, gestionando recursos como memoria, procesadores, dispositivos de entrada/salida y comunicaciones entre programas. El núcleo se compone de módulos que pueden cargarse y descargarse dinámicamente, lo que permite una configuración más específica para el *hardware* y un mejor rendimiento.

El desarrollo del núcleo de Linux es liderado por Linus Torvalds y una comunidad global de desarrolladores que colaboran en su mejora. El modelo de desarrollo abierto permite la revisión y mejora constante del código, lo que resulta en mejoras de rendimiento, seguridad y compatibilidad con el *hardware* en cada nueva versión.

3.1.1.3. GNU/LINUX

Un profesor de la Universidad de Holanda llamado Andrew Tanenbaum, tomó la decisión de escribir un sistema operativo para que todos los estudiantes pudiesen utilizarlo para estudiar. De igual manera que Stallman, en ese momento solo habían logrado utilizar el código fuente del UNIX de AT&T. Su idea era poder diseñar un sistema operativo que pudiera ser objeto de estudios y también que pudiese ser modificado por cualquier persona. En el año de 1987 fue donde diseñó un proyecto que llamó mini UNIX, que dio lugar a MINIX. Ya que no utilizo ninguna línea de código de UNIX de AT&T, este sistema era prácticamente libre y no tenía ninguna restricción para poder utilizar su código.

Tanenbaum tenía como fin crear un sistema que solo tuviera fines académicos, y para ellos utilizó la arquitectura Microkernel, que tenía como

característica una fácil comprensión. Este sistema era muy útil para fines docentes, pero no tenía muy buenas características para poder ser utilizado en máquinas que se distribuían a todas las personas.

En el año de 1991 un estudiante de la universidad de Helsinki llamado Linus Torvalds decidió crear un propio núcleo para un nuevo sistema operativo, que era Linux. La idea de Torvalds era poder crear un UNIX para PC que tendría la característica de que cualquier persona lo podría utilizar en su ordenador.

En los primeros años de la existencia de GNU/LINUX, este sistema se podía identificar como un sistema operativo para los hackers. Era demasiado complicado poder instalar este sistema operativo, al igual que era bastante complicado manipularlo y tenía una deficiencia con los drivers. Todas estas cosas hacían que este sistema operativo fuera muy difícil de utilizar, por lo que para que una persona lo utilizara, tenía que ser experta en el tema. Fueron los primeros usuarios de este sistema operativo los que empezaron a diseñar drivers para los distintos accesorios que se utilizaban, como impresoras, tarjetas, entre otros. Y gracias a estas personas fue que este sistema se empezó a conocer cada vez más, hasta llegar a la actualidad en donde ya existen muchos usuarios que están desarrollando sus propias distribuciones de GNU/Linux.

3.1.1.4. Distribuciones

En la actualidad, hay muchas distribuciones que están basadas en el sistema operativo GNU/LINUX. Esto se da por la naturaleza del *software* libre, que permite que cualquier persona pueda utilizar el código que se ha desarrollado al momento y pueda realizar modificaciones para que el sistema funcione para sus propias necesidades.

Las distribuciones de Linux son variantes del sistema operativo que incluyen el núcleo Linux, las herramientas GNU y otros componentes como controladores de *hardware*, sistema de administración de paquetes y entornos gráficos. Cada distribución puede tener un enfoque y objetivos específicos. Por ejemplo, Ubuntu se enfoca en la usabilidad y la experiencia de usuario, mientras que CentOS se orienta a la estabilidad y la compatibilidad con servidores. Las distribuciones también pueden variar en cuanto a la selección de *software* preinstalado, el sistema de administración de paquetes (como APT o RPM) y las configuraciones por defecto.

Entre las distribuciones más conocidas de GNU/LINUX se encuentran las siguientes:

- Ubuntu
- Debian
- Fedora
- CentOS
- Red Hat
- Arch Linux
- Linux Mint
- Kali Linux

3.1.2. Conceptos y comandos básicos

En el siguiente inciso se describen los comandos básicos de programación.

3.1.2.1. Introducción

Un comando es un programa que tiene como función realizar una tarea determinada que se relaciona con el sistema operativo.

Cada distribución llega a tener sus propias aplicaciones, tanto para administrar el sistema como para poder gestionar el sistema operativo.

Cada uno de los comandos del sistema llega a tener muchos parámetros distintos. Al utilizar los parámetros se puede, con un mismo comando, hacer muchas acciones distintas. Los parámetros son opciones determinadas de cada comando, que se pueden añadir después del comando con un espacio intermedio o en algunas ocasiones en lugar del espacio se coloca un guion.

3.1.2.2. Usuarios y grupos

En la actualidad la gran mayoría de los sistemas operativos tienen las características de ser multiusuario y multitarea. Esto significa que en un mismo sistema operativo pueden trabajar varias personas en simultaneo, en donde cada usuario puede ejecutar distintas tareas al mismo tiempo. Por esta razón es importante que el sistema operativo permita controlar a los usuarios, utilizando un sistema de ingreso al sistema (*login*), al igual que también debe dejar controlar los programas que cada usuario puede utilizar, y todos los mecanismos de seguridad para que el *hardware* del equipo pueda estar seguro.

Los sistemas operativos que están basados en UNIX tienen toda esta información organizada por usuarios y por grupos. Cuando se ingresa al sistema, cada usuario se debe identificar con un *login* y su respectiva contraseña.

En todos los sistemas operativos debe existir un super usuario (*root),* que tendrá todos los permisos del sistema operativo para poder realizar cualquier acción dentro del sistema.

3.1.2.3. El sistema de ficheros

Una característica muy importante es que los sistemas de tipo UNIX manejan todos los dispositivos que están conectados al equipo como si se tratara de ficheros. Esto permite aprovechar todos los mecanismos y funciones que se utilizan con los ficheros, y se aplican a todos los dispositivos. En el directorio dev/ se encuentran todos los dispositivos que el sistema reconoce.

3.1.2.3.1. La jerarquía del sistema de fichero

Todos los sistemas operativos deben guardar una gran cantidad de archivos, desde los archivos de configuración del sistema, los de log, los de los usuarios, entre otros. Cada sistema operativo utiliza su propio sistema de ficheros. GNU/LINUX tienen la capacidad de poder leer y escribir archivos con cualquier sistema de ficheros que existe en la actualidad.

Todos los sistemas operativos basados en UNIX tienen la característica de que todos los dispositivos del sistema operativo pueden tratarse como si fueran ficheros.

Se debe tener claro que todos los sistemas de ficheros tienen origen de una misma raíz. Para que todos los ficheros estén organizados, el sistema proporciona directorios, en donde estos directorios pueden ser carpetas donde

se pueden almacenar más archivos y más directorios. De esta manera se pueden tener todos los ficheros ordenados de una forma jerárquica.

3.1.2.3.2. Sistema de directorios de Linux

La mayoría de todos los sistemas operativos actualmente siguen el estándar FHS, en donde están especificadas todas las características que debería tener cualquier sistema operativo. Una de las características es la distribución de directorios que se debe tener para que los ficheros estén bien organizados. La mayoría de las distribuciones basadas en GNU/LINUX siguen estas recomendaciones. Entre los principales directorios se encuentran los siguientes:

- *Root* (/): es el directorio base del sistema operativo, donde se encuentran también los archivos de usuario y programas.
- /bin: binarios (ejecutables) esenciales de los programas (del sistema) que se comparten entre múltiples usuarios. Los comandos de CLI están acá (cp, pwd, df, touch, bash, entre otros.).
- /boot: archivos de configuración para iniciar (bootear) el sistema operativo y archivos del Kernel.
- /cdrom: existe por retrocompatibilidad. Originalmente, acá se montaba la unidad de CD-ROM. Ya no es utilizada en computadoras que actualmente tienen unidad óptica.
- /dev: acá se montan todos los dispositivos físicos (y virtuales) de entrada y salida del sistema. Discos duros, dispositivos USB, unidades ópticas, puertos seriales, tarjetas de red, entre otros.
- /entre otros: archivos de configuración general de aplicaciones y del sistema. Configuración del GRUB (gestor de arranque), configuración de

red, servicios del sistema (apache, mosquito), bluetooth, entre otros. Estos archivos de configuración, normalmente se editan utilizando un editor de texto (GUI o CLI).

- /home: acá se encuentran los archivos personales de cada usuario. Existen sub-directorios para cada usuario. También hay configuraciones específicas del sistema operativo para cada usuario.
- /lib: librerías esenciales para la ejecución de aplicaciones.
- /lost+found: si el sistema operativo falla en su ejecución, los archivos corruptos serán movidos a esta carpeta.
- /media: acá se montan los dispositivos de almacenamiento que están enumerados en la carpeta /dev como discos duros, memorias USB, CD-ROOM.
- /mnt: existe por retrocompatibilidad. Acá se montaban los dispositivos de almacenamiento externo antes.
- /opt: *software* no esencial (opcional). Acá se instala el *software* que no está diseñado nativamente para Linux.
- /proc: contiene archivos que contienen el comportamiento de procesos de ejecución en el sistema.
- /root: es el home Administrador del sistema. Análogo a la ruta /home/root.
- /run: relativamente nuevo. Acá se encuentran archivos de intercambio de información en tiempo de ejecución.
- /sbin: similar a /bin. Contiene ejecutables escenciales del sistema operativo con comandos extra (como iw, reboot, fdisk).
- /srv: archivos de servicios de usuario. Páginas web de Apache por ejempo.
- /tmp: archivos temporales.
- /usr: es el análogo a /bin, pero para aplicaciones de usuario, no del sistema.
- /var: archivos variados, que no entran en ninguna otra categoría. Es común encontrar los logs acá.

3.1.2.3.3. Permisos de usuario

Cada usuario puede acceder a su carpeta personal y modificar archivos y directorios ahí dentro.

- /home/usuario1
- /home/usuario1/documentos/libros
- /home/usuario1/escritorio

Para realizar cambios al sistema o acceder a carpetas de otros usuarios, se necesitan credenciales elevadas (super usuario). Es necesario para instalar aplicaciones en el sistema o realizar cambios que puedan afectar a otros usuarios. Evita realizar cambios peligrosos por error o malintencionados.

El modo de acceso a credenciales elevadas depende de la distribución de Linux.

- Sudo (Ubuntu, Pop!, Fedora): concede permisos de super-usuario desde el usuario que lo solicita.
- Su – (Debian, Manjaro, Arch Linux): concede permisos de super-usuario, accediendo al usuario ‘root’.

3.1.2.4. Métodos abreviados de teclado

Permiten autocompletar instrucciones, nombres de archivos o directorios mientras se escribe el comando. Agilizan el uso de consola, una vez se tenga práctica.

Entre los más importantes se encuentran los siguientes:

- TAB: permite completar el nombre de ficheros, directorios o comandos, solo con escribir las primeras letras y presionar esta tecla.
- CTRL + L: limpia la pantalla.
- CTRL + D: sale del intérprete de comandos.
- CTRL + Z: pone el proceso actual en segundo plano suspendido.
- CTRL + C: elimina el proceso actual.
- CTRL + H: funciona igual que el retroceso.
- CTRL + A: va al principio de la línea.
- CTRL + W: elimina la última palabra escrita.
- CTRL + U: elimina todos los caracteres antes del cursor.
- CTRL + E: va al final de la línea.
- CTRL + T: cambia el orden de los últimos caracteres.
- SHIFT + REPAG: muestra media pantalla anterior.
- SHIFT + AVPAG: muestra media pantalla posterior.

3.1.3. Línea de comandos de Linux (CLI)

La línea de comandos de Linux, también conocida como terminal o Shell, es una interfaz de texto que permite a los usuarios interactuar con el sistema operativo mediante comandos escritos en lugar de acciones gráficas. Aunque las interfaces gráficas de usuario (GUI) son comunes en la información moderna, la CLI sigue siendo una herramienta esencial para administradores de sistemas, desarrolladores y usuarios avanzados. La CLI proporciona un nivel de control más profundo y automatización de tareas que no siempre es posible con una GUI.

El formato de un comando desde CLI es: <comando> argumentos –opciones o también <comando> --opciones argumentos.

3.1.3.1. Porque CLI

- No hay carga de interfaz gráfica (GUI).
- En teoría, cualquier tarea puede realizarse desde CLI.
- Pueden programarse tareas automatizadas y en serie.
- Puede iniciarse sesión remota a través de computadoras en red.
- Pueden iniciarse aplicaciones con interfaz gráfica desde CLI.
- El trabajo es mucho más ágil, una vez se tenga práctica con CLI.
- Comandos avanzados que no pueden realizarse en GUI, ser realizan desde CLI.

3.1.3.2. Comandos básicos

Los comandos son instrucciones que se escriben en la CLI para realizar tareas específicas, como crear archivos, copiar directorios, listar archivos, entre otros.

Algunos de los comandos básicos son los siguientes:

- Pwd: devuelve el directorio actual.
- Cd: cambia el directorio actual. Puede ser relativo o absoluto.
- Ls: muestra el archivo y carpetas del directorio actual.
- Mkdir: crea un nuevo archivo en la dirección actual.
- Cp: copia un archivo o directorios.
- Mv: mueve archivos o directorios. También utilizado para renombrar. Con este comando hay que tener cuidado si se usa junto a SUDO.
- Rm: remueve (borra) archivos o directorios.
- Touch: crea uno (o varios) archivos vacíos en el directorio actual.

Algunos comandos para visualizar archivo sin modificarlos son los siguientes:

- Less
- Cat
- More

Algunos editores de texto en CLI son los siguientes:

- Nano (principiante)
- Vi/vim (avanzado)
- Emacs (avanzado)

3.2. Bases de datos

Las bases de datos son fundamentales en la gestión de información, permitiendo el almacenamiento de datos en una variedad de aplicaciones.

3.2.1. Introducción a las bases de datos

Las bases de datos son una herramienta esencial en la era digital, facilitando la organización y acceso a grandes volúmenes de información.

3.2.1.1. Qué es una base de datos

Una base de datos es pocas palabras es un contenedor o almacén de información que se encuentra organizada y estructurada, donde estos datos deben tener alguna relación o vínculo entre ellos.

Para gestionar las bases de datos existen los gestores de base de datos o DBMS (Data Base Management System) que son programas que nos dejan guardar toda la información de una manera estructurada y también nos permite acceder a la información de una forma eficiente.

El DBSM nos deja realizar 4 operaciones que son las siguientes:

- Definición
- Recuperación
- Actualización
- Administración

3.2.1.2. Modelo de una base de datos

El modelo de una base de datos es el que determina cómo está estructurada la base de datos internamente. Esto se refiere a la manera en la que se van almacenando los datos. Algunos de los distintos modelos son los siguientes:

- Modelo relacional (este modelo está basado en tablas).
- Modelo dimensional (es una evolución del modelo relacional).
- Modelo orientado a objetos.
- Modelo de grafos.

3.2.1.3. Modelo relacional

Este modelo de base de datos es el modelo más extendido y su idea fundamental es el uso de las relaciones. Entre sus principales características se encuentran las siguientes:

- Todos los datos se representan por medio de registros y tuplas, y estos datos se agrupan dentro de relaciones.
- Las tablas tienen un conjunto de campos y atributos.
- Todas las conexiones de las tablas están establecidas mediante claves primarias y claves foráneas.
- Por medio de las claves primarias y claves foráneas garantizan la integridad de la información.

3.2.1.4. Elementos del modelo relacional

Existen varios elementos que confirman un modelo de bases de datos relacional. Entre ellas están los siguientes:

- Relaciones
- Dominios
- Procedimientos almacenados
- Restricciones
- Claves
- Estructuras

3.2.1.5. Modelo Entidad-Relación

Existen muchas herramientas para poder modelar los datos, entre ellas se encuentra el modelo entidad-relación, que permite representar todas las entidades relevantes para un sistema de información, al igual que sus relaciones y todas sus propiedades (atributos). Los elementos de un modelo de entidad-relación son:

- Entidad
- Atributo
- Relación

3.2.1.6. Diagrama de Entidad-Relación

Es la forma gráfica de poder representar el modelo entidad-relación, donde se plasman todas las relaciones de las entidades y los atributos, pero de una forma gráfica.

3.2.1.7. Pasar del modelo de entidad relación a las tablas

Del modelo de entidad-relación se tienen las entidades que luego se convierten en tablas. Los atributos de las entidades se convierten en columnas de la tabla. Las relaciones generalmente también se convierten en tablas. Las relaciones que contienen atributos siempre se convierten en tablas. Entre los distintos tipos de relaciones los más importantes son los siguientes:

- Relaciones uno a uno
- Relaciones uno a N
- Relaciones N a M

3.2.1.8. Normalización de las bases de datos

Existen una serie de reglas y de normas que se aplican al momento de diseñar una base de datos. Entre las más importantes se encuentran las siguientes:

- Evitar la redundancia de datos.
- Disminuir los problemas que se derivan de las actualizaciones de los datos en las tablas.
- Proteger la integridad de datos.

Todas estas reglas se denominan Formas Normales. Para cumplir las necesidades de la mayoría de las bases de datos, normalmente solo se necesitan cumplir las primeras tres formas normales.

3.2.2. Lenguaje SQL (Structured Query Language)

El lenguaje SQL es un lenguaje de acceso para las bases de datos que utilizan el modelo relacional.

3.2.2.1. Elementos del lenguaje SQL

Entre los elementos más importantes del lenguaje SQL se encuentran los siguientes:

- Base de datos (DATABASE)
- Tabla (TABLE)
- Campo (COLUMN, FIELD)
- Registro (REGISTER, ROW)
- Clave primaria (PK, PRIMARY KEY)
- Clave foránea (FK, FOREIGN KEY)
- Índice (INDEX)

3.2.2.2. Tipos de datos

Existen múltiples tipos de datos que se pueden insertar en una base de datos, entre los principales se encuentran los siguientes:

- Numéricos
- Fecha y hora
- Cadenas
- NULL

3.2.2.3. SQL DML (Data Manipulation Language)

Existen múltiples comandos para poder manipular toda la información que se encuentra en una base de datos, y para ello se cuenta con múltiples comandos que nos sirven para realizar distintas acciones. Entre los principales comandos se encuentran los siguientes:

- SELECT (Consulta)
- WHERE (Filtrado)
- GROUP BY (Agrupación y agregación)
- JOIN (Unión de tablas)
- HAVING (Filtrado de agregaciones)
- ORDER BY (Ordenación)
- UNION (Combinación)
- DISTINCT (Eliminación de duplicados)
- LIKE (Patrones)
- LIMIT/OFFSET (Paginación)
- INSERT (Inserción)
- UPDATE (Actualización)

- DELETE (Borrado)

3.3. Python

Python es un lenguaje de programación versátil y de alto nivel ampliamente utilizado en desarrollo de *software* y análisis de datos.

3.3.1. Qué es Python

Es un lenguaje de programación que fue creado por Guido van Rossum. Es un lenguaje de programación muy parecido a Perl, pero su sintaxis es más sencilla de comprender y es mucho más legible y entendible.

3.3.2. Por qué Python

Es un lenguaje con una sintaxis muy sencilla de entender. Tiene una capacidad de procesamiento de datos muy grande lo que permite desarrollar aplicaciones de una manera rápida y sencilla. No es un lenguaje de programación para bajo nivel.

3.3.3. Tipos básicos de variables

En este lenguaje de programación existen tipos básicos de variables que se dividen de la siguiente manera:

- Números, tanto enteros, flotantes y complejos.
- Cadenas de texto.
- Booleanos, que pueden ser verdadero o falso.

A diferencia de muchos lenguajes de programación, en Python no se debe declarar el tipo de variable al momento de que se está creando.

3.3.3.1. Números

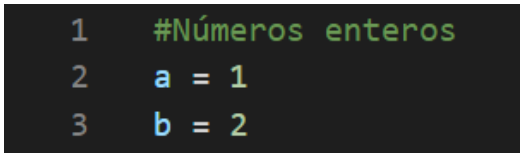
En Python los números se pueden representar como enteros, reales y complejos.

3.3.3.1.1. Enteros

Los números enteros con aquellos números que se pueden representar como positivos y negativos, y no deben tener decimal. En Python para poder representarlos mediante el tipo int o el tipo long. La diferencia entre el tipo int y el tipo long es que el tipo long permite almacenar números más grandes.

Figura 1.

Números enteros



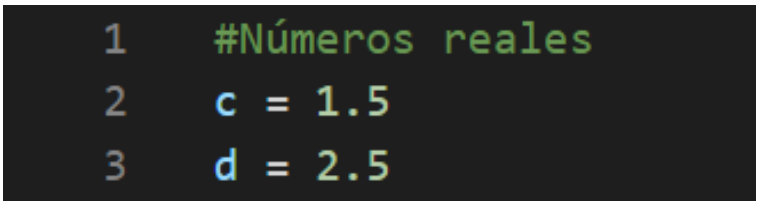
Nota. Ejemplo de números enteros. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

3.3.3.1.2. Reales

Los números reales son todos aquellos números que tienen decimales. En Python se pueden expresar mediante el tipo *floa*t. Al contrario que en otros lenguajes de programación en Python no se tiene el tipo *double*.

Figura 2.

Números reales



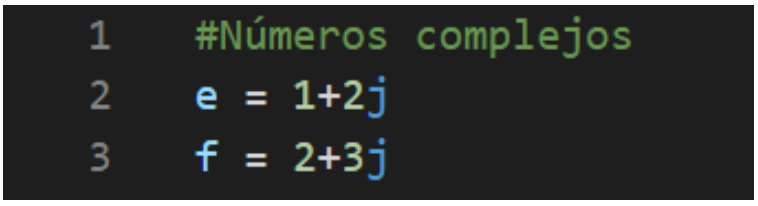
Nota. Ejemplo de números reales. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

3.3.3.1.3. Complejos

Los números complejos son los números que poseen una parte real y una parte imaginaria. Este tipo de números no es utilizado con mucha frecuencia, ya que sus aplicaciones son muy específicas por lo que no se requiere ser experto en el manejo de estos números.

Figura 3.

Números complejos



Nota. Ejemplo de números complejos. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

3.3.4. Variables tipo colección

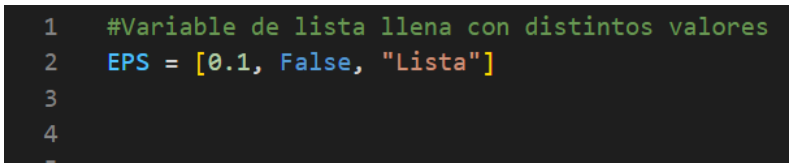
Existen distintos tipos de colecciones de datos. Entre los principales tipos de colecciones de datos se encuentran: listas, tuplas y diccionarios.

3.3.4.1. Listas

Las listas son un tipo de colección ordenado. Es el equivalente a los *arrays* y los vectores en otros lenguajes de programación. Las listas contienen todo tipo de datos como los números, cadenas, booleanos y listas.

Figura 4.

Lista



Nota. Colección ordenada. Elaboración propia, Realizado con Visual Studio Code versión 1.72.2.

3.3.4.2. Tuplas

Las tuplas son muy parecidas a las listas, con la diferencia que cuando se define la variable se utilizan paréntesis en lugar de los corchetes. Tienen la particularidad de que son inmutables, esto quiere decir que mientras se esté ejecutando el código no se puede cambiar o alterar su contenido.

Figura 5.

Tupla

```
1   #Tupla
2   EPS = (1, False, "Tupla")
3
4
```

Nota. Ejemplo de Tupla. Elaboración propia, Realizado con Visual Studio Code versión 1.72.2.

3.3.4.3. Diccionarios

Los diccionarios son colecciones que hacen la relación entre una llave y uno o muchos valores. Tienen la propiedad de que, al declarar una llave, el valor asociado a esa llave puede ser cualquier valor, incluso puede ser una lista, tupla o diccionario, y estas incluso pueden tener muchos valores dentro. Esta propiedad no solo aplica para los diccionarios, sino que también aplica para las listas y las tuplas.

Figura 6.

Diccionario

```
1   #diccionario
2   EPS = {'EPS1': 0.1, 'EPS2': 0.2, 'EPS3': 0.3}
3
4
```

Nota. Ejemplo de diccionario. Elaboración propia, Realizado con Visual Studio Code versión 1.72.2.

3.3.5. Control de flujo

El control de flujo en Python es fundamental para dirigir la ejecución de un programa, permitiendo tomar decisiones y repetir acciones.

3.3.5.1. Sentencias condicionales

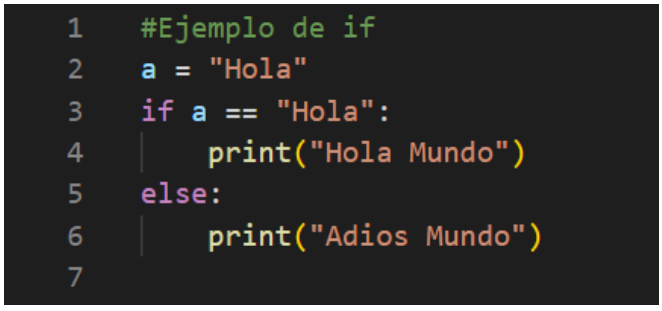
Las condicionales permiten comprobar distintas condiciones y hacen que los programas se comporten de distintas maneras, que ejecuten distintos códigos, dependiendo de la condición.

3.3.5.1.1. Sentencia condicional *If*

Es la sentencia condicional más simple, que permite evaluar una condición. Su estructura empieza con if, seguido de la condición a evaluar, dos puntos y en la siguiente línea se coloca el código que se debe ejecutar en caso de que se cumpla la condición que se está evaluando.

Figura 7.

Sentencia condicional If



Nota. Ejemplo de sentencia condicional. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

3.3.5.2. Bucles

Al contrario que las condicionales, los bucles permiten ejecutar múltiples códigos, dependiendo de distintas condiciones. Los bucles permiten que se ejecute un mismo fragmento de código un cierto número de veces, mientras se logre cumplir la condición determinada.

3.3.5.2.1. Bucle While

El bucle while permite ejecutar un fragmento de código mientras se cumple la condición establecida. Mientras la condición establecida no se llegue a cumplir, el clico no se finalizará. Hay que tener mucho cuidado en el uso de este ciclo dentro de distintos procesos ya que se puede llegar a caer en un ciclo infinito, lo que llevaría a que se consuma toda la memoria del equipo causando un error en el sistema.

Figura 8.

Bucle While

```
1  #Ejemplo de while
2  a = ""
3  while a != "EPS":
4      a = input("Escribe EPS: ")
5      print("Escribiste: ", a)
6
```

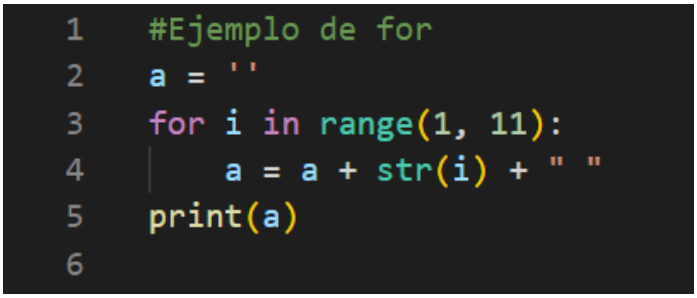
Nota. Ejemplo de bucle while. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

3.3.5.2.2. Bucle For

En Python este ciclo se utiliza para poder iterar sobre una secuencia. Es una estructura que permite ejecutar un bloque de código repetidamente un número específico de veces. Es especialmente útil cuando se sabe cuántas veces se desea que se repita un conjunto de instrucciones.

Figura 9.

Bucle For



Nota. Ejemplo de bucle For. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

3.3.6. Funciones

Una función es un fragmento de código al que se le da un nombre y que realiza distintas tareas, para luego devolver un valor. A los fragmentos de código que de igual manera se les asocia un nombre pero que no devuelven ningún valor, se les llama procedimientos. En el lenguaje de programación Python no existen los procedimientos.

Las funciones ayudan a optimizar el código ya que al utilizar funciones se puede reutilizar el código.

Figura 10.

Función

```
1   #Ejemplo de una funcion
2   def EPS(proceso1, proceso2):
3       print(proceso1)
4       print(proceso2)
5
```

Nota. Ejemplo de función. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

3.3.7. Librerías para la ciencia de datos

La ciencia de datos es un área de la programación donde se aplican diferentes técnicas de matemática, estadística, biología, física, sociología, entre otras. Su misión es poder modelar, analizar, visualizar y obtener toda la información posible a partir de los datos que se analizaron. El ciclo de este proceso de obtención de datos, es primero poder obtener una fuente de datos, donde no tiene importancia el origen de estos datos, luego se preparan, ya que no se pueden empezar a analizar si todos los datos se encuentran desordenados, luego de tener ordenados los datos para su análisis se procede a desarrollar un modelo específico para los datos que se tienen, y así poder analizar de manera correcta los datos para poder obtener los mejores resultados posibles, luego se deben evaluar todos los resultados obtenidos para poder publicarlos, se requiere una evaluación muy precisa ya que estos datos le servirán a otras personas para poder tomar decisiones por lo que los resultados deben ser lo más certeros posible.

Para realizar ciencia de datos se tienen múltiples lenguajes de programación, pero Python ofrece múltiples librerías que facilitan el trabajo, así también reúne todas las características necesarias para poder trabajar en ciencia de datos. Todas estas librerías *open source* conforman una gran herramienta matemática, estadística y de ciencia en general para poder trabajar con la ciencia de datos.

3.3.7.1. Matplotlib

Esta librería permite generar distintos tipos de gráficos para poder visualizar resultados de los datos previamente procesados. Tienen una interfaz orientada a objetos para que se pueda tener un control mucho más preciso de los resultados obtenidos.

3.3.7.2. Numpy

Esta librería es muy fundamental para la ciencia de datos y la computación científica. Entre sus principales características se encuentran las siguientes:

- Permite trabajar con objetos de tipo matriz de N-dimensiones.
- Se puede integrar con C/C++ y Fortran, lo cual expande el alcance de esta librería.
- Se pueden realizar cálculos de álgebra lineal, transformadas de Fourier, entre otros cálculos.

3.3.7.3. Pandas

Esta librería es específicamente para trabajar con análisis de datos, aunque también se pueden realizar otros trabajos utilizando esta librería. Esta

librería tiene múltiples usos tanto en el ámbito académico como en el área comercial, por ejemplo: finanzas, neurociencia, economía, estadística, publicidad, análisis web, entre otros.

Los principales datos que se pueden representar con esta librería son: datos tabulares y series temporales.

Entre sus principales características se encuentran las siguientes:

- Contienen herramientas para poder trabajar con la lectura y escritura de datos en archivos CSV, texto, Microsoft Excel, bases de datos SQL, entre otros.
- Contienen estructuras tabulares de datos, como DateFrame.
- Permite un mejor trabajo al utilizar la librería Numpy.
- Permite trabajar de forma más eficiente con las series temporales.
- Permite alinear datos de una manera inteligente.
- Permite seleccionar y filtrar de una manera muy simple tablas de datos en función de la posición, valor o etiquetas.
- Permite realizar gráficas.

3.3.7.4. Plotly

Esta librería está enfocada en los gráficos interactivos. Permite trabajar con más de 40 tipos de gráficos que se utilizan para representar datos estadísticos, financieros, geográficos, científicos y tridimensionales. Permite combinar datos de tipo numérico, con datos categóricos.

Cuenta con datos integrados, como data sets clásicos para poder realizar distintas pruebas. Entre ellos se encuentran los siguientes:

- Carshare
- Election
- Gapminder
- Iris
- Tips
- Wind

3.3.7.5. Scipy

Esta librería está basada en Numpy y tienen amplias herramientas para poder trabajar con el análisis de datos. Entre sus principales características se encuentran las siguientes:

- Múltiples funciones estadísticas.
- Permite trabajar con álgebra lineal.
- Permite trabajar con interpolación.
- Permite trabajar con algoritmos espaciales.
- Permite trabajar con el análisis de imágenes.

3.4. Visualización de datos

La visualización de datos es una disciplina clave en la ciencia de datos y la toma de decisiones, que permite presentar los datos de manera efectiva.

3.4.1. Visualización de datos

En la actualidad todos vivimos rodeados de datos que nuestro cerebro está procesando continuamente para poder comprender todo lo que nos rodea y poder tomar distintas decisiones sobre el futuro. En la actualidad todos los seres

humanos están expuestos a mucha información, y esto se debe a que cada vez se produce más información que proviene de distintos medios como las redes sociales y que cada vez es más sencillo tener acceso a esta información.

3.4.1.1. Visualización: explotación de los datos

La visualización de datos es la manera en la que podemos representar de forma gráfica la información o los datos de cualquier origen. La visualización es una herramienta que nos permite poder descubrir y poder comprender toda la lógica que conforma un grupo de datos.

Esto se puede visualizar por ejemplo con una tabla de datos que nos muestra en una columna los casos de COVID-19 y en la siguiente columna nos muestra el día, con lo cual tenemos una tabla que nos muestra los datos de COVID-19 por día. Solo viendo los datos podríamos pasar mucho tiempo viendo y analizando la información sin llegar a ninguna conclusión. En cambio, si procesamos estos datos mediante un lenguaje de programación, y realizamos distintos procesos a estos datos como regresiones lineales, podemos obtener gráficas donde podemos ver en el eje x los días, o el transcurrir del tiempo, y en el eje y los casos, con lo que obtendremos una gráfica que nos mostrará una curva de comportamiento del COVID-19 conforme pasan los días, y así podremos tener una predicción de cómo se estará comportando el virus a lo largo del tiempo. Es importante que en todo el proceso de tratamiento de datos se lleve un buen orden, ya que en este punto es donde ya se ve el resultado final, y normalmente estos resultados finales son los que se utilizan para tomar decisiones.

3.4.1.1.1. Tipos de visualización de datos

Existen muchas técnicas para poder visualizar los datos, pero desde el punto de vista de los datos se pueden resumir estás técnicas según la complejidad de los datos y la elaboración de la información. Algunas técnicas de visualización de datos básicos son los siguientes:

- Gráficas
- Mapas
- Tablas

3.4.1.2. Visualización de datos y Open Data

En los últimos años se han definido políticas y normas para empujar a los principales administradores de datos a que puedan hacer públicos sus datos a la sociedad, esto con el fin de hacer más transparente el uso de los datos y para poder utilizar toda esta información para poder mejorar múltiples procesos que se dan en el mundo todos los días.

También se han definido formatos de publicación y buenas prácticas para que estos administrados de datos no solo publiquen la información, sino que, al publicarla, esta información pueda ser entendible para cualquier persona y así el intercambio y acceso a la información sea más fácil y que cumpla con los estándares que faciliten la automatización de la reutilización del open data. Los datos abiertos se han convertido en una parte importante del mundo digital, ya que ayudan a promover la transparencia, la responsabilidad y la participación pública.

3.4.2. *Software* de visualización de datos

Todo el proceso para el análisis de datos es muy amplio, ya que se inicia desde la recolección de datos, y estos datos se pueden obtener de cualquier fuente, luego estos datos se deben procesar utilizando cualquier herramienta, y por último todos estos datos ya procesados se deben poder visualizar, y para esto existen múltiples herramientas de visualización de datos, donde estas herramientas nos deben permitir visualizar los datos de una manera clara y efectiva, para que los usuarios que hagan uso de estos resultados puedan sacar conclusiones de una manera muy sencilla.

3.4.2.1. Microsoft Excel

Esta es una herramienta que casi todas las personas saben utilizar, pero que no le sacan el máximo provecho.

En Microsoft Excel se pueden realizar múltiples tareas de ciencia de datos ya que tienen muchas herramientas y fórmulas que permiten trabajar de una manera sencilla. Pero para poder trabajar análisis de datos con Microsoft Excel se necesitan conocimientos avanzados para poder sacarle el máximo provecho.

Para el análisis de datos, en Microsoft Excel se pueden realizar múltiples acciones, entre las principales se encuentran las siguientes:

- Análisis de datos con tablas dinámicas.
- Creación de tablas dinámicas, tanto desde Microsoft Excel como desde distintas fuentes de datos.
- Permite trabajar con cubos OLAP desde distintas bases de datos multidimensionales SQL Server.

- Permite trabajar con múltiples filtros y segmentaciones.
- Permite realizar la importación de datos y crear distintas relaciones.

Para la visualización de datos, en Microsoft Excel se cuenta con las siguientes herramientas:

- Trabajo con distintos tipos de gráficos, como gráficos de barras, líneas, entre otros.
- Permite realizar gráficos dinámicos.
- Permite realizar mini gráficos.
- Power View

3.4.2.2. Power BI

Power BI es una aplicación que nos permite realizar conexiones con datos, al igual que nos permite manipular, transformar y visualizar los datos. Esta aplicación nos permite realizar conexiones con distintos orígenes de datos, y nos permite combinar todos estos datos en un modelo de datos, al cual se le pueden crear objetos visuales y recopilaciones de objetos visuales que se pueden compartir como informes.

Esta herramienta tiene múltiples usos, los cuales van a variar dependiendo de la persona que esté trabajando con esta herramienta. Entre los principales usos de Power BI se encuentran los siguientes:

- Conexiones de datos de distintas fuentes.
- Procesar y optimizar los datos para poder crear distintos modelos de datos.
- Crear objetos visuales, como los gráficos.

Una de las aplicaciones muy útiles para el análisis de datos de Power BI es la creación de informes. Ya que nos permite realizar estos informes a partir de distintas fuentes de información, lo cual nos permite ingresar información desde documentos de Microsoft Excel o directamente de una base de datos. Entonces podemos realizar la recolección de datos, luego procesar estos datos con Python, y al tener ya los datos procesados, podemos generar distintos documentos que pueden ser leídos por Power BI para poder crear un informe que nos muestre gráficas, tablas y otros tipos de elementos visuales para que las personas encargadas de analizar estos datos puedan tomar las mejores decisiones basándose en los resultados obtenidos.

4. DESARROLLO DE LAS PRÁCTICAS DEL CURSO DE PROYECTOS DE COMPUTACIÓN APLICADOS A LA INGENIERÍA ELECTRÓNICA

Las prácticas propuestas son una guía para el catedrático. En cada una se proponen temas, actividades y la guía para la práctica a realizar por los estudiantes, dichas prácticas pueden ser realizadas de manera individual o en grupo, esto será a criterio del catedrático encargado del laboratorio, al igual que el proyecto que se describe en otro capítulo, quedará a discreción del catedrático si será realizado en grupo o de manera individual.

4.1. Guía de laboratorio 1

- Inicio
  - Presentación del laboratorio

- Tema
  - Ciencia de datos en general

- Descripción
  - Presentar a los estudiantes el contenido del laboratorio de forma general, para que puedan tener un conocimiento superficial de los temas de los que tratará el laboratorio.

- Actividades propuestas
  - Realizar un resumen individual de cada estudiante, de cualquier distribución de Linux, que deberá incluir características, ventajas y

desventajas; dicho resumen deberá ser realizado en el periodo de clase. Se les debe recalcar a los estudiantes que es muy importante la elección de la distribución ya que en base a ella se harán el resto de las prácticas.

- Tarea
  - Instalación del sistema operativo que se investigó durante el periodo de clase, esto se podrá realizar con alguna máquina virtual, dual boot o directamente sobre su equipo como sistema principal. Se recomienda utilizar una distribución que no sea tan difícil de instalar y utilizar, ya que se utilizará durante todo el laboratorio.
  - Investigación de los comandos más utilizados en la consola de Linux.

4.2. Guía de laboratorio 2

- Tema
  - Trabajo con Linux desde línea de comandos.

- Descripción
  - Se introducirá a los estudiantes en el uso de la consola del sistema operativo Linux. Al igual que se mostrarán las ventajas de usar la línea de comandos antes que la interfaz gráfica.

- Actividades propuestas
  - Mostrar comandos específicos, por ejemplo, la creación de archivos, copiar y pegar desde línea de comandos, realización de búsquedas de archivos o palabras en distintos directorios.

- Realizar un ejercicio de niveles, en donde los estudiantes tendrán que ir encontrando las contraseñas para pasar al siguiente nivel, donde dichas contraseñas se encontrarán en el interior de distintos archivos que los estudiantes deberán ir encontrando. Para esto el catedrático lo deberá tener preparado. Se recomienda que esta actividad esté sobre alguna página o algún servidor en línea para que todos los estudiantes puedan trabajar al mismo tiempo.

- Tarea
  - Investigación del lenguaje de programación Python, y su sintaxis.

4.2.1. Práctica 2. Trabajo con Linux desde línea de comandos

- Nombre
  - Práctica de laboratorio: Creación de un módulo de Odoo

- Objetivos
  - Aplicar conocimientos de línea de comandos de Linux para la creación de una estructura de módulo de Odoo.
  - Familiarizarse con el uso de Git y GitHub para gestionar cambios y colaborar en proyectos.
  - Comprender la importancia de la estructura de carpetas y archivos en el desarrollo de *software*.

- Marco teórico
  - Línea de comandos en Linux: la línea de comandos es una interfaz textual que permite a los usuarios interactuar directamente con un sistema operativo. En Linux, esta herramienta es fundamental para realizar tareas de administración, gestión de archivos y ejecución

de programas. Algunos comandos esenciales incluyen 'cd' (cambiar directorio), 'mkdir' (crear directorio), 'ls' (listar archivos y directorios), 'cp' (copiar archivos), 'mv' (mover o renombrar archivos), 'rm' (eliminar archivos), entre otros.

- Git y GitHub: Git es un sistema de control de versiones distribuido que permite rastrear los cambios en el código fuente a lo largo del tiempo. GitHub es una plataforma de alojamiento de código en línea que utiliza Git y proporciona herramientas para la colaboración en proyectos de *software*. Los repositorios en GitHub almacenan versiones del código, facilitando la colaboración, el seguimiento de problemas y la revisión de código.
- Estructura de un módulo de Odoo: Odoo es un sistema de gestión empresarial modular. Cada módulo en Odoo sigue una estructura específica de carpetas y archivos para organizar su funcionalidad. Por ejemplo, la carpeta models almacena las definiciones de modelos de datos, la carpeta views contiene las interfaces de usuario definidas en XML, la carpeta static alberga recursos estáticos como archivos CSS y JavaScript, y la carpeta "data" contiene archivos de inicialización de datos.

- Investigación

Antes de comenzar con la práctica, es importante realizar las siguientes investigaciones:

- Instalación de Git: asegurarse de tener instalado Git en el sistema Linux. Si no se tiene instalado deben buscar la forma adecuada para instalarlo en la distribución de Linux que se esté utilizando. Por ejemplo, en Ubuntu se puede utilizar desde la consola utilizando el

comando ‘sudo apt-get install git’ para instalarlo. Leer cuidadosamente las instrucciones de GitHub para agregar un repositorio remoto a tu proyecto local. Esto asegurará que se puedan sincronizar los cambios entre el repositorio local y el remoto.

- Configuración de Git: antes de usar Git, se debe tener configurado el nombre de usuario y dirección de correo electrónico, para ello se pueden utilizar los comandos ‘git config –global user.name Tu nombre” y ‘git config –global user.email tu@email.com.
- Investigar en la documentación oficial de Odoo la estructura completa de un módulo personalizado de Odoo, tanto la estructura de las carpetas como la convención de nombres de carpetas y archivos.

- Descripción de la práctica

En esta práctica, los estudiantes crearán la estructura básica de un módulo de Odoo utilizando la línea de comandos de Linux y aprovecharán GitHub para gestionar los cambios en el código del módulo.

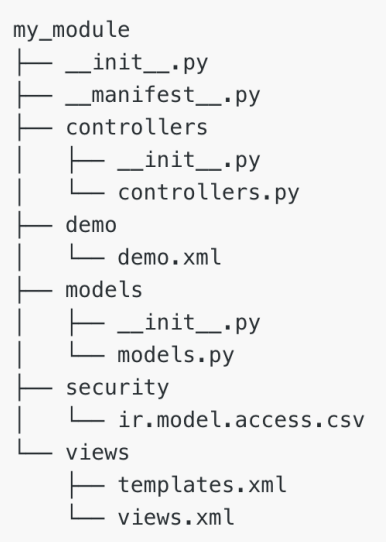
- Creación de la estructura del Módulo de Odoo:
  - Abre una terminal en tu sistema Linux.
  - Crea un directorio para tu proyecto de módulo de Odoo, donde se albergarán todos los archivos relacionados.
  - Utiliza el comando ‘cd’ para ingresar al directorio del proyecto recién creado.
  - Crea la estructura de carpetas siguiendo las convenciones de Odoo. Esto incluye directorios como *models, views, static, data*, entre otros.

- Crea los archivos dentro de las carpetas con los nombres establecidos en la documentación oficial de Odoo, el contenido no importa, solo es importante tener los archivos para que al subir las carpetas al repositorio no se tengan problemas.

- Inicialización del repositorio en GitHub:
  - Crea una cuenta en GitHub si aún no tienes una.
  - Crear un nuevo repositorio público en GitHub, utilizando como nombre del módulo nombre_apellido
  - Sigue las instrucciones de GitHub para agregar un repositorio remoto a tu proyecto local mediante los comandos proporcionados.
  - Utiliza los comandos *git add* y *git commit* para agregar todos los archivos y carpetas de tu proyecto al repositorio local.
  - Finaliza un push de tus cambios al repositorio en GitHub mediante el comando *git push.*
  - Deberá realizar varias veces el *push*, esto para que se tenga un registro detallado. Si es posible realizar un *push* para cada carpeta que se esté subiendo, de igual manera agregar un comentario por cada *push*. Debe dejar documentado cada *push* con un comentario detallado de la carpeta que se está subiendo a GitHub.
  - A continuación, se muestra un ejemplo de la estructura de un módulo de Odoo:

Figura 11.

Estructura de módulo



Nota. Ejemplo de estructura de módulo. Obtenido de Odoo docs. (s.f.). *Su primer módulo.* (https://www.odoo.com/documentation/15.0/es/administration/odoo_sh/getting_started/first_module.html), consultado el 09 de agosto de 2023. De dominio público.

- Formato de entrega de la práctica

  Los estudiantes deben entregar un informe que incluya lo siguiente:

  - Introducción que explique el propósito de la práctica y su importancia.
  - Descripción detallada de las tareas realizadas, indicando los comandos de la línea de comandos utilizados en cada paso.
  - Capturas de pantalla que ilustren el proceso de creación de la estructura del módulo y la conexión con el repositorio en GitHub.

- Enlace al repositorio de GitHub donde se encuentra el proyecto del módulo de Odoo.

- Fecha de entrega

La práctica, junto con el enlace al repositorio de GitHub, debe ser entregada antes de la fecha límite especificada por el catedrático.

4.3. Guía de laboratorio 3

- Tema
  - Programación en lenguaje Python

- Descripción
  - Presentar a los estudiantes la sintaxis y herramientas que tiene este lenguaje de programación, como lo son todos los tipos de variables y todos los tipos de estructuras lógicas que se pueden implementar.

- Actividades propuestas
  - Realizar distintos programas utilizando los distintos tipos de variables y los distintos ciclos en Python, para que el estudiante pueda comprender cómo se implementan y como se pueden solucionar problemas propuestos. Dichos programas podrán ser sobre cualquier tema, por ejemplo, resolución de matrices o resolución de series. Lo importante es que se pueda mostrar el funcionamiento en conjunto de todos los tipos de variables con los distintos ciclos y las sentencias condicionales.

- Actividad individual donde cada estudiante deberá resolver algún problema propuesto por el catedrático. Dicho problema también debe evaluar los temas explicados por el catedrático.

- Tarea
  - Investigación sobre las bases de datos y el lenguaje SQL.

4.3.1. Práctica 3. Programación en lenguaje Python

- Nombre
  - Práctica de laboratorio: Análisis de datos sobre la inteligencia artificial.

- Objetivos
  - Investigar y analizar cómo la inteligencia artificial (AI) está impactando el trabajo no solo de los desarrolladores de *software*, sino también cómo está afectando el trabajo de todas las áreas enfocadas a la tecnología e incluso en otros campos.
  - Familiarizar a los estudiantes con el uso de bibliotecas de Python como pandas, *matplotlib* y *plotly* para análisis y visualización de datos relacionados con la inteligencia artificial.

- Marco teórico
  - Python: Python es un lenguaje de programación versátil que es ampliamente utilizado en ciencia de datos y aplicaciones de inteligencia artificial. Su sintaxis sencilla y su rica biblioteca de módulos hacen que sea ideal para implementar algoritmos y realizar análisis complejos.
  - Bibliotecas para ciencia de datos y AI.

        - Pandas: biblioteca que ofrece estructuras de datos y herramientas para el análisis de datos en tablas.
        - Matplotlib: biblioteca para la visualización de datos en gráficos y gráficas.
        - Plotly: biblioteca que ayuda con la presentación de datos. Contiene al igual que *matplotlib* múltiples gráficos. Los gráficos de esta librería tienen la distinción de que son dinámicos, lo que le permite al usuario interactuar con las gráficas generadas.
    - Impacto de la inteligencia artificial en el desarrollo de *software*: la inteligencia artificial está revolucionando la forma en que los desarrolladores crean *software*. Desde la generación automática de código hasta la detección de errores, la IA está transformando la eficiencia y la precisión del proceso de desarrollo.

- Investigación

Antes de comenzar con la práctica, es importante realizar las siguientes investigaciones:

- Librerías: investigar las librerías de Python más utilizadas para la ciencia de datos.
- Tendencias en la automatización de desarrollo: investiga las tendencias actuales y futuras en la automatización del desarrollo con IA. Examina cómo la IA está afectando la colaboración entre desarrolladores y como las habilidades necesarias están evolucionando. Investigar también como la inteligencia artificial está afectando no solo a los desarrolladores, sino que también analizar

cómo está afecta a otras disciplinas como los abogados, contadores, entre otros.

- Descripción de la práctica

    - Descarga y preparación de los datos:
        - Descarga datos de prueba relacionados con la implementación de IA en el desarrollo de *software* o bien de la integración de la tecnología en la vida diaria de fuentes como Kaggle (por ejemplo, el conjunto de datos GitHub Repositories). Es importante descargar una buena fuente de datos para disminuir el trabajo de organizarlos.
        - Utiliza pandas para cargar los datos en un DateFrame y explorar su estructura.

    - Análisis de la influencia de la IA:
        - Utiliza pandas y matplotlib para realizar un análisis exploratorio de los datos descargados.
        - Ejemplo de código.

Figura 12.

Uso de matplotlib

```python
import pandas as pd
import matplotlib.pyplot as plt

# Carga de datos
data = pd.read_csv('ruta_del_archivo.csv')

# Análisis exploratorio
print(data.head())  # Muestra las primeras filas
plt.hist(data['tipo_de_dato'], bins=10)
plt.xlabel('Tipo de Dato')
plt.ylabel('Frecuencia')
plt.title('Distribución de Tipos de Dato')
plt.show()
```

Nota. Ejemplo de uso del matplotlib. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

- Evaluación de la Automatización:
  - Utiliza scikit-learn para construir un modelo de clasificación que determine si una tarea de desarrollo específica es candidata para la automatización con IA.
  - Ejemplo de código.

Figura 13.

Uso de scikit-learn

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# División de datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(data[['feature_1', 'feature_2']], data['target'], test_size=0.2)

# Creación y entrenamiento del modelo de clasificación
model = LogisticRegression()
model.fit(X_train, y_train)

# Evaluación del modelo
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print('Accuracy:', accuracy)
```

Nota. Ejemplo de uso de scikit-learn Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

- Implicaciones y análisis de resultados:
  - Escribe tu análisis que resuma las implicaciones positivas y negativas de la creciente automatización en el desarrollo de *software*.
  - Realiza un análisis sobre cómo los desarrolladores pueden beneficiarse de esta transformación y cómo deben adaptarse a los cambios en la industria.

- Formato de entrega de la práctica

  Los estudiantes deben entregar un informe que incluya lo siguiente:

  - Introducción que explique el propósito de la práctica y cómo se relaciona con el impacto de la IA en el desarrollo de *software*.
  - Descripción detallada de cada tarea realizada, incluyendo ejemplos de código utilizando las bibliotecas pandas, matplotlib y scikit-learn.
  - Gráficos y visualizaciones generadas durante el análisis.
  - Incluir los análisis realizados en la sección anterior.
  - Todos los gráficos, archivos de Python y el informe deben estar dentro de un archivo comprimido que se debe llamar carnet_nombre.zip, y debe estar todo bien identificado en carpetas. El informe debe estar en formato PDF.

- Fecha de entrega

  La práctica debe ser entregada antes de la fecha límite especificada por el catedrático.

4.4. Guía de laboratorio 4

- Tema
  - Bases de datos y lenguaje SQL.

- Descripción
  - Introducción teórica y práctica a las bases de datos, describiendo sus usos en múltiples ámbitos.

- Actividades propuestas
  - Explicar todas las características de las bases de datos, para poder comprender de manera correcta cómo están estructuradas.
  - Realizar hoja de trabajo donde cada estudiante deberá realizar un modelo de entidad relación del tema que asigne el catedrático.

4.4.1. Práctica 4. Base de datos y lenguaje SQL

- Nombre
  - Práctica de laboratorio: Diseño y administración de bases de datos y lenguaje SQL para inteligencia artificial.

- Objetivos
  - Introducir a los estudiantes en el diseño, creación y administración de bases de datos utilizando lenguaje SQL.
  - Familiarizar a los estudiantes con los conceptos fundamentales de las bases de datos y cómo pueden aplicarse en aplicaciones de inteligencia artificial.
  - Desarrollar habilidades prácticas en la creación, consulta y manipulación de bases de datos utilizando SQL.

- Marco teórico
    - Bases de datos y su importancia en inteligencia artificial: las bases de datos son elementos fundamentales para aplicaciones de inteligencia artificial, ya que almacenan y proporcionan acceso a los datos utilizados en la capacitación, prueba y despliegue de modelos de IA. La correcta estructuración y administración de bases de datos es crucial para garantizar un flujo de datos eficiente y preciso.
    - Lenguaje SQL: SQL (Structured Query Language) es un lenguaje de programación utilizado para administrar bases de datos. Permite definir, manipular y consultar datos en sistema de gestión de bases de datos relacionales (RDBMS) como PostgreSQL. Las sentencias SQL son utilizadas para crear tablas, insertar y actualizar datos, realizar consultas y más.

- Investigación

Antes de comenzar con la práctica, es importante realizar las siguientes investigaciones:

    - Realizar una investigación sobre los siguientes conceptos: SQL DML y SQL DDL.
    - Investigar sobre PostgreSQL y su diferencia con los otros *software*s para la gestión de bases de datos.

- Descripción de la práctica

    - Diseño conceptual de la base de datos:

- Diseña conceptualmente una base de datos que almacenará información relacionada con aplicaciones de inteligencia artificial. Define entidades, atributos y relaciones.
- Esboza un diagrama de entidad relación que represente la estructura de la base de datos.

- Creación de la base de datos:
  - Utilizando PostgreSQL, crea una base de datos llamada nombre_apellido.
  - Crea las siguientes tablas en la base de datos: Usuarios y Productos con columnas: ID (clave primaria), nombre, precio solo para la tabla de Productos.

- Inserción y manipulación de datos:
  - Inserta al menos 3 registros en la tabla Usuarios.
  - Inserta al menos 5 registros en la tabla Productos.

- Realiza consultas SQL para:
  - Recuperar el nombre de todos los usuarios.
  - Obtener el precio promedio de los productos.
  - Encontrar los productos con un precio mayor a Q50.

- Mantenimiento y actualización de las bases de datos:
  - Actualiza el precio de un producto específico en la tabla Productos.
  - Elimina un usuario de la tabla Usuarios según su ID.

- Análisis de datos:
  - Realiza un análisis sobre cómo los datos almacenados en la base de datos podrían utilizarse en aplicaciones de IA.
  - Considera qué tipos de datos serían útiles para entrenar modelos de IA y cómo podrían ser utilizados.

- Formato de entrega de la práctica

  Los estudiantes deben entregar un informe que incluya:

  - Una descripción del diseño del diseño conceptual de la base de datos y el diagrama de entidad relación.
  - Capturas de pantalla que muestren la estructura y contenido de las tablas creadas en PostgreSQL.
  - Ejemplos de las consultas SQL realizadas y sus resultados.
  - Incluir en una carpeta comprimida la base de datos creada luego de realizar todos los procedimientos.

- Fecha de entrega

  La entrega deberá realizarse antes de la fecha límite especificada por el catedrático.

4.5. Guía de laboratorio 5

- Tema
  - Bases de datos y Python.

- Descripción
    - Se introducirá a los estudiantes en el uso de Python para la lectura y modificación de datos de una base de datos.

- Actividades propuestas
    - Mostrar las librerías de Python que permiten trabajar en conjunto con bases de datos.
    - Realización de un examen de laboratorio, que cubrirá todos los temas desde el inicio del laboratorio hasta esta práctica. La estructura y metodología del examen quedan a criterio del catedrático del laboratorio.

- Tarea
    - Investigación sobre la combinación de Microsoft Excel con bases de datos y Python para poder mostrar valores procesados en gráficas de Microsoft Excel.

4.5.1. Práctica 5. Base de datos y Python

- Nombre
    - Práctica de laboratorio: Integración de bases de datos PostgreSQL con Python.

- Objetivos
    - Enseñar a los estudiantes cómo utilizar Python para acceder y manipular bases de datos PostgreSQL.
    - Proporcionar a los estudiantes una comprensión sólida de cómo las bases de datos pueden integrarse con aplicaciones de Python.

- Desarrollar habilidades prácticas en la escritura de scripts de Python para realizar consultas y modificaciones en bases de datos.

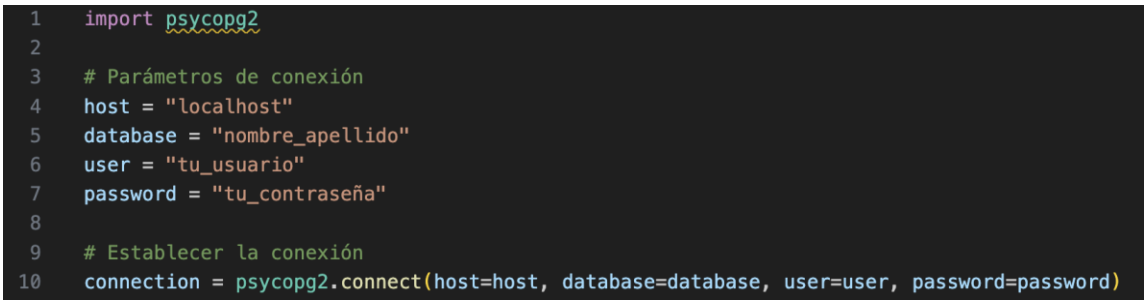
- Marco teórico
  - Integración de bases de datos y Python: la integración de bases de datos con Python permite a los desarrolladores acceder y manipular datos almacenados en bases de datos desde sus aplicaciones. Utilizando módulos y librerías específicas de Python, es posible conectar aplicaciones a sistemas de gestión de bases de datos como PostgreSQL y realizar operaciones de consulta y modificación de manera programática.

- Descripción de la práctica

  - Conexión a la base de datos:
    - Instala el módulo ‘psycopg2’ de Python, necesario para interactuar con bases de datos PostgreSQL.
    - Escribe un script en Python que establezca una conexión a la base de datos nombre_apellido que previamente creaste.
    - Ejemplo de código Python para establecer la conexión a la base de datos.

Figura 14.

Conexión a base de datos



```python
import psycopg2

# Parámetros de conexión
host = "localhost"
database = "nombre_apellido"
user = "tu_usuario"
password = "tu_contraseña"

# Establecer la conexión
connection = psycopg2.connect(host=host, database=database, user=user, password=password)
```

Nota. Ejemplo de conexión a base de datos. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

- Consulta SQL desde Python: Utilizando el script de Python realiza consultas SQL para:
  - Recuperar todos los archivos de la tabla.
  - Obtener los productos con un precio superior a Q50 de la tabla Productos. Asegúrate de ejecutar las consultas y mostrar los resultados en la consola.
  - Ejemplo de código Python para realizar las consultas SQL.

Figura 15.

Consultas SQL

```python
# Crear un cursor para ejecutar consultas
cursor = connection.cursor()

# Consulta para recuperar todos los usuarios
query_users = "SELECT * FROM Usuarios"
cursor.execute(query_users)
users = cursor.fetchall()

# Consulta para obtener productos con precio superior a $50
query_expensive_products = "SELECT * FROM Productos WHERE precio > 50"
cursor.execute(query_expensive_products)
expensive_products = cursor.fetchall()

# Mostrar los resultados
print("Usuarios:")
for user in users:
    print(user)

print("\nProductos caros:")
for product in expensive_products:
    print(product)

# Cerrar el cursor
cursor.close()
```

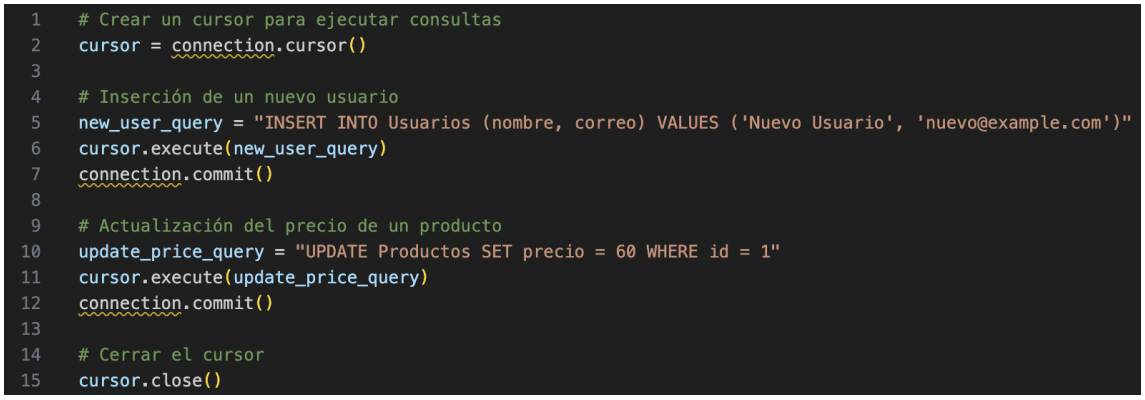
Nota. Ejemplo de consultas SQL. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

- Inserción y actualización de datos desde Python:
  - Modificar el script de Python para insertar un nuevo usuario en la tabla Usuarios.
  - Actualiza el precio de un producto en la tabla Productos utilizando Python.

- Ejemplo de código Python para inserción y actualización de datos.

Figura 16.

Inserción y actualización de datos



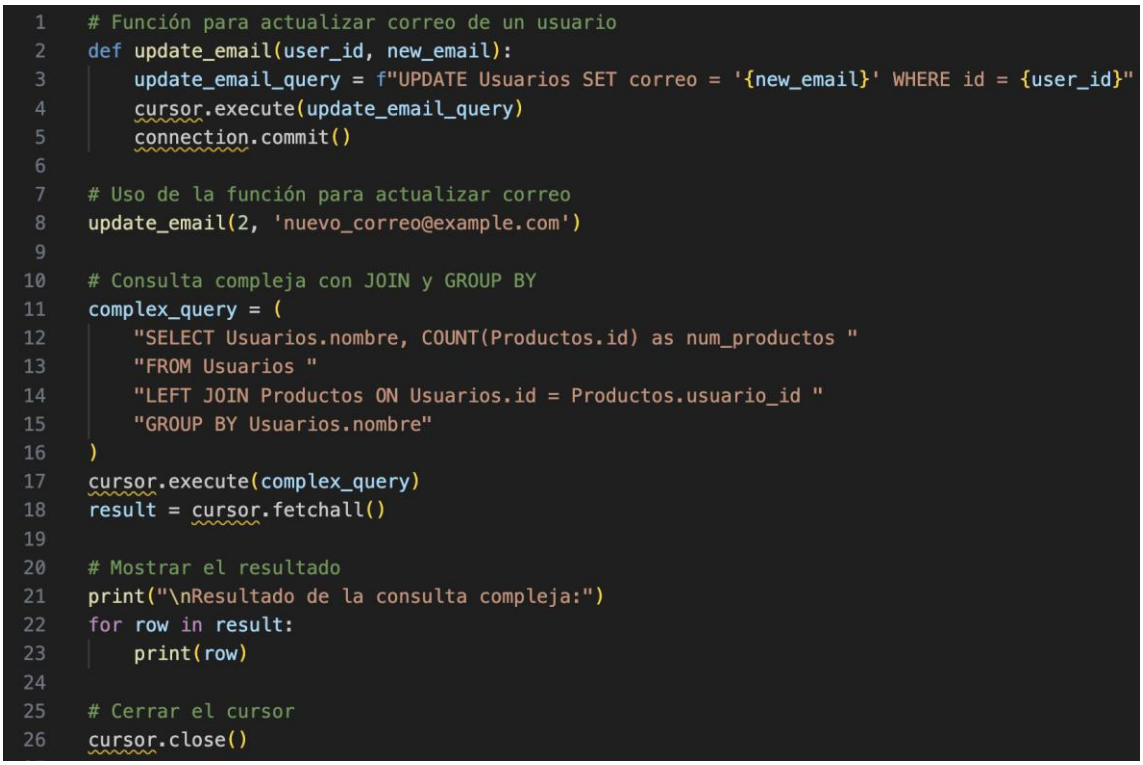
```python
# Crear un cursor para ejecutar consultas
cursor = connection.cursor()

# Inserción de un nuevo usuario
new_user_query = "INSERT INTO Usuarios (nombre, correo) VALUES ('Nuevo Usuario', 'nuevo@example.com')"
cursor.execute(new_user_query)
connection.commit()

# Actualización del precio de un producto
update_price_query = "UPDATE Productos SET precio = 60 WHERE id = 1"
cursor.execute(update_price_query)
connection.commit()

# Cerrar el cursor
cursor.close()
```

Nota. Ejemplo de inserción y actualización de datos. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

- Manipulación avanzada desde Python:
  - Escribe una función en Python que permita a un usuario modificar su dirección de correo electrónico en la tabla Usuarios.
  - Crea una consulta compleja que involucre JOIN y GROUP BY para obtener información detallada de usuarios y productos.
  - Ejemplo de código Python para manipulación avanzada.

Figura 17.

Manipulación avanzada



```python
# Función para actualizar correo de un usuario
def update_email(user_id, new_email):
    update_email_query = f"UPDATE Usuarios SET correo = '{new_email}' WHERE id = {user_id}"
    cursor.execute(update_email_query)
    connection.commit()

# Uso de la función para actualizar correo
update_email(2, 'nuevo_correo@example.com')

# Consulta compleja con JOIN y GROUP BY
complex_query = (
    "SELECT Usuarios.nombre, COUNT(Productos.id) as num_productos "
    "FROM Usuarios "
    "LEFT JOIN Productos ON Usuarios.id = Productos.usuario_id "
    "GROUP BY Usuarios.nombre"
)
cursor.execute(complex_query)
result = cursor.fetchall()

# Mostrar el resultado
print("\nResultado de la consulta compleja:")
for row in result:
    print(row)

# Cerrar el cursor
cursor.close()
```

Nota. Ejemplo de manipulación avanzada. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

- Análisis de datos:
  - Realiza un análisis sobre cómo la integración de bases de datos con Python puede ser útil en aplicaciones de inteligencia artificial.

- Formato de entrega de la práctica
  - Los estudiantes deben entregar un informe que incluya:
    - El código Python utilizado en cada parte de la práctica.

- Capturas de pantalla de la consola que muestren los resultados de las consultas y modificaciones realizadas desde Python.
- Incluir el análisis realizado en la sección anterior.
- Incluir en una carpeta comprimida todos los archivos de Python desarrollados para esta práctica.

- Fecha de entrega

La entrega deberá realizarse antes de la fecha límite especificada por el catedrático.

4.6. Guía de laboratorio 6

- Tema
  - Introducción a Microsoft Excel con Python.

- Descripción
  - Mostrar a los estudiantes las librerías de Python que permiten la manipulación y creación de archivos de Microsoft Excel.
  - Mostrar a los estudiantes las herramientas integradas de Microsoft Excel para crear gráficos partiendo de datos.

4.6.1. Práctica 6. Introducción a Microsoft Excel con Python

- Nombre
  - Práctica de laboratorio: visualización de datos desde bases de datos PostgreSQL con Python y Microsoft Excel.

- Objetivos
  - Enseñar a los estudiantes cómo utilizar Python para acceder a una base de datos PostgreSQL y proporcionar los datos a Microsoft Excel para su visualización.
  - Familiarizar a los estudiantes con la creación de gráficos y visualizaciones en Microsoft Excel a partir de datos obtenidos mediante Python.

- Marco teórico
  - Integración de datos y visualización: la integración de datos desde una base de datos de PostgreSQL a través de Python permite a los desarrolladores extraer información y proporcionarla en un formato amigable para su visualización. Microsoft Excel ofrece herramientas poderosas para crear gráficos y visualizaciones a partir de datos.
  - Microsoft Excel como herramienta de visualización de datos: Microsoft Excel es una potente herramienta de hoja de cálculo que va más allá de las simples operaciones matemáticas. Una de sus capacidades más destacadas es su capacidad para visualizar datos de manera efectiva y comprensible. Microsoft Excel ofrece diversas herramientas para representar datos en forma de gráficos y tablas, lo que facilita la identificación de patrones, tendencias y relaciones entre variables.
  - Gráficos: Microsoft Excel proporciona una variedad de tipos de gráficos, como gráficos de barras, gráficos de líneas, gráficos de dispersión y más. Estos gráficos permiten representar datos de manera visual, lo que facilita la interpretación y el análisis. Los gráficos se generan automáticamente a partir de los datos en las hojas cálculo de Microsoft Excel y se pueden personalizar para resaltar aspectos específicos.

- Integración con otras herramientas: Microsoft Excel se integra bien con otras herramientas de Microsoft, como PowerPoint y Word, lo que permite incrustar gráficos y tablas en documentos y presentaciones. Además, Microsoft Excel es compatible con muchas otras aplicaciones y lenguajes de programación, lo que facilita la importación y exportación de datos desde y hacia diferentes plataformas.

- Descripción de la práctica

  - Consultas y preparación de datos:
    - Utiliza el conocimiento para acceder a la base de datos PostgreSQL y obtener los datos necesarios.
    - Escribe un script en Python que realice las consultas necesarias y prepare los datos para su visualización.

  - Generación de archivo Microsoft Excel con Python:
    - Utiliza la librería ‘openpyxl’ de Python para crear un archivo de Microsoft Excel.
    - Escribe los datos obtenidos en el paso anterior en hojas de cálculo de Microsoft Excel.
    - Determina qué herramientas de la librería ‘openpyxl’ son las que se adaptan de mejor manera para la solicitud, para no tener problemas de compatibilidad por la extensión del archivo generado.
    - Ejemplo de código Python para generar un archivo de Microsoft Excel.

Figura 18.

Generación de archivo de Microsoft Excel

```python
import openpyxl

# Crear un nuevo archivo Excel
workbook = openpyxl.Workbook()

# Crear una hoja de cálculo
sheet = workbook.active
sheet.title = "Datos desde BD"

# Escribir los datos en la hoja
# Por ejemplo, asumiendo que "datos" contiene los datos obtenidos
for row in datos:
    sheet.append(row)

# Guardar el archivo Excel
workbook.save("datos_desde_bd.xlsx")
```

Nota. Ejemplo de generación de archivo de Microsoft Excel. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

- Visualización en Microsoft Excel:
  - Abre el archivo de Microsoft Excel generado en el paso anterior.
  - Utiliza las herramientas de Microsoft Excel para crear gráficos y visualizaciones a partir de los datos.

- Formato de entrega de la práctica

  Los estudiantes deben entregar un informe que incluya:

  - El código Python utilizado para acceder a la base de datos y generar el archivo de Microsoft Excel.

- El archivo de Microsoft Excel generado con los datos obtenidos y con las gráficas generadas.

- Fecha de entrega

La entrega deberá realizarse antes de la fecha límite especificada por el catedrático.

4.7. Guía de laboratorio 7

- Tema
  - Caso real de ciencia de datos.

- Descripción
  - Describir el proceso completo desde la obtención, procesamiento y visualización de datos. Dicho proceso es el mismo que han llevado los estudiantes al desarrollar todas las prácticas.

- Actividades propuestas
  - Dar a los estudiantes todos los parámetros para la realización del proyecto de ciencia de datos.
  - Realización del examen final, todos los temas vistos durante todas las prácticas.
  - Aclarar todas las dudas que puedan tener los estudiantes con respecto al proyecto del laboratorio.

5. PROYECTO DEL LABORATORIO

El proyecto del laboratorio desempeña un papel fundamental, ya que constituye un componente esencial para evaluar los conocimientos adquiridos por cada estudiante al final del laboratorio. Este proyecto no solo será un indicador de su comprensión y habilidades, sino que también proporcionará una oportunidad significativa para aplicar en la práctica los conceptos y las herramientas discutidas a lo largo de cada práctica. De esta manera, el proyecto del laboratorio se convierte en una ventana hacia el mundo real, donde los estudiantes pueden demostrar su capacidad para abordar problemas reales y desafiantes relacionados con la temática del curso. Este proyecto es una propuesta que puede ser modificada o reemplazado totalmente por el catedrático del laboratorio.

5.1. Nombre del proyecto

El nombre del proyecto es: *Impacto de la inteligencia artificial en el trabajo de distintos sectores*.

5.2. Introducción

En un mundo cada vez más impulsado por la tecnología, la inteligencia artificial (IA) está transformando la forma en que las empresas desarrollan *software* y crean soluciones innovadoras. Este proyecto tiene como objetivo explorar el impacto de la IA en el trabajo de los desarrolladores, desde la automatización de tareas hasta la integración de los datos y la toma de decisiones basada en análisis. A través de una serie de actividades que abarcan el uso de

la línea de comandos de Linux, programación en Python, manipulación de bases de datos y visualización de datos en Microsoft Excel, los estudiantes investigarán cómo la IA está remodelando el rol tradicional de los desarrolladores.

5.3. Objetivo

El objetivo principal de este proyecto es analizar y presentar el impacto de la inteligencia artificial en el trabajo de los desarrolladores. Los estudiantes realizarán actividades prácticas en diversas áreas, desde la automatización de tareas hasta la generación y análisis de datos, para comprender cómo la IA está influyendo en la forma en que se desarrollan y gestionan las aplicaciones.

5.4. Descripción del proyecto

Este proyecto tiene como objetivo principal ofrecer una comprensión profunda de los conceptos y técnicas aprendidas en el laboratorio.

5.4.1. Automatización de tareas con líneas de comandos (Linux)

Utiliza la línea de comandos de Linux para crear un script que organice automáticamente los archivos de un directorio específico en subdirectorios según su extensión. Por ejemplo, todos los archivos ‘.txt’ podrían moverse a un directorio llamado txt. Esto no es obligatorio y no se penalizará si se realiza un script para automatizar las carpetas, pero servirá si en algún otro punto de calificación obtiene una nota baja, se podrá utilizar esta parte para compensar ciertos puntos. Si se realiza deben tener en cuenta que se calificará el comportamiento completo para poder otorgar los puntos.

Figura 19.

Código en Bash

```bash
#!/bin/bash

# Directorio de origen
source_dir="/ruta/del/directorio"

# Iterar a través de los archivos en el directorio
for file in $source_dir/*; do
    # Obtener la extensión del archivo
    extension="${file##*.}"

    # Crear subdirectorio si no existe
    mkdir -p "$source_dir/$extension"

    # Mover el archivo al subdirectorio correspondiente
    mv "$file" "$source_dir/$extension"
done
```

Nota. Ejemplo del código en Bash. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

Se deberá tener un registro de esta automatización donde puedan demostrar que la optimización si está organizando los archivos.

5.4.2. Análisis de datos con Python

Adquirir una fuente de datos, no importando la fuente de los datos, lo importante es que se obtengan datos bien ordenados y estructurados para evitar el trabajo de ordenamiento.

Crear una base de datos a partir de los datos obtenidos. Es importante que queden bien especificadas todas las tablas y las relaciones. El *software* donde se debe crear la base de datos debe ser PostgreSQL.

Accede a la base de datos PostgreSQL utilizando Python y ejecuta consultas para obtener información sobre el impacto de la IA en el trabajo de los desarrolladores.

Ejemplo de código en Python:

Figura 20.

Acceso a datos con Python

```python
import psycopg2

# Conexión a la base de datos
connection = psycopg2.connect(
    database="nombre_basedatos",
    user="nombre_usuario",
    password="contraseña",
    host="localhost"
)

# Crear cursor
cursor = connection.cursor()

# Ejecutar consulta
query = "SELECT COUNT(*) FROM proyectos WHERE tecnologia='IA';"
cursor.execute(query)
result = cursor.fetchone()

# Cerrar cursor y conexión
cursor.close()
connection.close()

# Mostrar resultado
print("Número de proyectos relacionados con IA:", result[0])
```

Nota. Ejemplo de acceso a datos con Python. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

Luego realizar un análisis exploratorio de los datos para identificar tendencias y patrones relevantes. Por ejemplo, se puede calcular el porcentaje de crecimiento anual de proyectos relacionados con IA.

5.4.3. Integración de datos y visualización en Microsoft Excel

Utiliza Python para extraer los resultados del análisis y generar un archivo Microsoft Excel que contenga los datos relevantes.

Ejemplo de código en Python:

Figura 21.

Integrar datos con Microsoft Excel

```python
import openpyxl

# Crear archivo Excel
workbook = openpyxl.Workbook()
sheet = workbook.active

# Agregar encabezados
sheet.append(["Año", "Proyectos IA"])

# Agregar datos
data = [(2018, 150), (2019, 220), (2020, 300)]
for row in data:
    sheet.append(row)

# Guardar archivo Excel
workbook.save("resultados_proyectos.xlsx")
```

Nota. Ejemplo de integración de datos con Microsoft Excel. Elaboración propia, realizado con Visual Studio Code versión 1.72.2.

Luego crear un gráfico de línea en Microsoft Excel que represente el crecimiento anual de proyectos relacionados con IA.

5.5. Conclusiones y presentación

Elaborar un informe final que incluya los resultados de todas las actividades, junto con las capturas de pantalla correspondientes. De igual manera debe incluir todo el análisis de los resultados obtenidos y las gráficas generadas con Microsoft Excel. Debe preparar una presentación ya que la entrega se realizará mediante una presentación por estudiante hacia el catedrático del laboratorio.

5.6. Entrega

El proyecto debe ser presentado en un informe estructurado que incluya:

- Documentación detallada de las actividades realizadas en cada fase.
- Código fuente en cada actividad.
- Capturas de pantalla que muestren los resultados de las actividades.
- Una carpeta comprimida con todas las carpetas y archivos dentro de las carpetas, todo esto debe estar bien estructurado como se menciona en la primera sección del enunciado del proyecto.

5.7. Fecha de entrega

El informe final y los archivos del proyecto serán entregados en las fechas establecidas por el coordinador del laboratorio.

6. SEGUIMIENTO DE MEJORA

6.1. Atribuciones del personal docente

El personal docente designado para este curso será el encargado de guiar a los estudiantes, impartiendo clases tanto teóricas como prácticas para que los estudiantes puedan comprender de manera adecuada todos los temas vistos en el laboratorio. Dentro de las atribuciones del personal docente se encuentran las de capacitar a los estudiantes e informar todas las normas y los procedimientos que se deberán seguir en el transcurso del laboratorio.

6.1.1. Perfil del personal docente

Se requiere que el personal docente que se postule para impartir este laboratorio tenga amplios conocimientos en todos los temas del programa del laboratorio, al igual que se espera que esta persona tenga amplia experiencia en la ciencia de datos. Todas estas habilidades aportarán en gran manera a la formación académica de todos los estudiantes de la carrera de Ingeniería Electrónica que cursen este laboratorio.

CONCLUSIONES

1. Se presenta el diseño del laboratorio de Proyectos de Computación Aplicados a Ingeniería Electrónica, lo cual es un paso hacia la mejora de la calidad educativa para los estudiantes de ingeniería electrónica en la Escuela de Ingeniería Mecánica Eléctrica de la Facultad de Ingeniería.

2. La propuesta de laboratorio logra una integración efectiva entre la teoría y la práctica, diseñando actividades que refuerzan el contenido teórico y enriquecen la información de los estudiantes de ingeniería electrónica.

3. La guía didáctica proporciona herramientas esenciales para guiar al catedrático del laboratorio, mientras que la inclusión de habilidades en Linux, SQL, Python, y herramientas de visualización les da herramientas a los estudiantes para enfrentar desafíos de ciencia de datos.

4. La presentación de fundamentos teóricos y ejemplos prácticos enriquecen el laboratorio de Proyectos de Computación Aplicados a Ingeniería Electrónica, consolidando la base de programación y aplicaciones prácticas para los futuros ingenieros electrónicos.

RECOMENDACIONES

1. Renovar de manera continua las prácticas para que el contenido impartido siempre sea el más actualizado. Dado que el área de la ciencia de datos es un campo en crecimiento y constante desarrollo.

2. Tener un límite de estudiantes por sección de laboratorio, esto para que el catedrático pueda tener un mejor alcance, y para que los estudiantes puedan estar más cómodos en el área de trabajo.

3. Realizar una clase introductoria a inicio de semestre, donde se le indiquen todos los parámetros y reglas establecidas por el catedrático a los estudiantes, para que los estudiantes puedan tener claras todas las actividades que se desarrollaran durante el semestre.

4. Contratar por lo menos un auxiliar para el laboratorio, dependiendo del cupo que se establezca por sección de laboratorio se podrían tener más de un auxiliar por sección. Esto es muy importante ya que por la carga académica muchas veces el catedrático no puede darle tiempo y atención a las consultas de todos los estudiantes de todas las secciones del laboratorio, sin tomar en cuenta que el catedrático que imparte este laboratorio puede estar dando otros cursos dentro de la Facultad de Ingeniería. Por lo que si no se asigna la cantidad correcta de auxiliares se puede generar una carga muy grande en el catedrático, pudiéndose evitar esto teniendo el apoyo de auxiliares para las distintas actividades del laboratorio, como las calificaciones de tareas, hojas de trabajo, entre otros.

REFERENCIAS

FIUSAC. (s.f.). *Antecedentes.* USAC. https://portal.ingenieria.usac.edu.gt/index.php/aspirante/antecedentes

Odoo docs. (s.f.). *Su primer módulo.* OD. https://www.odoo.com/documentation/15.0/es/administration/odoo_sh/getting_started/first_module.html

Picén, E. (2012). *Análisis de la red curricular de la carrera de Ingeniería Eléctrica a nivel Iberoamericano*. [Tesis de licenciatura, Universidad de San Carlos de Guatemala]. Archivo digital. http://biblioteca.usac.edu.gt/tesis/08/08_2665_IN.pdf

USAC. (s.f.). *Misión, Objetivo y POAS.* CIP. https://cip.usac.edu.gt/index.php/mision-objetivos-y-poas/

APÉNDICE

Apéndice 1.

Instalación de librerías de ciencia de datos

A continuación, se muestra el proceso para poder instalar distintas librerías de Python para poder trabajar en la ciencia de datos.

Para poder utilizar Python en una máquina, primero se debe tener instalado Python. Se puede utilizar Python desde línea de comandos, pero no es tan sencillo, por lo que se recomienda tener un IDE que facilite el trabajo con el código.

Luego de tener instalado un IDE para trabajar con Python, normalmente no se instalan todas las librerías, por lo que se deben ir instalando conforme se vayan necesitando.

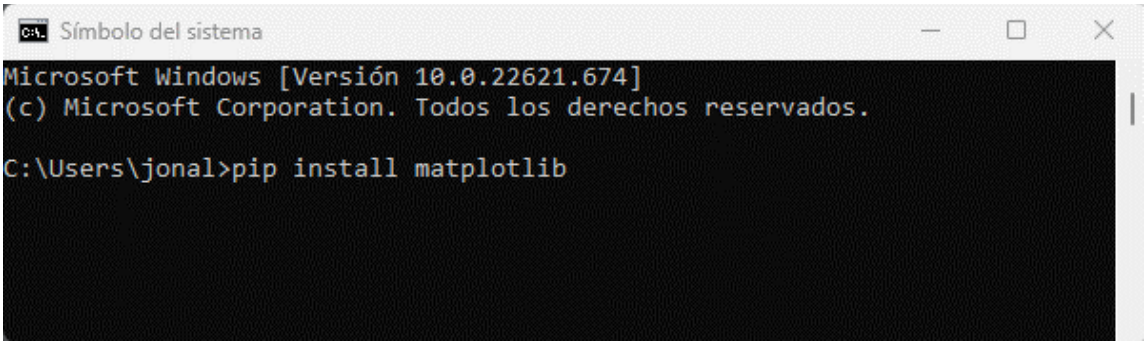
Lo primero es abrir la consola del sistema, dependiendo del sistema operativo que se esté utilizando el proceso para abrir la consola cambiara. Si no se tienen instalado PIP para Python, se debe verificar antes de intentar instalar las librerías, ya que debe estar instalado para poder instalar librerías.

Al tener verificados todos los requisitos, ya se puede ingresar a la consola. Para este ejemplo se mostrará como instalar la librería Matplotlib, y para ellos solo se debe escribir en la consola el siguiente comando: pip install matplotlib.

Continuación apéndice 1.

Si la versión que instalo en su equipo es Python 3, entonces para poder instalar librerías debería hacer un pequeño cambio que es el siguiente: pip3 install matplotlib.

Instalación de librerías de ciencia de datos



Este sería el mismo proceso para cualquier librería que se necesite. Independientemente si la librería es para ciencia de datos o es para otro uso, el procedimiento siempre es el mismo. Siempre se deben verificar todos los requisitos para poder instalar las librerías, los cuales son tener instalado Python en el equipo y verificar que este instalado PIP para que podamos instalar librerías de esta manera.

Nota. Metodología para la instalación de librerías. Elaboración propia, realizado con Word.