

GYM

Manual Tecnico

Versión: 0001

Fecha: 25/09/2025

[Versión: 0001]

HOJA DE CONTROL

|  |  |  |  |
| --- | --- | --- | --- |
| **Organismo** | USAC CUNOC | | |
| **Proyecto** | GYM | | |
| **Entregable** | Manual Tecnico | | |
| **Autor** | Selvyn Estuardo Ixtabalan Tistoj | | |
| **Versión/Edición** | 0001 | **Fecha Versión** | 25/09/2025 |
| **Aprobado por** | Ing. Bryan Monzón | **Fecha Aprobación** | 25/09/2025 |
|  |  | **Nº Total de Páginas** | 14 |

REGISTRO DE CAMBIOS

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Causa del Cambio** | **Responsable del Cambio** | **Fecha del Cambio** |
| 0001 | Versión inicial | Selvyn Estuardo Ixtabalan Tistoj | 25/09/2025 |
|  |  |  |  |
|  |  |  |  |

CONTROL DE DISTRIBUCIÓN

|  |
| --- |
| **Nombre y Apellidos** |
| Selvyn Estuardo Ixtabalan Tistoj |
|  |
|  |
|  |
|  |

ÍNDICE

[1 DESCRIPCION GENERAL SOLUCION 4](#__RefHeading__808_995473275)

[2 REQUERIMIENTOS MINIMOS 5](#__RefHeading__816_995473275)

[3 DICCIONARIO CLASES](#__RefHeading__822_995473275) 6

[4 DICCIONARIO METODOS/PAQUETES](#__RefHeading__836_995473275) 9

# DESCRIPCIÓN GENERAL SOLUCION

La aplicación de escritorio está desarrollada en Java utilizando NetBeans, aprovechando Swing para crear una interfaz gráfica intuitiva y amigable. A través de esta interfaz, los usuarios pueden gestionar clientes, entrenadores, rutinas, pagos, sucursales y equipos, además de generar reportes dinámicos sobre asistencia, ingresos y utilización de recursos. La interacción con la base de datos se realiza mediante PostgreSQL, asegurando la persistencia y consistencia de la información.

La arquitectura sigue el patrón MVC, separando claramente la lógica de negocio, la presentación y el control de eventos. El modelo se encarga de la manipulación de datos y de ejecutar las consultas SQL necesarias; la vista presenta la información al usuario mediante formularios, tablas y menús interactivos; y el controlador gestiona los eventos generados por la interfaz, validando los datos y coordinando las operaciones con el modelo.

Además, la aplicación garantiza la trazabilidad de todas las operaciones, registrando movimientos de clientes, pagos y cambios en el inventario, lo que permite mantener un historial confiable de las acciones realizadas. La combinación de Java, Swing y PostgreSQL proporciona una solución multiplataforma, mantenible y escalable, capaz de adaptarse a futuras necesidades del sistema sin comprometer la estabilidad ni la integridad de los datos.

## 

## 

# REQUERIMIENTOS MINIMOS

- Una distribución de Windows 10 o 11 (Windows 8.1 recomendado).

- NetBeans 17 IDE en versión compatible con JDK 20 (Linux/Windows recomendado)

- Procesador: Intel Core I5

- Memoria RAM: 4 GB

- Espacio de disco duro disponible: 8 GB

En equipos con características similares o mejores debe funcionar sin problema.

## 

# 

# DICCIONARIO CLASES

En el paquete **Modelo** se encuentran las siguientes clases:

**Admin.java**: Representa a los administradores del sistema. Se encarga de gestionar usuarios, permisos y generar reportes.

**Asis.java**: Maneja todo lo relacionado con la asistencia de los clientes, registrando entradas y salidas.

**Ej.java**: Se centra en los ejercicios, almacenando datos como nombre, tipo, nivel y descripción.

**Entrena.java**: Relacionada con los entrenamientos, sirve para definir, asignar y consultar planes de entrenamiento.

**IM.java**: Permite calcular el Índice de Masa Corporal y consultar los resultados relacionados con la salud del usuario.

**Inicio.java**: Gestiona el inicio de sesión, cierre de sesión y la carga de la pantalla principal del sistema.

**Inv.java**: Representa el inventario de productos o equipos, permitiendo consultar existencias y registrar movimientos.

**Membre.java**: Se ocupa de las membresías, ya sea crearlas, renovarlas, cancelarlas o consultarlas.

**Pago.java**: Administra los pagos realizados por los usuarios, registrando transacciones y generando recibos.

**RE.java**: Posiblemente corresponde a un registro de ejercicios o entrenamientos realizados por los usuarios.

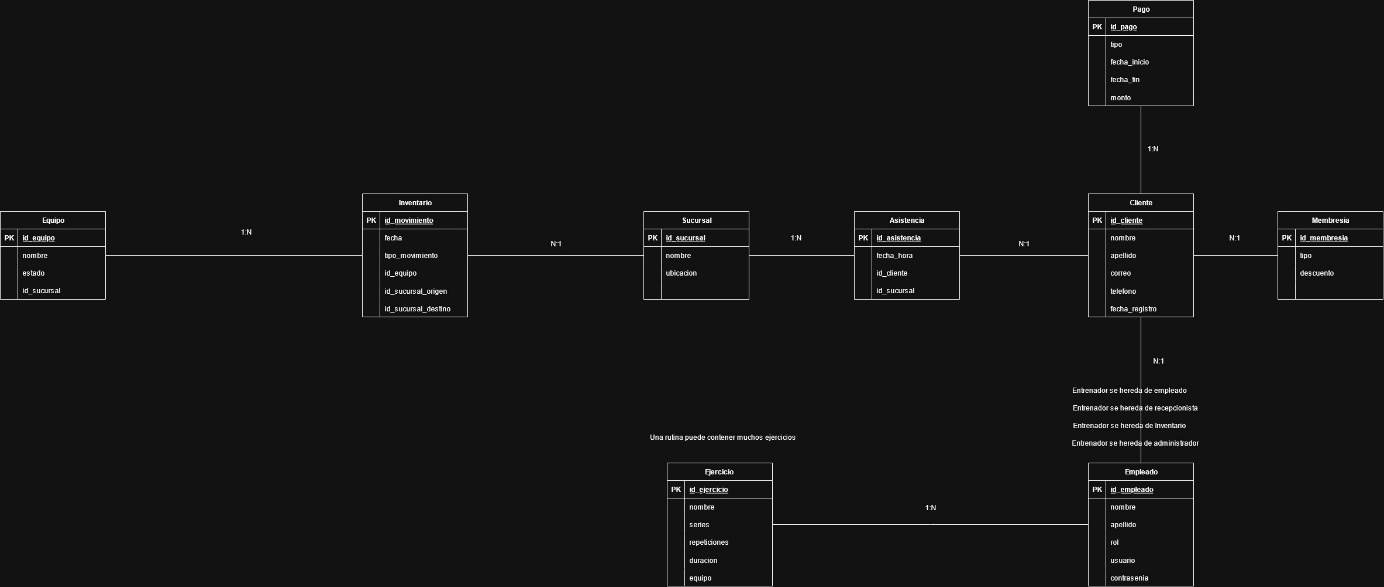
**Recep.java**: Modela las funciones de un recepcionista, como registrar clientes, gestionar accesos y atender consultas.

**Ruti.java**: Permite crear, editar y asignar rutinas de ejercicio personalizadas a los clientes.

**Ser.java**: Maneja los servicios adicionales ofrecidos por el sistema, como clases especiales o asesorías.

**Servis.java**: Probablemente es una clase de apoyo con métodos utilitarios para validaciones o generación de códigos.

**Traslado.java**: Administra traslados, ya sea de inventario, productos o incluso de usuarios entre sedes.



# 

# DICCIONARIO DE METODOS/FUNCIONES PRINCIPALES

Cada clase suele incluir ciertos métodos clave:

**Admin**: dispone de funciones para crear, modificar y eliminar usuarios, así como generar reportes de gestión.

**Asis**: tiene métodos para registrar asistencias, consultar la asistencia de un usuario y listar el historial.

**Ej**: permite agregar, editar, eliminar y consultar ejercicios almacenados.

**Entrena**: maneja la creación de entrenamientos, su asignación a los clientes y la consulta de sus detalles.

**IM**: incluye un método principal para calcular el IMC y otro para mostrar los resultados obtenidos.

**Inicio**: contiene el login, logout y la carga del menú principal del sistema.

**Inv**: gestiona el inventario mediante funciones para agregar productos, eliminar elementos, consultar el stock y actualizar cantidades.

**Membre**: ofrece la creación de membresías, la renovación, la cancelación y la consulta de información de cada socio.

**Pago**: registra nuevos pagos, permite consultar pagos anteriores y generar recibos o comprobantes.

**RE**: incluye métodos para registrar un ejercicio realizado y consultar los registros históricos.

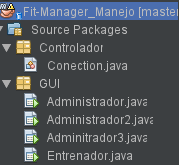
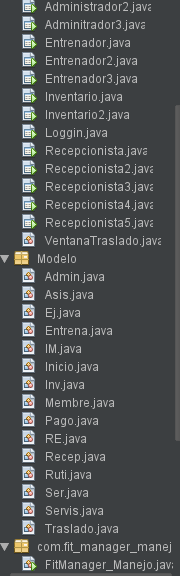
**Recep**: facilita el registro de nuevos clientes, la consulta de datos y la gestión de accesos.

**Ruti**: crea, edita, asigna y consulta rutinas personalizadas de ejercicios.

**Ser**: administra la lista de servicios adicionales, con funciones para agregarlos, eliminarlos y consultarlos.

**Servis**: contiene métodos de apoyo como validación de datos y generación de códigos.

**Traslado**: registra movimientos de traslados y permite consultar el historial de estos procesos.



POM:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com</groupId>

<artifactId>Fit-Manager\_Manejo</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>21</maven.compiler.source>

<maven.compiler.target>21</maven.compiler.target>

<exec.mainClass>com.fit\_manager\_manejo.FitManager\_Manejo</exec.mainClass>

</properties>

<dependencies>

<!-- Driver PostgreSQL -->

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.6.0</version>

</dependency>

<!-- Logging básico con SLF4J -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>2.0.9</version>

</dependency>

<dependency>

<groupId>unknown.binary</groupId>

<artifactId>AbsoluteLayout</artifactId>

<version>SNAPSHOT</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compilador de Java -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>${maven.compiler.source}</source>

<target>${maven.compiler.target}</target>

</configuration>

</plugin>

<!-- Ejecución con mvn exec:java -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>${exec.mainClass}</mainClass>

</configuration>

</plugin>

<!-- Generar JAR ejecutable con dependencias incluidas -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-shade-plugin</artifactId>

<version>3.5.0</version>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>shade</goal>

</goals>

<configuration>

<!-- Sobrescribir el JAR normal -->

<shadedArtifactAttached>false</shadedArtifactAttached>

<createDependencyReducedPom>false</createDependencyReducedPom>

<transformers>

<transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">

<mainClass>${exec.mainClass}</mainClass>

</transformer>

</transformers>

<filters>

<filter>

<artifact>\*:\*</artifact>

<excludes>

<exclude>META-INF/\*.SF</exclude>

<exclude>META-INF/\*.DSA</exclude>

<exclude>META-INF/\*.RSA</exclude>

<exclude>META-INF/LICENSE\*</exclude>

<exclude>META-INF/versions/\*/module-info.class</exclude>

</excludes>

</filter>

</filters>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>unknown-jars-temp-repo</id>

<name>A temporary repository created by NetBeans for libraries and jars it could not identify. Please replace the dependencies in this repository with correct ones and delete this repository.</name>

<url>file:${project.basedir}/lib</url>

</repository>

</repositories>

</project>