Trabajo Práctico Final - Programación 1 con Python

Ejemplos de temas para el proyecto

Los sistemas aquí presentados deberán simular **sistemas de gestión de datos reales**. Cada sistema deberá incluir, como mínimo, tres entidades[1] de datos interrelacionadas, y cubrir las funcionalidades solicitadas en la consigna general del trabajo práctico:

- Operaciones **CRUD** (*crear, leer, actualizar y eliminar*) sobre todas las entidades.
- Funcionalidades de **búsqueda** que relacionen datos entre entidades (por ejemplo, consultar elementos asociados a un identificador de otra entidad).
- Generación de **estadísticas básicas** a partir de los datos (por ejemplo: promedios, cantidades, porcentajes).

En relación con las **estructuras de datos**, se sugiere lo siguiente:

- Utilizar **diccionarios** cuando se trabaja con volúmenes reducidos o medios de información, con necesidad de acceso rápido a los datos a través de claves específicas (por ejemplo: clientes, productos).
- Utilizar **matrices (listas de listas)** cuando se trabaja con grandes volúmenes de información, operaciones masivas, estadísticas globales o estructuras tabulares homogéneas (por ejemplo: consulta de datos históricos o que irán incrementándose a lo largo del tiempo, como ventas, reservas).

Todas las entidades deberán contar con un **identificador único**, que puede ser generado de forma **autoincremental** o **aleatoria** (asegurando que no se repita).

Los equipos podrán seleccionar uno de los temas propuestos a continuación, o bien **proponer un sistema propio**, sujeto a la aprobación del docente.

[1] En este contexto, una entidad es una unidad conceptual claramente definida dentro del sistema que representa un tipo específico de información, habitualmente relacionada con objetos o elementos reales que el sistema debe administrar. Una entidad está compuesta por datos agrupados de manera lógica y organizada, que pueden ser almacenados en estructuras como diccionarios, listas o matrices según el tipo y volumen de información. Las entidades típicamente mantienen relaciones con otras entidades para facilitar operaciones conjuntas, búsquedas eficientes y generación de reportes o estadísticas útiles dentro del sistema desarrollado.

1. **Sistema de gestión de inventarios**

Este sistema permite controlar de manera eficiente el stock disponible en un negocio o almacén, gestionando detalladamente productos, proveedores y categorías. Las entidades principales sugeridas son: **Productos**, que contienen información sobre artículos individuales; **Categorías**, que permiten clasificar los productos según tipo o finalidad; y **Proveedores**, que agrupan la información de quienes suministran los productos. La entidad de unión natural es ***Productos***, ya que cada producto está asociado a una categoría y es ofrecido por un proveedor.

El sistema deberá incluir funciones para realizar operaciones básicas de gestión (CRUD) sobre estas entidades, así como permitir el ingreso (agregar productos al inventario) y egreso (retiro o venta de productos) del stock. Además, ofrecerá búsquedas rápidas para identificar productos específicos según su proveedor o categoría, y proporcionará estadísticas útiles como el total de productos en inventario, promedios por categoría y el porcentaje de stock según cada proveedor.

2. **Aplicación de seguimiento de proyectos**

Este sistema permite gestionar y monitorear el avance de proyectos de desarrollo de software, brindando visibilidad sobre las tareas planificadas, su estado de avance y los responsables asignados. Las entidades principales sugeridas son: **Proyectos**, que representan el conjunto general de tareas y objetivos a cumplir; **Tareas**, que detallan las actividades a realizar dentro de cada proyecto; e **Integrantes del equipo**, que son los usuarios responsables de ejecutar dichas tareas. La entidad de unión natural es ***Tareas***, ya que cada tarea está asociada a un proyecto y asignada a un integrante.

El sistema deberá incluir operaciones CRUD para cada entidad, así como funcionalidades para asignar tareas a integrantes, actualizar su estado (pendiente, en progreso, completada), y registrar nuevas tareas o proyectos. Se debe poder realizar búsquedas por integrante para ver las tareas que tiene asignadas, y generar estadísticas como la cantidad de proyectos activos, el porcentaje de tareas completadas, y el promedio de tareas por integrante. También se espera el uso de estructuras como matrices para organizar asignaciones, listas y diccionarios para almacenar información específica, y la aplicación de recursividad para generación automática de reportes jerárquicos por proyecto.

3. Plataforma de seguimiento y análisis de ventas

Este sistema tiene como objetivo registrar, consultar y analizar las ventas realizadas por una empresa, permitiendo visualizar el comportamiento comercial de productos y clientes. Las entidades principales sugeridas son: **Ventas**, que representan cada operación concreta de compra; **Productos**, que contienen información detallada sobre los artículos vendidos; y **Clientes**, que registran los datos de quienes realizan las compras. La entidad de unión es ***Ventas***, ya que vincula a un producto con un cliente en una fecha determinada.

El sistema deberá permitir operaciones CRUD sobre estas entidades, además de funcionalidades para registrar nuevas ventas, consultar el historial de un cliente específico y generar reportes a partir de archivos CSV o JSON. Las estadísticas básicas incluirán el total de ventas por producto, el promedio de ventas por cliente y el porcentaje de participación en ventas por producto dentro del total. Se utilizarán matrices para representar volúmenes de ventas, diccionarios para organizar los datos de productos y clientes, y se aplicará recursividad para cálculos acumulativos o agrupamientos complejos. El proyecto deberá incluir manejo de excepciones, control de versiones con Git y pruebas unitarias para validar sus funcionalidades clave.

4. Sistema de reservas de salas

Este sistema permite gestionar la disponibilidad y asignación de salas de reuniones en una organización, asegurando el registro ordenado de reservas y el control de los espacios disponibles. Las entidades principales sugeridas son: **Salas**, que contienen información sobre cada espacio físico (nombre, capacidad, equipamiento, etc.); **Usuarios**, que representan a quienes solicitan las reservas; y **Reservas**, que vinculan a un usuario con una sala en una fecha y horario determinados. La entidad de unión es ***Reservas***, ya que conecta a cada usuario con una sala específica en un intervalo de tiempo.

El sistema deberá implementar operaciones CRUD sobre todas las entidades, permitiendo registrar nuevas reservas, cancelar o modificar las existentes, y consultar la agenda de cada sala o usuario. También deberá ofrecer búsquedas específicas como todas las reservas realizadas por un usuario, o disponibilidad de una sala en una fecha y hora determinada, utilizando recursividad para verificar conflictos de horario. Se utilizarán matrices para representar el cronograma de ocupación, listas y diccionarios para organizar los datos, y manejo de archivos para persistencia. Entre las estadísticas sugeridas se encuentran: cantidad total de reservas por sala, promedio diario de reservas, y porcentaje de ocupación por sala en un período determinado. El sistema deberá contar con manejo de excepciones, control de versiones con Git y pruebas unitarias para validar su funcionamiento.

5. Gestión de contactos

Esta aplicación permite registrar, organizar y consultar contactos personales o profesionales, agrupándolos por criterios definidos por el usuario y registrando interacciones que ayuden a mantener un seguimiento de la relación con cada persona. Las entidades sugeridas son: **Contactos**, que almacenan los datos individuales de cada persona (nombre, teléfono, correo, etc.); **Grupos**, que permiten clasificar los contactos según afinidad o propósito (por ejemplo: trabajo, familia, clientes); e **Interacciones**, que registran los encuentros, comunicaciones u otros eventos vinculados a cada contacto. La entidad de unión es ***Interacciones***, ya que se relaciona directamente con un contacto específico y, de forma indirecta, con su grupo.

El sistema debe permitir operaciones CRUD sobre cada una de estas entidades, permitiendo agregar, editar, eliminar o consultar contactos, agruparlos y registrar nuevas interacciones. Las búsquedas podrán incluir, por ejemplo, el historial de interacciones con un contacto específico, utilizando recursividad para navegar ese historial en forma ordenada o filtrada. Se utilizarán diccionarios para representar cada contacto y sus datos, listas para agrupar contactos o interacciones, y matrices para organizar vistas tabulares o listados. Entre las estadísticas básicas a generar se incluyen: cantidad de contactos por grupo, promedio de interacciones por contacto, y porcentaje de contactos considerados activos (aquellos que han tenido interacciones recientes). También deberá incluir manejo de excepciones, lectura/escritura de archivos para guardar la información, uso de Git para el control de versiones y pruebas unitarias para validar el comportamiento del sistema.

6. Sistema de evaluación académica

Este sistema permite registrar y consultar calificaciones de estudiantes en distintas materias, con el objetivo de llevar un seguimiento académico ordenado y generar estadísticas educativas útiles. Las entidades sugeridas son: **Estudiantes**, que almacenan los datos personales y académicos de cada alumno; **Materias**, que representan las asignaturas disponibles en el sistema; y **Calificaciones**, que relacionan a cada estudiante con una materia y su correspondiente nota. La entidad de unión es *Calificaciones*, ya que vincula directamente a un estudiante con una materia específica.

El sistema debe permitir operaciones CRUD sobre cada entidad: registrar nuevos estudiantes o materias, asignar o modificar calificaciones, y consultar los datos académicos almacenados. Las búsquedas deben incluir, por ejemplo, el detalle de calificaciones de un estudiante en sus materias cursadas. Se recomienda el uso de matrices para representar los datos numéricos de calificaciones, listas y diccionarios para la organización general de datos y relaciones, y recursividad para el cálculo de promedios o exploración de notas por niveles o períodos. Entre las estadísticas a generar se incluyen: promedio general por materia, cantidad de estudiantes aprobados y desaprobados, y el porcentaje de estudiantes con promedio alto. El sistema deberá contemplar manejo de excepciones, almacenamiento de datos en archivos, control de versiones con Git, y pruebas unitarias para asegurar la confiabilidad del código.

7. Gestión de biblioteca

Este sistema permite registrar, organizar y controlar los préstamos de libros dentro de una biblioteca, tanto desde el punto de vista de los usuarios como del inventario bibliográfico. Las entidades sugeridas son: **Usuarios**, que representan a las personas habilitadas a solicitar libros; **Libros**, que almacenan los datos de cada ejemplar (título, autor, categoría, estado); y **Préstamos**, que registran cuándo un usuario retira un libro y su eventual devolución. La entidad de unión es ***Préstamos***, ya que conecta a un usuario con un libro en un período determinado.

El sistema deberá implementar operaciones CRUD para las tres entidades, permitiendo agregar nuevos libros al catálogo, registrar usuarios y gestionar los préstamos (altas, devoluciones, modificaciones). Deberá ofrecer búsquedas específicas como los libros prestados a un usuario determinado, empleando recursividad si se desea recorrer historiales de préstamos de manera jerárquica o cronológica. Se usarán matrices para organizar la disponibilidad o ubicación de libros, listas y diccionarios para representar usuarios y préstamos, y se aplicará manejo de excepciones para validar datos como fechas y disponibilidad. Entre las estadísticas a incluir se encuentran: cantidad de libros por categoría, promedio de préstamos por usuario, y porcentaje de libros actualmente prestados respecto al total del catálogo. El sistema debe guardar y recuperar la información desde archivos, mantener control de versiones mediante Git y contar con pruebas unitarias que verifiquen su correcto funcionamiento.

8. Registro de asistencia

Este sistema permite llevar un control preciso de la asistencia de estudiantes a clases, facilitando tanto el seguimiento individual como la generación de estadísticas generales. Las entidades sugeridas son: **Estudiantes**, que contienen los datos de los alumnos registrados; **Clases**, que representan cada clase dictada con materia, fecha y horario; y **Registros de Asistencia**, que vinculan a los estudiantes con las clases a las que asistieron o faltaron. La entidad de unión es ***Registros de Asistencia***, ya que enlaza a cada estudiante con cada clase específica.

El sistema deberá incluir operaciones CRUD sobre todas las entidades, permitiendo registrar nuevos estudiantes, generar clases futuras y marcar asistencias o ausencias. Se debe permitir realizar búsquedas como la asistencia completa de un estudiante, incluyendo porcentaje de asistencia total y visualización de fechas. El uso de matrices será útil para representar tablas de asistencia (filas: estudiantes; columnas: clases), mientras que listas y diccionarios servirán para almacenar y relacionar los datos. Se utilizará recursividad para calcular estadísticas globales o para analizar la evolución de asistencia a lo largo del tiempo. Entre las estadísticas a generar se encuentran: porcentaje promedio de asistencia por clase, cantidad total de asistencias y porcentaje de inasistencias por estudiante. Además, se debe contemplar manejo de excepciones para validar datos, persistencia en archivos, control de versiones con Git y pruebas unitarias para validar las funciones clave.

9. Sistema de gestión de eventos de un teatro

Este sistema permite gestionar funciones teatrales, incluyendo la programación de espectáculos, la asignación de funciones por fecha y hora, y la reserva de entradas por parte de los espectadores. Está orientado a teatros, salas culturales o centros artísticos que ofrecen múltiples funciones y desean mantener un control eficiente de la asistencia y ocupación. Las entidades sugeridas son: **Eventos (espectáculos)**, que representan las obras o shows disponibles en cartelera; **Funciones**, que corresponden a cada presentación en una fecha y horario específico; y **Reservas**, que registran los datos de quienes asisten a cada función. La entidad de unión es **Reservas**, ya que conecta a un espectador con una función determinada de un evento.

El sistema deberá implementar operaciones CRUD sobre todas las entidades, permitiendo crear nuevos espectáculos, programar funciones, registrar y cancelar reservas. También deberá permitir búsquedas como espectadores registrados para una función específica. Se utilizarán matrices para representar el calendario de funciones y la ocupación por día y horario, listas y diccionarios para estructurar los datos de espectáculos, funciones y espectadores, y recursividad para calcular ocupación acumulada o buscar funciones según filtros múltiples. Las estadísticas incluirán: cantidad total de funciones realizadas, promedio de espectadores por función, y porcentaje de ocupación por espectáculo. Se deberá incluir validación de datos mediante manejo de excepciones, almacenamiento en archivos, uso de Git para el control de versiones y pruebas unitarias que validen el correcto funcionamiento del sistema.

10. Venta de pasajes

Este sistema permite gestionar la venta de pasajes de micro/avión a distintos destinos y fechas, facilitando el registro de clientes, la planificación de viajes y el control de la ocupación. Está pensado para empresas de transporte terrestre/aérea o agencias que comercializan pasajes a nivel regional o nacional. Las entidades sugeridas son: **Clientes**, que representan a las personas que adquieren los pasajes; **Destinos**, que agrupan la información de las ciudades o localidades disponibles para viajar; y **Pasajes**, que vinculan a un cliente con un destino, en una fecha y horario determinado. La entidad de unión es ***Pasajes***, ya que conecta los datos del cliente con el destino y fecha del viaje.

El sistema debe implementar operaciones CRUD sobre cada entidad, permitiendo registrar nuevos destinos, dar de alta o baja clientes, y vender, modificar o anular pasajes. Deberá permitir búsquedas como los pasajes adquiridos por un cliente específico, utilizando recursividad si se desea navegar por el historial de viajes del cliente o filtrar por fechas. Se utilizarán matrices para representar la ocupación de los servicios por fecha y destino, y listas y diccionarios para manejar los datos de clientes y destinos. Entre las estadísticas a generar se incluyen: cantidad de pasajes vendidos por destino, promedio de ventas mensuales, y porcentaje de ocupación por destino (en relación a la capacidad disponible). El sistema debe contemplar validaciones mediante manejo de excepciones, almacenamiento persistente en archivos, control de versiones con Git y pruebas unitarias para asegurar la estabilidad del sistema.

11. Sistema de venta de entradas para recitales y eventos masivos

Este sistema permite gestionar la venta de entradas para recitales, conciertos, festivales o eventos deportivos que se desarrollan en estadios o espacios con gran capacidad. Está orientado a organizaciones que necesitan controlar sectores, fechas, categorías de entradas y la asignación de espectadores. Las entidades sugeridas son: **Espectáculos**, que representan cada evento programado (nombre del artista, fecha, lugar); **Entradas**, que vinculan un cliente con un espectáculo y un sector determinado (campo, platea, VIP, etc.); y **Clientes**, que registran a los asistentes. La entidad de unión es ***Entradas***, ya que relaciona cliente, espectáculo y sector.

El sistema debe permitir operaciones CRUD sobre todas las entidades: agregar nuevos eventos, habilitar sectores o categorías de entradas, registrar ventas y cancelar entradas. Se deberá permitir búsquedas como entradas adquiridas por un cliente específico o por sector. Las matrices pueden usarse para representar la ocupación de sectores por evento, y listas y diccionarios para organizar la información de clientes, espectáculos y entradas. Se aplicará recursividad para buscar entradas dentro de grandes listados filtrando por múltiples criterios. Las estadísticas sugeridas incluyen: cantidad de entradas vendidas por espectáculo, promedio de entradas por cliente, y porcentaje de capacidad vendida por espectáculo, considerando los diferentes sectores. Se debe contemplar el manejo de excepciones para validaciones (por ejemplo, evitar ventas duplicadas o sobreventa), almacenamiento en archivos, control de versiones con Git y pruebas unitarias que aseguren el correcto funcionamiento del sistema.

12. Reserva de turnos médicos

Este sistema permite gestionar de manera ordenada y eficiente la asignación de turnos médicos en un centro de salud, considerando distintas especialidades, profesionales y horarios de atención. Las entidades sugeridas son: **Pacientes**, que contienen los datos de las personas que solicitan atención; **Médicos**, que representan a los profesionales del centro, incluyendo su especialidad y disponibilidad; y **Turnos**, que vinculan a un paciente con un médico en un día y horario determinados. La entidad de unión es ***Turnos***, ya que conecta directamente a médicos y pacientes en una franja horaria específica.

El sistema deberá implementar operaciones CRUD sobre todas las entidades: registrar pacientes y médicos, asignar turnos, modificarlos o cancelarlos. También deberá permitir búsquedas como turnos reservados por un paciente específico o disponibilidad de turnos para una especialidad determinada, utilizando recursividad para explorar horarios futuros o filtrar por criterios múltiples. Se utilizarán matrices para representar la grilla de horarios de atención, listas y diccionarios para estructurar los datos de pacientes, médicos y especialidades, y manejo de archivos para guardar toda la información. Las estadísticas a generar deben incluir: cantidad total de turnos asignados por médico, promedio de turnos diarios por especialidad, y porcentaje de ocupación por médico en función de su agenda disponible. Además, se deberá incorporar validación de datos mediante manejo de excepciones, control de versiones con Git y pruebas unitarias para verificar el correcto funcionamiento del sistema.

13. Búsqueda y reserva de paquetes turísticos

Este sistema permite a una agencia de viajes gestionar su catálogo de paquetes turísticos, ofreciendo a los clientes la posibilidad de buscar, seleccionar y reservar viajes organizados a distintos destinos. Las entidades sugeridas son: **Paquetes**, que representan cada oferta turística disponible (incluyendo destino, fechas, duración, servicios incluidos y tipo de paquete: estándar, premium, etc.); **Clientes**, que registran los datos de las personas interesadas o que han realizado reservas; y **Reservas**, que vinculan a un cliente con un paquete turístico específico. La entidad de unión es ***Reservas***, ya que conecta el paquete con el cliente en una fecha determinada.

El sistema deberá implementar operaciones CRUD sobre cada entidad, permitiendo dar de alta paquetes nuevos, registrar clientes, gestionar reservas (crearlas, modificarlas o cancelarlas) y consultar el estado de las mismas. Se deberá permitir búsquedas como los paquetes reservados por un cliente específico, utilizando recursividad para filtrar por destino, tipo de paquete o período de viaje. Se utilizarán matrices para organizar la oferta de paquetes por destino o por fecha, listas y diccionarios para estructurar la información de clientes, paquetes y reservas, y archivos para persistencia de datos. Las estadísticas sugeridas incluyen: cantidad de reservas por destino, promedio de paquetes vendidos por período, y porcentaje de reservas según el tipo de paquete (por ejemplo, cuántos eligen paquetes premium frente a los estándar). Además, el sistema debe incluir validación robusta mediante manejo de excepciones, control de versiones con Git y pruebas unitarias para asegurar el correcto funcionamiento de todas las funciones implementadas.

14. Sistema de gestión hotelera

Este sistema permite administrar eficientemente un hotel, facilitando el control de habitaciones, la gestión de clientes y la organización de reservas. Está orientado a establecimientos de alojamiento que desean automatizar el proceso de registro, asignación y seguimiento de estadías. Las entidades sugeridas son: **Clientes**, que contienen los datos personales de los huéspedes; **Habitaciones**, que representan cada unidad disponible en el hotel con sus características (número, tipo, capacidad, estado); y **Reservas**, que vinculan a un cliente con una habitación en un período determinado. La entidad de unión es ***Reservas***, ya que relaciona habitación, cliente y fechas de ingreso y egreso.

El sistema debe implementar operaciones CRUD sobre todas las entidades, permitiendo registrar nuevos clientes, gestionar habitaciones (crear, modificar estado, dar de baja) y administrar reservas (asignar, modificar, cancelar). Las búsquedas deben permitir, por ejemplo, consultar todas las reservas realizadas por un cliente específico, o verificar la disponibilidad de una habitación en una fecha dada, utilizando recursividad para explorar la grilla de ocupación. Se utilizarán matrices para representar la ocupación diaria de habitaciones (por ejemplo, una matriz con habitaciones en filas y fechas en columnas), y listas y diccionarios para organizar la información general. Las estadísticas básicas incluirán: porcentaje de ocupación por habitación, promedio de duración de reservas, y cantidad total de reservas realizadas en un período. El sistema deberá validar correctamente los datos ingresados (como solapamiento de fechas o habitaciones ya ocupadas), guardar y recuperar información

mediante archivos, utilizar Git para el control de versiones y contener pruebas unitarias que aseguren la integridad de su funcionamiento.

15. Sistema de gestión de recursos humanos

Este sistema permite administrar de manera integral la información del personal de una organización, incluyendo sus datos personales, historial laboral, sector asignado, y registro de licencias. Está diseñado para brindar soporte a áreas de RRHH que necesiten mantener actualizada la base de empleados, hacer seguimiento de ausencias y generar reportes estadísticos para la toma de decisiones. Las entidades sugeridas son: **Empleados**, que contienen datos personales, categoría laboral y sector; **Sectores o Áreas**, que representan las unidades organizativas donde trabaja cada empleado; y **Licencias**, que registran ausencias justificadas (enfermedad, vacaciones, licencias especiales, etc.). La entidad de unión es ***Licencias***, ya que se relaciona directamente con un empleado y una fecha o período.

El sistema deberá implementar operaciones CRUD para cada entidad: alta, modificación y baja de empleados, asignación o cambio de sector, y registro de licencias o ausencias. Las búsquedas deben permitir, por ejemplo, consultar las licencias tomadas por un empleado específico o listar todos los empleados de un sector. Se utilizarán listas y diccionarios para organizar la información de los empleados y licencias, matrices para representar el historial de ausencias por día o mes, y recursividad para realizar búsquedas filtradas por criterios múltiples (por ejemplo: licencias por tipo o por rango de fechas). Las estadísticas sugeridas incluyen: promedio de licencias por empleado, cantidad de empleados por sector, y porcentaje de ausentismo por mes o por área. El sistema deberá contar con validaciones robustas mediante manejo de excepciones, persistencia de datos a través de archivos, control de versiones con Git y pruebas unitarias para asegurar la calidad del desarrollo.

16. Sistema de gestión de pedidos en un restaurante

Este sistema permite registrar y administrar los pedidos realizados por los clientes en un restaurante, facilitando la coordinación entre el salón, la cocina y la administración. Está diseñado para llevar un control eficiente de las mesas, los productos del menú, los mozos y los pedidos en curso o finalizados. Las entidades sugeridas son: **Mesas**, que representan cada unidad de atención en el salón (número, capacidad, estado: libre/ocupada); **Pedidos**, que vinculan a una mesa con un mozo y una lista de productos en un momento determinado; y **Productos del Menú**, que contienen los platos y bebidas ofrecidos, con su precio, categoría (entrada, plato principal, bebida, postre, etc.) y disponibilidad. La entidad de unión es ***Pedidos***, ya que conecta la mesa, el mozo y los productos solicitados.

El sistema deberá implementar operaciones CRUD para todas las entidades: agregar productos al menú, registrar nuevas mesas o mozos, crear, modificar o anular pedidos. Las búsquedas deben incluir consultas como los pedidos realizados por una mesa específica, o los pedidos atendidos por un mozo determinado, permitiendo recorrer registros usando recursividad para filtrar por fecha, estado u otras condiciones. Se utilizarán listas y diccionarios para organizar los datos de mesas, pedidos y productos, y matrices para representar el estado de ocupación o resumen de pedidos por turno. Entre las estadísticas sugeridas se incluyen:

promedio de productos por pedido, cantidad total de pedidos por mozo, y porcentaje de ventas por categoría de producto. El sistema debe incluir validaciones (por ejemplo, evitar que una mesa reciba más de un pedido activo a la vez), almacenamiento persistente en archivos, control de versiones mediante Git y pruebas unitarias para garantizar la solidez del sistema.

17. **Sistema de gestión de ventas y recetas en una farmacia**

Este sistema permite administrar la venta de medicamentos en una farmacia, asegurando el control del stock, la trazabilidad de los productos vendidos y la correcta vinculación con recetas médicas cuando sea necesario. Está orientado a farmacias que deseen llevar un registro detallado de sus operaciones y cumplir con los requerimientos de control sanitario. Las entidades sugeridas son: **Medicamentos**, que contienen información como nombre comercial, laboratorio, precio, stock disponible y si requieren o no receta médica; **Clientes**, que representan a las personas que realizan las compras, pudiendo incluir datos de obra social o afiliación; y **Ventas**, que vinculan a un cliente con uno o más medicamentos adquiridos, la fecha de venta y si presentó receta correspondiente. La entidad de unión es ***Ventas***, ya que relaciona cliente, medicamento y condiciones de venta.

El sistema deberá incluir operaciones CRUD para todas las entidades: alta, baja y modificación de medicamentos, registro de nuevos clientes y carga de ventas o recetas médicas. Las búsquedas deben permitir, por ejemplo, consultar los medicamentos adquiridos por un cliente específico o el historial de ventas de un producto determinado, utilizando recursividad para navegar por fechas o filtrar por condiciones. Se utilizarán listas y diccionarios para organizar la información de clientes, medicamentos y ventas, y matrices para representar el stock diario o el resumen de ventas. Las estadísticas a generar deben incluir: medicamentos más vendidos, porcentaje de ventas realizadas con receta médica, y promedio de unidades vendidas por medicamento. Además, el sistema deberá validar correctamente los datos ingresados (por ejemplo, controlar stock disponible o requerimientos de receta), guardar la información en archivos, utilizar Git para el control de versiones, y contar con pruebas unitarias para asegurar la fiabilidad del sistema.

18. **Sistema de gestión de ventas y stock en un supermercado**

Este sistema permite administrar el inventario de productos y registrar las ventas realizadas en un supermercado o hipermercado, facilitando el control de stock, la trazabilidad de los productos vendidos y el análisis de consumo por parte de los clientes. Está diseñado para cubrir las operaciones básicas de un comercio minorista o mayorista con múltiples categorías de productos. Las entidades sugeridas son: **Productos**, que contienen información como nombre, categoría (alimentos, limpieza, bebidas, etc.), precio, stock disponible y descuentos aplicables; **Clientes**, que registran los datos de quienes realizan las compras, con posibilidad de distinguir tipos de clientes (frecuente, mayorista, etc.); y **Ventas**, que vinculan productos, cliente y fecha de la operación. La entidad de unión es ***Ventas***, ya que relaciona múltiples productos con un cliente en una compra específica.

El sistema deberá incluir operaciones CRUD sobre todas las entidades: agregar nuevos productos, actualizar stock, registrar clientes, y gestionar ventas. Deberá permitir búsquedas

como compras realizadas por un cliente específico, o el historial de ventas de un producto determinado, utilizando recursividad para recorrer grandes volúmenes de datos filtrando por fecha, categoría o cliente. Se utilizarán listas y diccionarios para estructurar los datos de productos, clientes y ventas, y matrices para representar la evolución del stock o los resúmenes de ventas por categoría o día. Las estadísticas sugeridas incluyen: productos más vendidos, promedio de productos por venta, porcentaje de ventas por categoría de producto, y comparación entre stock disponible y productos vendidos. El sistema deberá incluir manejo de excepciones para validar datos críticos (como stock negativo o ventas sin productos), almacenamiento persistente en archivos, control de versiones mediante Git, y pruebas unitarias para garantizar el correcto funcionamiento de todas las funcionalidades.

19. Sistema de control de gastos personales o familiares

Este sistema permite registrar y analizar los gastos personales o familiares organizados por categorías, facilitando el seguimiento del presupuesto mensual y la toma de decisiones financieras. Es ideal para simular la gestión de una economía doméstica, incorporando conceptos como ingresos, egresos y ahorro. Las entidades sugeridas son: **Gastos**, que registran cada movimiento financiero (fecha, monto, categoría, descripción); **Categorías**, que permiten clasificar los gastos (alimentos, transporte, ocio, salud, educación, etc.); y **Presupuestos**, que establecen un monto límite por categoría o período. La entidad de unión es ***Gastos***, ya que se vincula con una categoría específica y puede asociarse a un presupuesto mensual.

El sistema deberá implementar operaciones CRUD para cada entidad: registrar nuevos gastos, modificar o eliminar registros, ajustar presupuestos o crear nuevas categorías. Las búsquedas deben permitir, por ejemplo, consultar todos los gastos realizados en una categoría determinada o por fecha específica, aplicando recursividad para filtrar rangos temporales. Se utilizarán listas y diccionarios para almacenar la información principal, y matrices para representar reportes mensuales por categoría o comparar gastos versus presupuestos. Las estadísticas sugeridas incluyen: porcentaje de gasto por categoría respecto al total mensual, promedio de gasto diario, y diferencia entre gasto real y presupuesto previsto por categoría. El sistema debe validar correctamente los montos ingresados, almacenar los datos en archivos, utilizar Git para el control de versiones y contar con pruebas unitarias que aseguren su funcionamiento correcto.

20. Sistema de gestión de clínica veterinaria

Este sistema permite gestionar la atención médica de animales en una clínica veterinaria, registrando turnos, tratamientos aplicados y la relación entre dueños y mascotas. Es útil para simular un entorno real de atención profesional, con seguimiento clínico y administración de clientes. Las entidades sugeridas son: **Mascotas**, que contienen los datos del animal (nombre, especie, raza, edad, etc.); **Dueños**, que agrupan la información de los responsables; y **Turnos**, que vinculan a la mascota con un servicio prestado, en una fecha determinada. La entidad de unión es ***Turnos***, ya que relaciona mascota, fecha y procedimiento realizado.

El sistema deberá implementar operaciones CRUD sobre las tres entidades: registrar mascotas nuevas, actualizar datos de los dueños, programar turnos o registrar tratamientos realizados. Se deberán realizar búsquedas como los tratamientos aplicados a una mascota específica, usando recursividad para recorrer historial clínico ordenado por fecha. Se utilizarán listas y diccionarios para organizar los datos, y matrices para representar calendarios de turnos o frecuencia de visitas. Las estadísticas sugeridas incluyen: cantidad de tratamientos por tipo, frecuencia de visitas por mascota, y porcentaje de turnos ocupados por especialidad o día. El sistema debe contemplar validación de datos (como duplicaciones o turnos superpuestos), almacenamiento persistente en archivos, control de versiones con Git y pruebas unitarias.

21. Sistema de gestión de socios y clases en un gimnasio

Este sistema permite gestionar socios, clases programadas y la asistencia a las distintas actividades ofrecidas en un gimnasio. Está orientado a controlar la inscripción, el acceso a clases y el seguimiento de pagos. Las entidades sugeridas son: **Socios**, que representan a los clientes con sus datos personales, tipo de abono y estado de pago; **Clases**, que corresponden a las actividades ofrecidas (zumba, funcional, spinning, etc.) con día, hora e instructor asignado; y **Asistencias o Inscripciones**, que vinculan a un socio con una clase específica. La entidad de unión es ***Asistencias***, ya que conecta socio, clase y fecha.

El sistema deberá permitir operaciones CRUD sobre todas las entidades: registrar nuevos socios, crear y modificar clases, y registrar o anular asistencias. Se podrán realizar búsquedas como las clases a las que asistió un socio específico, usando recursividad para filtrar por tipo de clase o período. Se utilizarán listas y diccionarios para almacenar los datos, y matrices para representar la ocupación de clases o el calendario semanal. Entre las estadísticas sugeridas están: promedio de asistencia por clase, porcentaje de ocupación de cada actividad, y cantidad de socios activos vs. inactivos. El sistema debe incorporar validaciones para evitar inscripciones duplicadas o fuera de cupo, almacenar la información en archivos, utilizar Git como sistema de control de versiones y contener pruebas unitarias para verificar la integridad del sistema.

Importante: Cualquier otro tema puede ser propuesto y considerado con la aprobación del docente.