Universidad Autónoma del Estado de Hidalgo

ICBI

Licenciatura
Licenciatura en Ciencias Computacionales

Semestre y grupo
6°2

Nombre de la materia
Bases de Datos Distribuidas

Nombre del alumno
Eduardo Vazquez Bonilla
Cesar Martinez Andrade
Sebastian Trejo Muñoz
Jesús Romo Guzmán
Fernando Hernandez Morales
Braulio Adrian Pérez Orta
Kevin Axel Chavez Quiroz

Nombre del proyecto
GYM

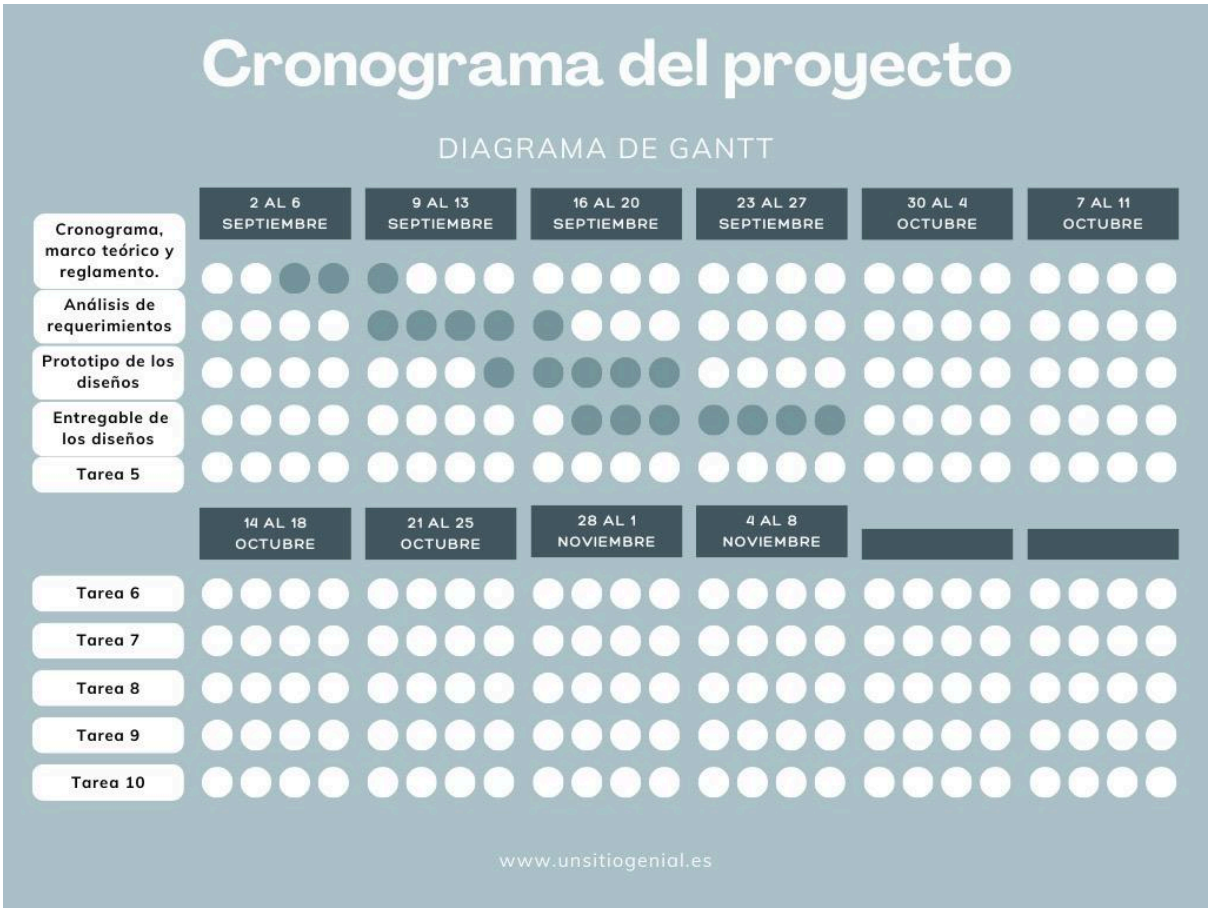
Nombre del equipo
DROP DATABASE

Nombre del profesor
Eduardo Cornejo Velazquez

Fecha
Septiembre 2024

Índice.

Cronograma



Introducción

En este proyecto se realizará una Base de Datos Distribuida Homogénea con la finalidad de llevar el control de un gimnasio, de igual manera se espera diseñar un sistema en java con el cual se pueda gestionar la base de datos, cabe mencionar que, no solo se hará desde una computadora, si no, desde 2 de manera simultánea para poder aplicar la base de datos distribuida La finalidad de este proyecto es emplear base de datos distribuida en pro de aplicar todos los fundamentos de base de datos distribuida para llevar a cabo y observar todas los beneficios de la misma.

Se utilizará una metodología SCRUM con registro de cada avance por medio de versiones del proyecto, esperando tener una buena coordinación del equipo y de igual manera un avance colaborativo y eficiente.

Se llevará a cabo un estudio de campo a fin de tener una base de datos eficiente y adecuada para el negocio a gestionar (gimnasio) y añadiendo como mínimo una propuesta de una idea innovadora a favor del negocio.

Planteamiento del problema

El problema consiste en el manejo de datos e información de múltiples gimnasios y que haya comunicación entre ellas debido a que, es lo mas optimo para el cliente el tener su información siempre actualizada y disponible y el hecho de no contar con una BDD afecta en la transparencia de datos y la actualizaciónes de los mismos.

De manera que, buscamos el diseño adecuado para este caso, en el cual considerando todo lo anterior logrará ser una base de datos que cumpla con todos los requerimientos de nuestro cliente, cabe destacar que, La base de datos distribuida es Homogénea, entonces todos los DBMS serán iguales y por consecuencia el diseño también lo será.

Objetivo General

El objetivo principal es implementar las base de datos en 2 computadoras que funcionan simultáneamente y mantengan comunicacion entre sí, asimismo, crear un diseño óptimo de la BDD para que gestione bien los datos de los gimnasios, sin dejar de lado las interfaces de las terminales, que si bien el objetivo principal es el diseño de la base de datos distribuida, se espera mejorar, optimizar e implementar un buen software para manejar las Bases de Datos.

Se cuenta con los recursos necesarios para llevar a cabo el proyecto, un equipo coordinado, responsable y que ven por el bien del proyecto, también por potencia de hardware no habrá inconvenientes ni mucho menos de tiempo disponible.

El progreso del proyecto es medible gracias a la metodología que estamos ocupando (SCRUM) y al software que hemos implementado como lo es github y LATex, que si bien no son explicitamente necesarios para el desarrollo de la base de datos distribuida, son software de mucha ayuda, para el control de versiones del proyecto.

Objetivos específicos

1. Registrar y gestionar la información de los clientes, el cual se creara un sistema para almacenar y actualizar los datos personales de los clientes o socios.
2. Controlar las suscripciones y pagos, implementando un sistema de seguimiento de pagos, donde registre los pagos realizados y generar advertencias para pagos pendientes.
3. Administrar el inventario de productos en venta, como suplementos, proteínas, pre-entreno, etc., crear un inventario para registrar el control de los productos, ofrece el gimnasio.

4. Registrar y gestionar la información de los empleados del gimnasio, el cual se realizará un sistema para almacenar información esencial de los empleados.
5. Administración de equipos para registrar y monitorear los equipos del gimnasio, como el estado, la ubicación y su mantenimiento.
6. Asegurar la escalabilidad del sistema, diseñando una base de datos y la infraestructura para soportar el aumento en números de clientes y en el aumento de datos sin comprometer el rendimiento, implementando prácticas de optimización y monitoreo continuo para que se pueda adaptar a expansiones y demandas futuras.
7. Mejorar la disponibilidad y fiabilidad de los datos, para que en caso de un fallo los datos sigan siendo accesibles.

Justificación

Actualmente, los gimnasios presentan desafíos para gestionar grandes volúmenes de datos, los cuales están relacionados con sus operaciones que se realizan diariamente, como información de los empleados, información de los clientes, inventario, pagos, entre otros más. La implementación para esta base de datos distribuida ofrece una solución eficiente y con escalabilidad para manejar los datos de una manera efectiva.

A medida que el gimnasio va creciendo y expandiéndose, la cantidad de datos también aumenta, permitiendo añadir nuevos nodos para tener una mejora en la disponibilidad y fiabilidad de los datos, ya que si ocurriera un fallo, los datos seguirán siendo accesibles porque al distribuir los datos en varios nodos, esto reduciría la carga en un solo servidor y genera una optimización al sistema.

Alcances y limitaciones

La base de datos distribuida ofrece sus alcances para el negocio, así como sus limitaciones.
Al estar los datos en múltiples nodos, la base de datos podrá seguir funcionando incluso si un nodo falla, dando una disponibilidad y redundancia en los datos que se almacenan en el sistema. Al manejar un volumen mayor de datos, su escalabilidad será más fácil, ya que se podrán añadir nuevos nodos, esto sin afectar el rendimiento del sistema, obteniendo un acceso más rápido y gestionando las diversas áreas del gimnasio, obteniendo una flexibilidad en la integración de diferentes tipos de datos y sistemas.

Como limitaciones nos encontramos con la configuración y el diseño de la base de datos distribuida, ya que esta es más compleja que una base de datos centralizada. Al estar dependiendo de la Red para que todos los nodos están en lugares

diferentes y contar con una red mala calidad y estabilidad, podría ser un problema afectando a los rendimientos de la base de datos.

Marco teórico

Metodología de análisis.

UML

Se implementará UML para la modelación visual del sistema, lo cual permitirá representar de manera clara la estructura y las interacciones entre los diferentes componentes del sistema. Esto facilitará la comprensión y el análisis del proyecto, promoviendo una mejor toma de decisiones en el diseño e implementación. Al aplicar la metodología SCRUM, podremos gestionar el desarrollo de manera iterativa y ágil, permitiendo adaptarnos rápidamente a los cambios en los requisitos y mejorar la eficiencia del equipo. La combinación de estas dos herramientas proporcionará una base sólida para un desarrollo estructurado y centrado en los usuarios, asegurando un proceso dinámico y adaptable.

Historias de usuario

Para este apartado se iniciará haciendo las historias de usuario. Podemos decir que estas son una breve descripción escrita que captura lo que un usuario (actor) quiere lograr con el sistema. Las historias de usuario están estrechamente vinculadas con los requerimientos del sistema. Generalmente las historias de usuario siguen una estructura sencilla que describe:

- *Quién* es el usuario.
- *Qué* desea hacer o lograr.
- *Para qué* desea hacerlo (su objetivo o beneficio).

Las historias de usuario se elaboran teniendo la perspectiva del usuario final. Si debemos definir un formato sería como:

> Como [tipo de usuario], quiero [acción o funcionalidad] para [beneficio o razón].

Su objetivo es definir solamente los requisitos de alto nivel, además de servir como una visión global del potencial alcance y los beneficios esperados.

Una parte importante de las historias de usuario son los denominadas estancias o escenarios, estos estan diseñados para escribir pruebas concretas de aceptación, positivas o negativas sea en el marco de testing.

Casos de usuario

Un diagrama de casos de uso nos permitirá identificar y visualizar todas las funcionalidades que el sistema debe ofrecer a los diferentes tipos de usuarios (actores). Además de que es visualmente sencilla de comprender, esto facilita la comunicación entre desarrolladores, clientes, y otros involucrados en el proyecto.

Teniendo en cuenta lo anterior hay que definir los puntos claves para la buena implementación del diagrama de casos de uso:

1. Actores.
2. Descripción
3. Precondiciones.
4. Postcondiciones.
5. Flujo normal.
6. Flujos alternativos.
7. Excepciones.

8. Prioridades
9. Frecuencia de uso.
10. Reglas.
11. Requerimientos especiales.
12. Suposiciones.

Una vez que tengamos definidas las historias de usuario, puedes crear diagramas de casos de uso para representar gráficamente cómo los actores (usuarios) interactúan con el sistema y las funcionalidades que necesitan. En SCRUM, estos diagramas se pueden usar en dos momentos clave:

- Definir el alcance: Puedes usar los diagramas de casos de uso durante la planificación inicial del proyecto para asegurarte de que todas las funcionalidades requeridas están cubiertas. Cada caso de uso debe estar vinculado a una o varias historias de usuario.
- Planificación de sprints: Al inicio de cada sprint, se seleccionan las historias de usuario más prioritarias para desarrollar. Cada historia de usuario puede estar respaldada por uno o varios casos de uso que guíen al equipo de desarrollo sobre cómo implementar la funcionalidad.

Metodología de diseño

El diseño de interfaces gráficas de usuario (GUI) es crucial para la creación de software eficiente y fácil de usar. En el contexto de un gimnasio, una interfaz bien diseñada puede mejorar la experiencia del usuario, facilitando la gestión de actividades, reservas y más.

Fundamentos del Diseño de Interfaces

El diseño de interfaces gráficas se basa en varios principios clave:

- Usabilidad: La interfaz debe ser intuitiva y fácil de navegar.
- Accesibilidad: Debe ser accesible para todos los usuarios, incluyendo aquellos con discapacidades.
- Estética: Un diseño visualmente atractivo puede mejorar la satisfacción del usuario.
- Consistencia: Mantener una apariencia y comportamiento consistentes en toda la aplicación.

Metodologías de Diseño

- Diseño Centrado en el Usuario (UCD): Este enfoque pone al usuario en el centro del proceso de diseño, asegurando que sus necesidades y preferencias sean prioritarias.
- Pensamiento de Diseño: Una metodología que fomenta la creatividad y la resolución de problemas a través de la empatía con el usuario.
- Desarrollo Ágil: Un enfoque iterativo que permite ajustes rápidos basados en la retroalimentación del usuario.

Herramientas de Diseño

- Figma: Una herramienta popular para el diseño de interfaces que permite la colaboración en tiempo real.
- Pencil: Utilizada ampliamente para el diseño de interfaces.
- Adobe XD: Ofrece herramientas robustas para el diseño y prototipado de interfaces.

Componentes Clave de la Interfaz

- Gestión de Usuarios: Debe proporcionar una visión general clara de las actividades y métricas importantes.
- Gestión de Empleados: Facilitar la creación y administración de perfiles de usuarios.
- Punto de venta: Un sistema intuitivo para reservar clases y equipos.

Consideraciones de Usabilidad

- Feedback del Usuario: Proporcionar retroalimentación inmediata para las acciones del usuario.
- Minimización de la Carga Cognitiva: Simplificar las tareas y reducir la cantidad de información que el usuario debe procesar.
- Diseño Responsivo: Asegurar que la interfaz funcione bien en diferentes dispositivos y tamaños de pantalla.

Metodología de desarrollo.

JAVA

Java es un lenguaje orientado a objetos, lo que significa que se basa en el concepto de clases y objetos para estructurar el código. La programación orientada a objetos facilita la reutilización del código, la modularidad y el mantenimiento, lo que es especialmente útil en proyectos grandes y complejos.
La interfaz de usuario será desarrollada en Java, lo que permitirá ofrecer una experiencia interactiva y amigable para los administradores del sistema. Además, el uso de Java facilita la escalabilidad y mantenimiento del software, garantizando que la aplicación pueda adaptarse al crecimiento de la franquicia.

INTERFACES GRÁFICAS

Una gran ventaja de trabajar con java, es que se puede trabajar con ventanas o interfaces gráficas. Y se comportan de la misma forma independientemente del sistema operativo con el que trabajemos (windows, linux etc) y, en muchas ocasiones, utilizan los mismos componentes básicos para introducir órdenes (menús, botones, cuadros de texto, etc.).

Las interfaces gráficas se usan para interaccionar con el usuario, mostrando todas las opciones que el usuario puede realizar. Por lo tanto el diseño consistirá en crear objetos que den lugar a ventanas y sobre esas ventanas se dibujaran otros objetos llamados controles. Cada objeto estará ligado a un código que estará inactivo hasta que se produzca el evento que lo activa.

CLASES

El lenguaje java es un lenguaje orientado a objetos. La base de la POO (Programación orientada a objetos) es la clase.Una clase es un tipo de objeto definido por el usuario.
Una característica que aporta la POO es la herencia ya que permite la reutilización del código escrito por nosotros o otros.

VARIABLES.

Una variable representa un espacio de memoria para almacenar un valor de un determinado tipo.La declaración de una variable puede realizarse en el ámbito de la clase (dentro de la clase pero fuera de todo método). en el ámbito del método o en el ámbito de un bloque cualquiera delimitada por { }.

MySQL

Para poder cumplir con la creación de un sistema eficiente y de calidad usaremos MySQL como sistema de gestión de bases de datos relacionales. Lo anterior debido a la experiencia que se tiene acerca de ese RDBMS, además de robustez, capacidad de manejo de grandes volúmenes de datos, sin olvidar su fácil integración con distintos lenguajes de programación. Una base de datos distribuida se compone de múltiples nodos interconectados, cada uno de los cuales almacena una parte de la información total. Algunos conceptos clave son:

- Fragmentación: Dividir la base de datos en fragmentos más pequeños para distribuirlos entre los nodos. Puede ser horizontal (por filas) o vertical (por columnas).
- Replicación: Mantener copias idénticas de los datos en varios nodos para mejorar la disponibilidad y la tolerancia a fallos.
- Transacciones distribuidas: Coordinar operaciones que afectan a múltiples nodos para mantener la consistencia.

MySQL Cluster

MySQL Cluster es una solución de base de datos distribuida proporcionada por MySQL. Algunos aspectos relevantes son:

- Nodos: Un clúster MySQL consta de varios nodos, incluyendo nodos de datos, nodos de gestión y nodos SQL.
- Almacenamiento distribuido: Los datos se distribuyen entre los nodos de datos para lograr alta disponibilidad y escalabilidad.
- Replicación síncrona: MySQL Cluster admite replicación síncrona para garantizar la consistencia de los datos.

Modelos

- Modelo Relacional: El modelo relacional es el más común en bases de datos. Se basa en tablas (relaciones) con filas y columnas. Cada tabla representa una entidad (por ejemplo, Clientes, Instructores, Cursos) y las relaciones entre ellas se establecen mediante claves primarias y foráneas.
- Modelo Entidad-Relación (ER): Este modelo se utiliza para diseñar la estructura lógica de la base de datos. Representa entidades, atributos y relaciones entre ellas mediante diagramas ER.
- Modelo Físico: El modelo físico describe cómo se implementa la base de datos en el sistema de gestión de bases de datos (DBMS). Incluye detalles como índices, claves y restricciones.

Palabras reservadas de MySQL

- SELECT: Recupera datos de una o más tablas. Es la base de las consultas y permite especificar qué columnas se desean ver y bajo qué condiciones.
- INSERT INTO: Agrega filas a una tabla. Se utiliza para insertar nuevos registros en la base de datos.
- UPDATE: Modifica los valores de una fila existente en una tabla. Útil para actualizar información.
- DELETE FROM: Elimina filas de una tabla. Permite borrar registros.
- ALTER TABLE: Modifica la estructura de una tabla existente, como agregar o eliminar columnas.
- CREATE TABLE: Crea una nueva tabla con sus columnas y restricciones.
- JOIN: Combina datos de dos o más tablas basándose en una condición. Por ejemplo, para obtener información de clientes junto con sus instructores.
- GROUP BY: Agrupa filas según una columna específica. Útil para cálculos agregados como contar o promediar.
- ORDER BY: Ordena los resultados según una o más columnas. Puede ser ascendente (por defecto) ó descendente.
- Funciones de agregación: Incluyen SUM, COUNT, AVG, MIN y MAX. Realizan cálculos sobre grupos de filas.

Replicación

La replicación en MySQL es un proceso mediante el cual se crean copias de la base de datos en varios servidores. Establece una relación maestro-esclavo, donde el servidor maestro (o primario) es la fuente de los datos, y los servidores esclavos (o secundarios) mantienen copias actualizadas de estos datos.
Cualquier cambio realizado en el servidor maestro se refleja automáticamente en los servidores esclavos. Esto mejora la disponibilidad y el rendimiento de la base de datos, ya que los esclavos pueden manejar consultas de lectura sin afectar al maestro.

- **Replicación Asincrónica**: El maestro envía los cambios a los esclavos sin esperar confirmación. Es más rápido pero puede haber pérdida de datos en caso de fallo.
- **Replicación Sincrónica**: Los esclavos confirman la recepción de los cambios antes de que el maestro los considere completos. Es más seguro pero con mayor latencia.

Análisis

Definición de requerimientos

Requerimientos funcionales:

- Gestión de membresías
    - Registro y actualización: La base de datos debe permitir el registro y actualización de los datos de los miembros.
    - Historial de pagos: El sistema debe almacenar y gestionar los pagos realizados por los miembros detallando la fecha cuando se realizó, método de pago y monto que se pagó.
    - Control de membresías: La base de datos debe tener un registro del estado de la membresías del usuarios para identificar rápidamente qué miembros tienen una membresía activa o inactiva, y el sistema debe activar o desactivar el acceso.
- Gestión de empleados
    - Registro de empleados y roles (actualización): El sistema debe gestionar los datos de los empleados, incluyendo nombre, dirección y roles.
    - Nomina: La base de datos debe tener un registro de los sueldos de los empleados, teniendo en cuenta diferentes factores como su rol y turnos de los empleados.
    - Turnos: Debe facilitar la planificación de horarios y turnos de los empleados.
    - Estatus: Aqui debe de marcarse el estado del empleado.
- Check-IN
    - Entrada y salida: Los miembros y empleados deben poder entrar al gimnasio mediante tarjetas de identificación o un sistema biométrico.

- Clases y actividades
    - Horarios: Almacenamiento y gestión de los horarios de clases grupales y personalizadas.
    - Reservación: Permitir a los miembros reservar espacio en las clases.
- Gestión de equipo
    - Estado: Seguimiento del estado y ubicación de los equipos de gimnasio.
    - Mantenimiento (booleano): Almacenamiento de datos relacionados con el mantenimiento y reparaciones de los equipos.
    - Registro: Especificaciones del equipo.
- Gestión de inventario
    - Registro: características de los productos además de su respectivo código de barras.
- Sistema de puntos
    - Historial de usuario: Permitir al miembro la obtención de puntos que pueden ser utilizados para distintos beneficios al hacer acciones específicas dentro del gimnasio.

Requerimientos no funcionales

- Disponibilidad: La base de datos distribuida debe estar disponible las 24 horas del día, los 7 días de la semana, permitiendo el acceso a los datos en tiempo real desde múltiples ubicaciones.
- Escalabilidad: El sistema debe ser capaz de escalar vertical y horizontalmente a medida que el gimnasio crece, además de permitir la adición de nuevos nodos o servidores sin afectar el rendimiento del sistema.
- Rendimiento: La base de datos debe manejar eficientemente las operaciones diarias, como las consultas de membresía y los pagos.

- Consistencia de datos: La base de datos debe garantizar la consistencia de los datos entre los diferentes nodos distribuidos, para asegurar que las actualizaciones y consultas reflejen la información correcta.

Requerimientos de almacenamiento y replicación

- Distribución geográfica: Los nodos de la base de datos deben estar distribuidos en ubicaciones estratégicas para garantizar una menor latencia.

- Replicación de datos: Implementar replicación de datos entre los nodos para garantizar que en caso de que un nodo falle, otro nodo tenga una copia actualizada de los datos.

- Tolerancia a fallos: Los nodos de la base de datos deben ser capaces de manejar fallos parciales sin comprometer la operación general del sistema. Esto implica tener mecanismos de recuperación ante desastres, como copias de seguridad automáticas y planes de recuperación de datos.

Diagrama de casos de uso

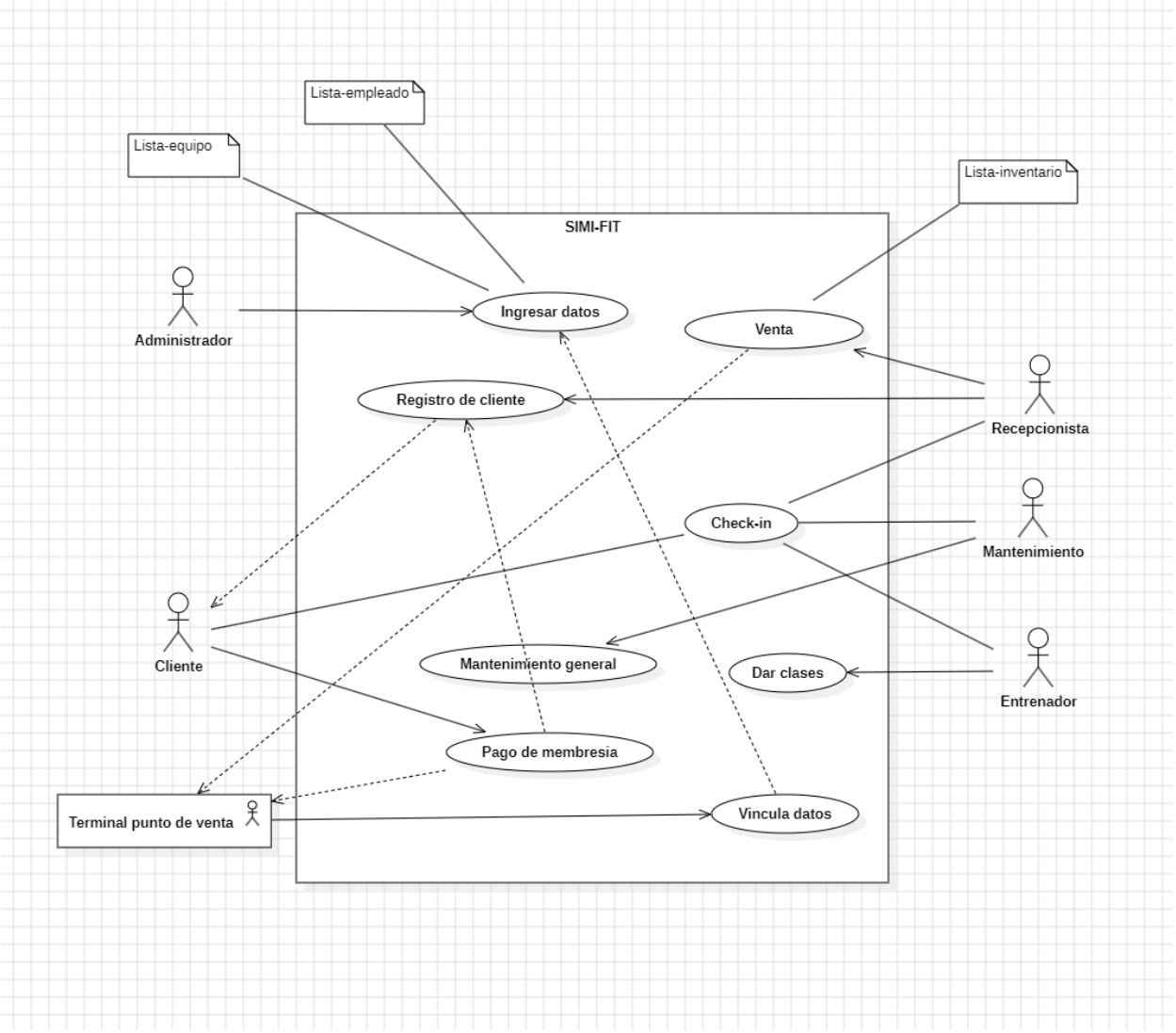


Tabla de Casos de uso (Borrador)

Caso de Uso #1	Registro e inscripción (mensualidad) de cliente a Gimnasio.	
Descripción	El cliente intenta inscribirse en el Gimnasio, el espacio y servicio están disponibles, así que solicita al Recepcionista que lo haga.	
Precondiciones	El Servicio debe existir y debe estar disponible.	
Trigger	El cliente solicita al recepcionista ser registrado e inscrito en el gimnasio.	
Succes	El cliente será registrado e inscrito al Gimnasio y tendrá acceso al equipo del Gimnasio.	
Abort	Cualquier adición al registro e inscripción hecha debe ser eliminada.	
Actores	Primary	Recepcionista.
	Secondary	Cliente.

Referencias bibliográficas.

- López, C. P. (2008). MySQL para Windows y Linux.
- Wiederhold, G. (1985). Diseño de bases de datos.
- Fontela, C. (2012). UML modelo de software para profesionales.
- Shaver, D. (1998). El siguiente paso en la mercadotecnia directa: cómo usar bases de datos orientándolas al consumidor.
- Hernandez. (2010). METODOLOGIA DE LA INVESTIGACIO.
- De Miguel Castaño, A., Fernandez, P. M., & Galan, E. C. (2001). Diseño de bases de datos: problemas resueltos.
- Cuadra, D., Castro, E., & Iglesias, A. M. (2013). Desarrollo de bases de datos: casos practicos desde el analisis a la implementacion.