

Universidad Nacional de Cuyo  
Facultad de Ingeniería  
Licenciatura en Ciencias de la Computación

---

## Ingeniería del Software II

Trabajo Integrador N°1: *Sistema de Software para Gimnasio*

**Grupo 2 - ElvaStudio**

---

Trinidad Perea  
Perla Valerio  
Giovanni Azurduy  
Adriano Fabris

Septiembre 2025

## Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Caso de Uso Crítico: Pago de Cuotas</b>	<b>4</b>
2.1. Diagrama de Caso de Uso . . . . .	4
2.2. Escenario de Caso de Uso . . . . .	5
2.3. Diagrama de Secuencia de Diseño . . . . .	6
2.4. Prototipado de Interfaz de Usuario . . . . .	7
<b>3. Diseño del Software</b>	<b>12</b>
3.1. Diagrama de Clases . . . . .	12
3.2. Diagrama de Paquetes . . . . .	13
<b>4. Modelo de Datos</b>	<b>15</b>
4.1. Diagrama Entidad-Relación . . . . .	15
<b>5. Requisitos</b>	<b>16</b>
5.1. Requisitos Funcionales . . . . .	16
5.2. Requisitos No Funcionales . . . . .	17
<b>6. Patrones de Diseño</b>	<b>17</b>
6.1. Patrones GRASP: General Responsibility Assignment <i>Software Prin-</i> <i>ciples</i> . . . . .	18
6.2. Patrones GoF: Gang of Four - <i>Software Desing Patterns</i> . . . . .	19
6.3. Oportunidades de Mejora . . . . .	19
<b>7. Nueva Funcionalidad: Sistema de Recompensas</b>	<b>19</b>
7.1. Diagrama de Clases del Módulo . . . . .	20
<b>8. Relaciones y Justificación</b>	<b>20</b>
<b>9. Conclusión</b>	<b>21</b>

## 1. Introducción

El presente trabajo práctico integrador tiene como objetivo aplicar los conceptos de Ingeniería de Software que vimos durante la materia en el desarrollo de un sistema realista. El proyecto consiste en el diseño e implementación de un software para la gestión de un gimnasio, llamado «Sport», que cuenta con varias sucursales en la provincia de Mendoza.

A lo largo del trabajo fuimos pasando por distintas etapas: primero el análisis de requisitos, después la elaboración de modelos UML (casos de uso, diagramas de clases, diagramas de secuencia), y finalmente la implementación en código. En la práctica notamos que los modelos iniciales sirvieron de guía, pero varias cosas cambiaron en la implementación final, lo cual nos parece normal ya que fue nuestro primer proyecto completo y porque el software siempre evoluciona durante el desarrollo.

El sistema que construimos permite gestionar socios, usuarios y sus roles, la cobranza de cuotas mensuales, la generación de facturas, la gestión de deudas y también rutinas de entrenamiento que los profesores asignan a los socios. Además, pensamos y agregamos una funcionalidad adicional: un módulo de recompensas por puntos, que busca motivar a los socios a asistir, pagar en término e invitar a nuevos miembros.

El enfoque de este informe es mostrar tanto los modelos previos como la implementación resultante, explicando las decisiones de diseño, los patrones aplicados y la experiencia de trabajo en equipo.

## 2. Caso de Uso Crítico: Pago de Cuotas

### 2.1. Diagrama de Caso de Uso

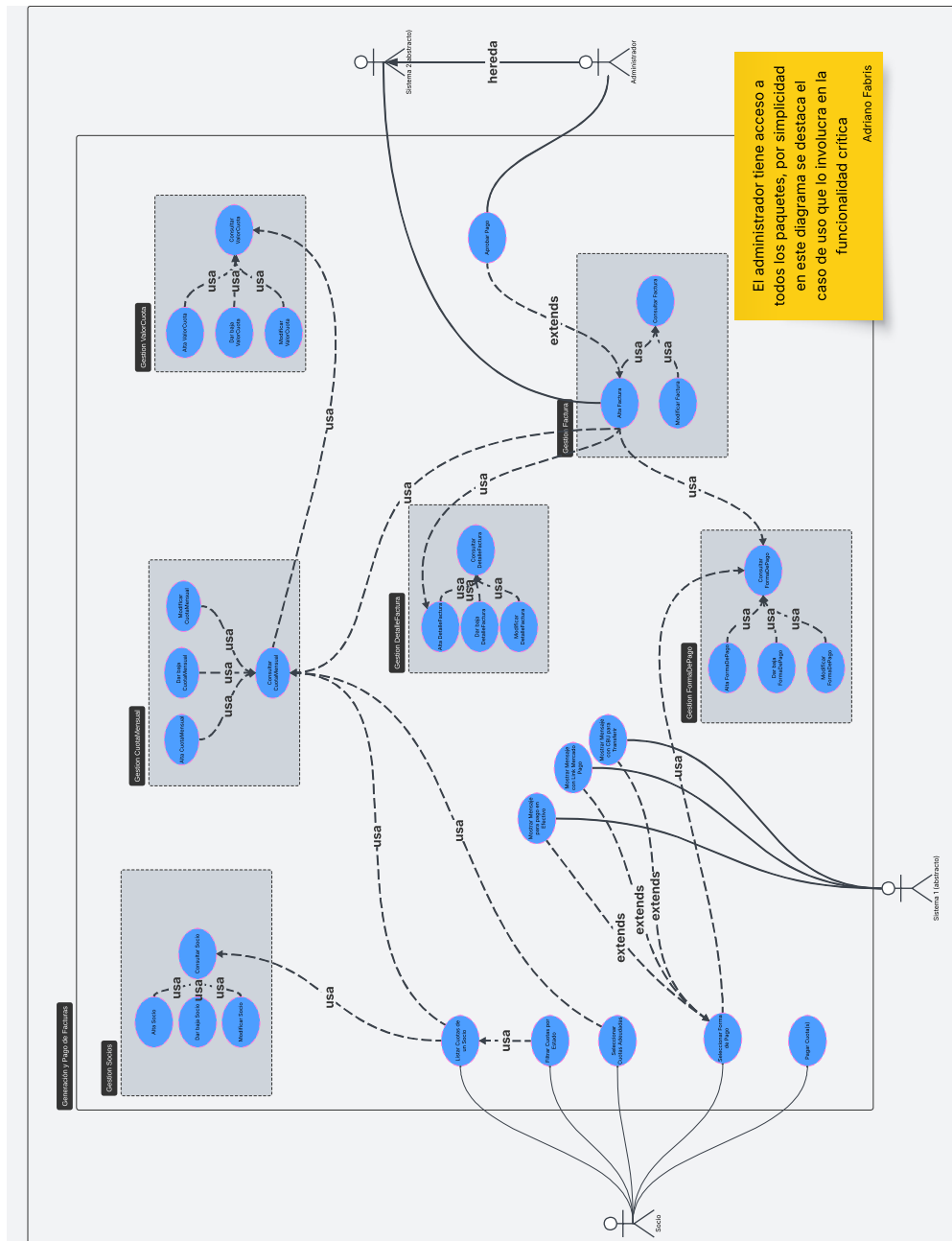


Figura 1: Diagrama de Caso de Uso

**Enlace:** [https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view\\_items=jFBTfsF-ByK\\_&invitationId=inv\\_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b](https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view_items=jFBTfsF-ByK_&invitationId=inv_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b)

## 2.2. Escenario de Caso de Uso

ESCENARIO DE CASO DE USO #1	
Prioridad	9/10 (muy alta)
Nombre	Pago de cuota con su generación de factura y forma de pago
Descripción	Un socio quiere pagar una o más cuotas
Actores principales	Socio Administrador
Actores secundarios (si aplica)	Sistema (abstracto)
Precondiciones	El socio debe existir, tener un estado activo, y tener una o más cuotas adeudadas El sistema debe permitir pagos mediante efectivo, transferencia o billetera virtual (Mercado Pago)
Postcondiciones	El socio habrá pagado una o más cuotas adeudadas El sistema habrá generado una factura para las cuotas seleccionadas y pagadas por el socio
Flujo principal de eventos	<ol style="list-style-type: none"> <li>1. El socio accede al listado de cuotas correspondientes a su perfil</li> <li>2. El socio selecciona las cuotas adeudadas que pagará</li> <li>3. El socio selecciona un método de pago</li> <li>4. El socio efectúa el pago</li> <li>5. El administrador aprueba el pago</li> <li>6. El sistema genera la factura correspondiente a las cuotas pagadas</li> </ol>
Flujos alternativos / excepciones	(A1) El socio puede filtrarlas según su estado: adeudada/pagada (A3.1) Si el método de pago es en efectivo el sistema muestra: "Acérquese a mesa de entrada" (A3.2) Si el método de pago es por transferencia el sistema muestra el CBU destino del gimnasio (A3.3) Si el método de pago es por Mercado Pago el sistema muestra un link de pago
Requisitos especiales	El link de pago para una cuota cuyo pago se efectuará por Mercado Pago tiene un tiempo de expiración predeterminado
Frecuencia de uso	Se estima que se ejecutará n/2 veces entre el 1er y 5to día del correspondiente mes, y el resto de veces en el período restante, n es el número de socios activos en el mes previo.
Notas adicionales	-

Figura 2: Escenario de Caso de Uso

Enlace: [https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view\\_items=5aMRDlw8F6nU&invitationId=inv\\_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b](https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view_items=5aMRDlw8F6nU&invitationId=inv_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b)

## 2.3. Diagrama de Secuencia de Diseño

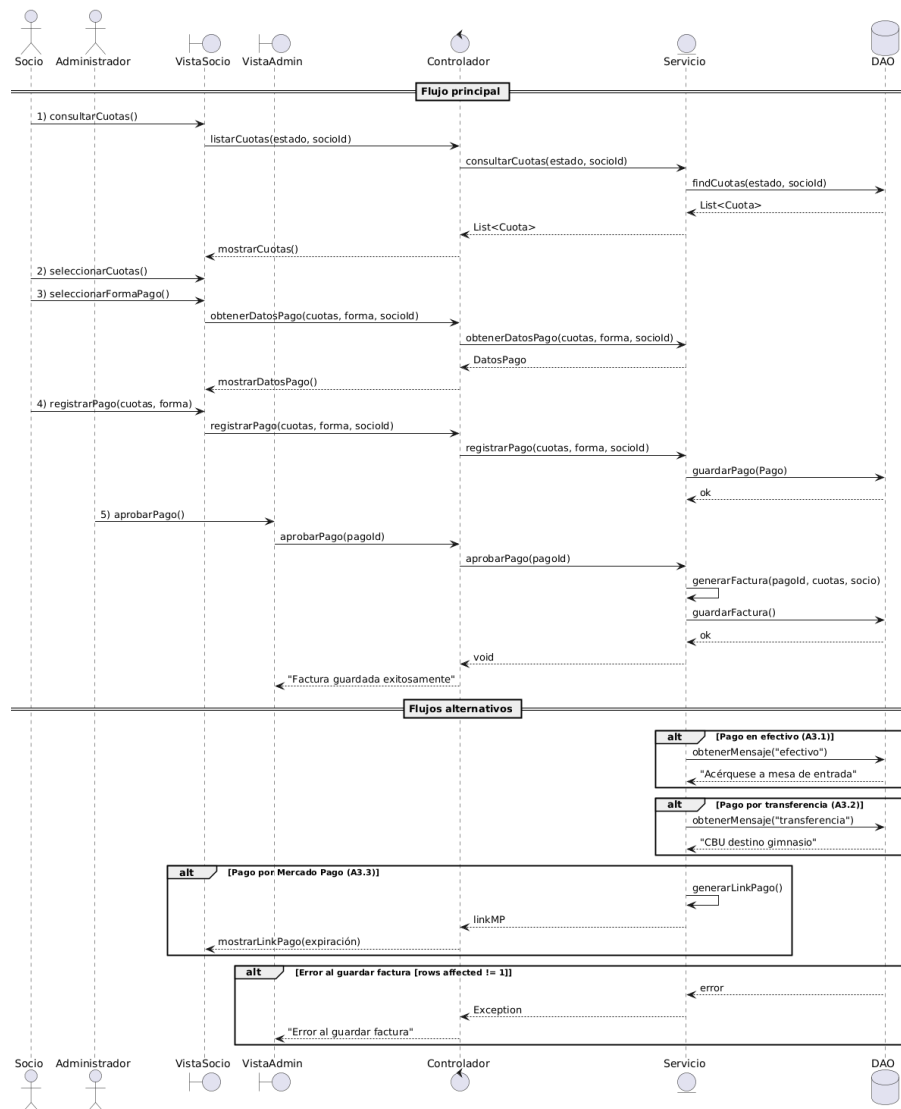


Figura 3: Diagrama de Secuencia de Diseño

Enlace: [https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view\\_items=fCKTkNhf-5qu&invitationId=inv\\_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b](https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view_items=fCKTkNhf-5qu&invitationId=inv_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b)

## 2.4. Prototipado de Interfaz de Usuario

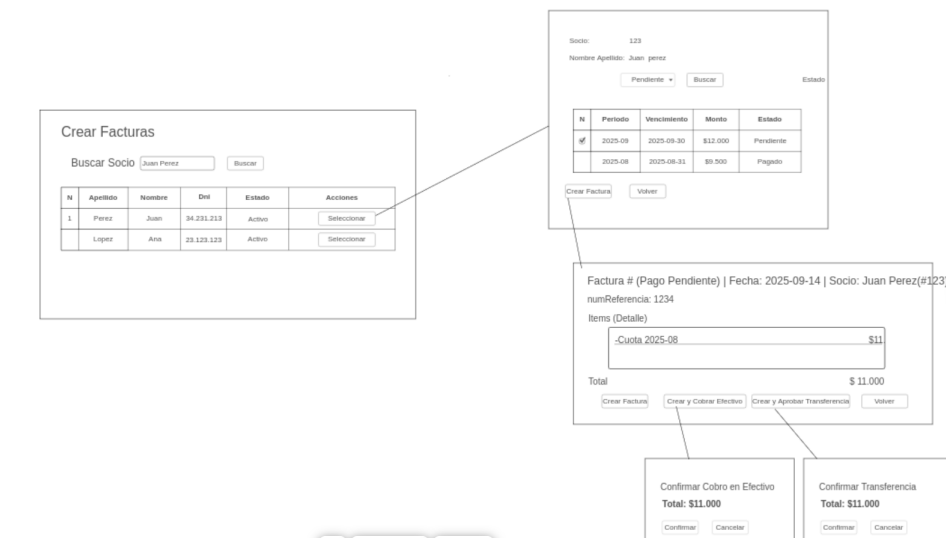


Figura 4: Prototipado 1/3

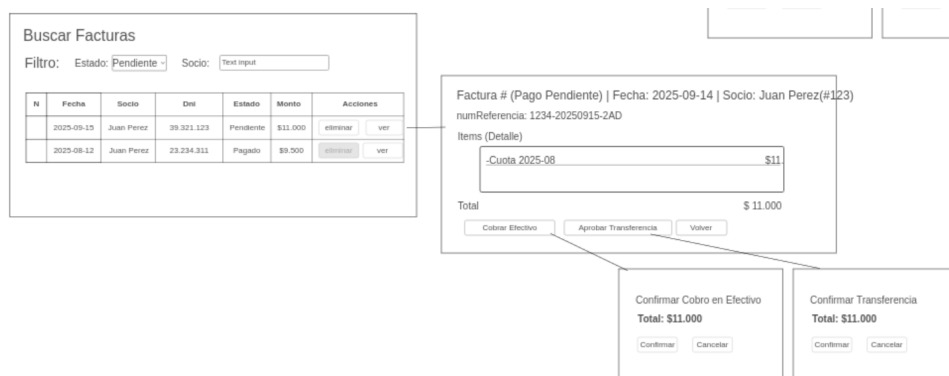


Figura 5: Prototipado 2/3

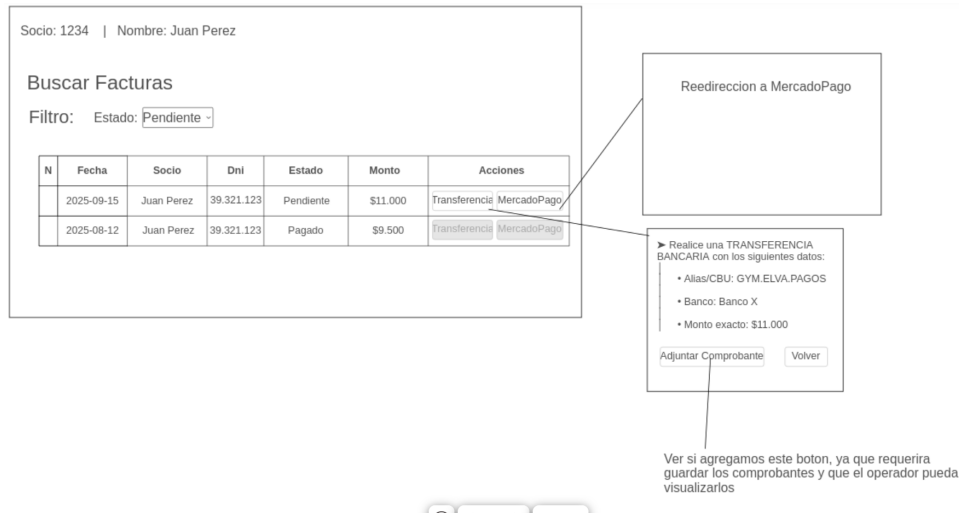


Figura 6: Prototipado 3/3

Enlace: <https://ninjamock.com/s/TVS89Lx>

### Mis Cuotas

Seleccione las cuotas que desea pagar

Desde:  Hasta:

	Mes-Año	Fecha vencimiento	Valor	Estado
<input type="checkbox"/>	JUNIO-2025	2025-06-10	400.00	ADEUDADA
<input type="checkbox"/>	JULIO-2025	2025-07-10	400.00	ADEUDADA
<input type="checkbox"/>	AGOSTO-2025	2025-08-10	500.00	ADEUDADA
<input type="checkbox"/>	SEPTIEMBRE-2025	2025-09-10	600.00	ADEUDADA

**Total a pagar: 0.00**

Figura 7: Interfaz Resultante 1/7



Mi App

Inicio Rutinas Deuda

juarp Salir

### Pago en Efectivo

Diríjase a la ventanilla para realizar el pago

- Diríjase a la ventanilla/puerta de entrada para realizar el pago en efectivo
- Presente su número de socio al personal de cobro

Aceptar

Figura 8: Interfaz Resultante 2/7

## Pago por Transferencia Bancaria

Sigue los pasos para completar tu pago

Información de pago

**Nro de socio:** 1001

**Total a Abonar:** 600.00

Detalles de la Transferencia

- Alias/CBU:** GYM.ELVA.PAGOS
- Banco:** Banco X
- Monto exacto:** 600.00

Una vez realizada la transferencia, subí el comprobante.

Un operador validará el pago y confirmará la factura.

Pendiente

Figura 9: Interfaz Resultante 3/7

## Factura 3

Emitida el 2025-09-25

Pago confirmado correctamente

### RESUMEN

Socio

Perez, Juan

Estado

**PAGADA**

Forma de pago

EFFECTIVO

Mes	Año	Importe
MARZO	2025	300.00
ABRIL	2025	300.00
MAYO	2025	400.00
<b>Total</b>		<b>1000.00</b>

Figura 10: Interfaz Resultante 4/7



## Pago Online

Procesá tu pago de manera rápida y segura con Mercado Pago

### Información de pago

Nro de socio: 1001

Total a Abonar: 600.00



Paga de forma segura

Figura 11: Interfaz Resultante 5/7

### Factura 3

Emitida el 2025-09-25

[← Volver](#)

#### RESUMEN

Socio

Perez, Juan

Estado

SIN\_DEFINIR

Mes	Año	Importe
MARZO	2025	300.00
ABRIL	2025	300.00
MAYO	2025	400.00
<b>Total</b>		<b>1000.00</b>

#### Confirmar pago manual

Forma de pago

EFFECTIVO

Observaciones

Ej: Caja principal

Comprobante / N° operación

Opcional

[Confirmar pago](#)

Figura 12: Interfaz Resultante 6/7

Mi App

Dashboard

Socios

Rutinas

Empleados

Cuota

admin [Salir](#)

NAVEGACIÓN

[Inicio](#)
[Socios](#)
[Facturas](#)
[Rutinas](#)
[Empleados](#)
[Cuota](#)
[Mensajes](#)

SISTEMA

[Empresas](#)
[Sucursales](#)

Facturas

Nueva

-- Socio --

-- Estado --

dd/mm/aaaa

dd/mm/aaaa

Filtrar

#	Fecha	Socio	Total	Estado	
1	2025-09-25	Perez, Juan	8000.00	PAGADA	<a href="#">Ver</a>
2	2025-09-25	Quinteros, Maria	4000.00	SIN_DEFINIR	<a href="#">Ver</a>
3	2025-09-25	Perez, Juan	1000.00	SIN_DEFINIR	<a href="#">Ver</a>

Figura 13: Interfaz Resultante 7/7

### 3. Diseño del Software

### 3.1. Diagrama de Clases

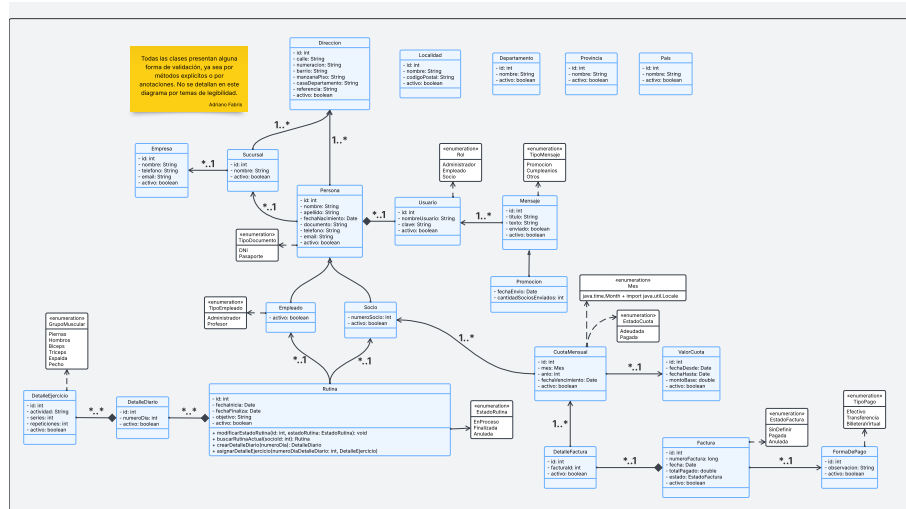


Figura 14: Diagrama de Clases del Dominio - Modelo

**Enlace:** [https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view\\_items=C7BT8wLo05My&invitationId=inv\\_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b](https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view_items=C7BT8wLo05My&invitationId=inv_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b)

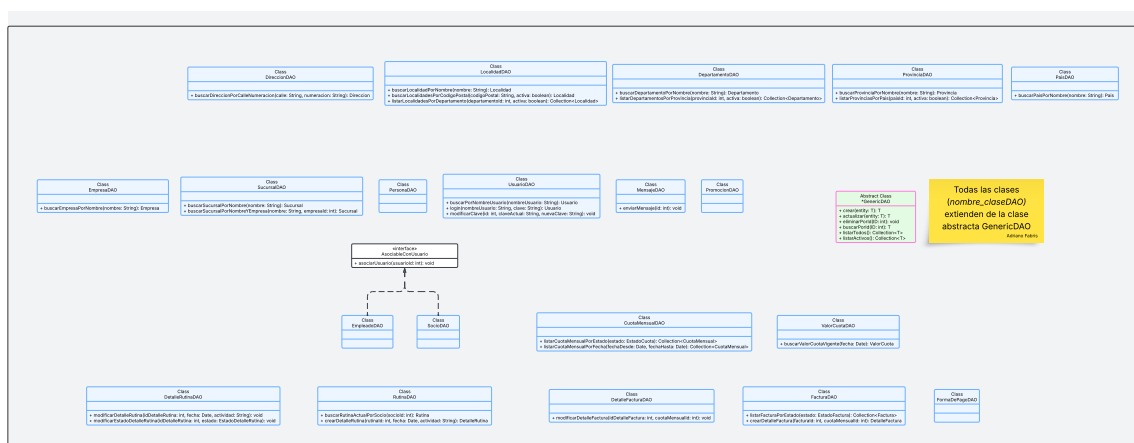


Figura 15: Diagrama de Clases del Dominio - DAO

**Enlace:** [https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view\\_items=p-BTdY7y-Yqd&invitationId=inv\\_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b](https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view_items=p-BTdY7y-Yqd&invitationId=inv_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b)

### 3.2. Diagrama de Paquetes

img/packages.pdf

Figura 16: Diagrama de Paquetes

**Enlace:** [https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?viewport\\_loc=-2087%2C14800%2C2962%2C2204%2C0\\_0&invitationId=inv\\_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b](https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?viewport_loc=-2087%2C14800%2C2962%2C2204%2C0_0&invitationId=inv_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b)

## 4. Modelo de Datos

### 4.1. Diagrama Entidad-Relación

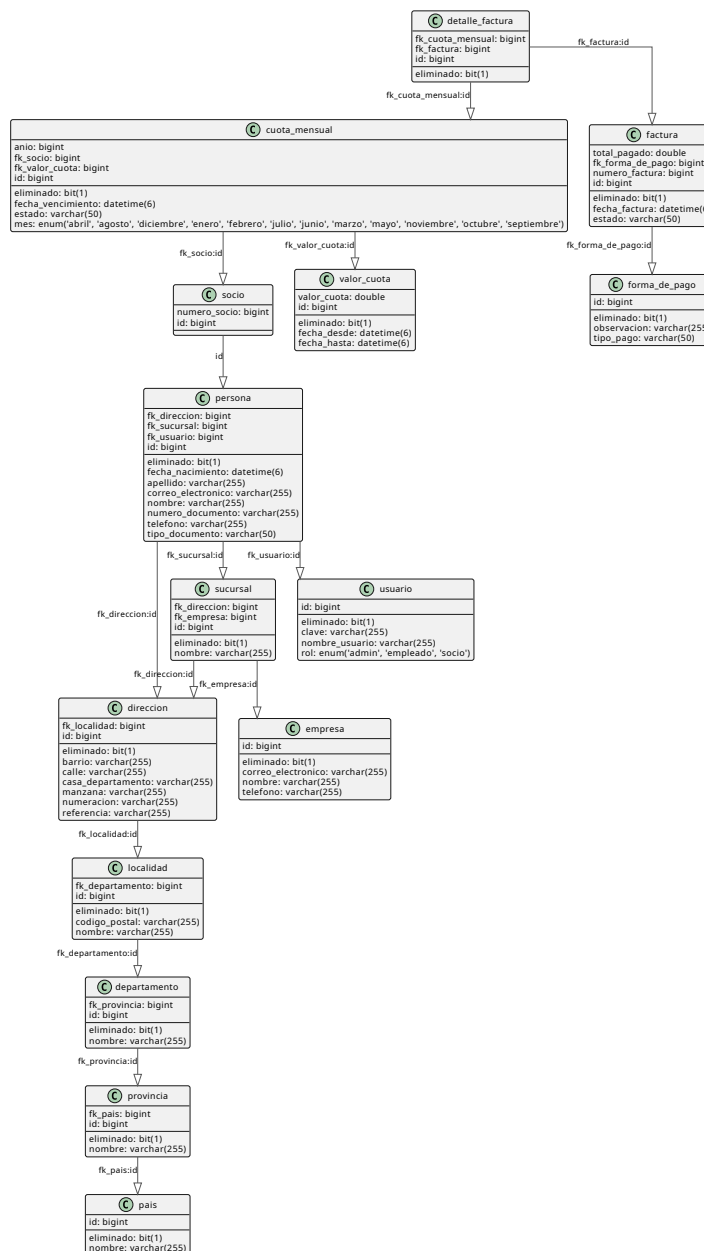


Figura 17: Diagrama Entidad-Relación

**Enlace:** [https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view\\_items=4yKT0Mf.aIp0&invitationId=inv\\_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b](https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view_items=4yKT0Mf.aIp0&invitationId=inv_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b)

## 5. Requisitos

### 5.1. Requisitos Funcionales

Durante el análisis inicial, a partir del enunciado que nos dieron, detectamos las siguientes funcionalidades que el sistema debía cubrir. Las enumeramos como requisitos funcionales (RF):

RF1: Alta, consulta, modificación y baja de socios/as.

RF2: Alta, consulta, modificación y baja de usuarios del sistema con sus respectivos roles.

RF3: Gestión del cobro de la cuota mensual.

RF4: Gestión de la deuda de los socios/as.

RF5: Generación y envío de campañas promocionales por correo electrónico.

RF6: Envío automático de saludos por cumpleaños a los socios/as.

RF7: Gestión y seguimiento de rutinas de entrenamiento que los profesores entregan a los socios/as.

RF8: Control de perfiles de usuario con distintos niveles de acceso:

- Administrador: acceso total.
- Empleado: acceso casi total, salvo gestión de usuarios y alta de valor de cuota.
- Socio/a: acceso a su rutina, pago de cuota mensual (Mercado Pago) e informe de deuda.

RF9: Visualización de la rutina diaria y posibilidad de marcar si fue completada.



## 5.2. Requisitos No Funcionales

Por otro lado, los requisitos no funcionales los fuimos definiendo nosotros en base a lo que veíamos necesario:

- **Usabilidad:** incorporamos una interfaz responsive por medio de una plantilla de Bootstrap, esto nos simplificó el desarrollo y permite que el sistema se use tanto en PC como en celular sin problemas.
- **Seguridad:** se incorporó autenticación y autorización mediante acceso protegido por roles (administrador, operador, socio); y el cifrado bcrypt de contraseñas para la protección de datos.
- **Portabilidad:** se hizo uso de ORM para facilitar migraciones hacia distintas bases de datos.
- **Rendimiento:** por el momento no se efectuaron pruebas de carga, sin embargo el sistema responde rápido en los casos que probamos (pago, consultas, alta de socio). Lo mínimo que pedimos es que las operaciones comunes respondan “en un tiempo razonable”.
- **Mantenibilidad y Escalabilidad:** intentamos que el diseño aguante crecimiento. Un ejemplo es la herencia entre Usuario, Socio y Empleado, que permitiría agregar más tipos de usuarios si el gimnasio crece. Un ejemplo de mantenibilidad se presenta en el módulo de domicilio (país, provincia, localidad, dirección), que hace más fácil escalar a distintos lugares.

## 6. Patrones de Diseño

En esta sección analizamos los patrones de diseño aplicados en el proyecto. Primero repasamos los patrones GRASP, y luego los GoF, mostrando en qué clases y paquetes aparecen. La idea es dejar documentado qué principios seguimos y cómo se ven reflejados en la implementación.

## 6.1. Patrones GRASP: General Responsibility Assignment *Software Principles*

- **Controller:** los controladores web reciben las peticiones y delegan la lógica. Ejemplo: `SociosController` (paquete `controller/`) que orquesta la vista y llama a `SocioService`.
- **Information Expert:** cada clase que tiene la información necesaria se encarga de usarla. Ejemplos: `FacturaServiceImpl` (cálculo de totales y actualización de estados de cuotas) y `ValorCuotaServiceImpl` (validación de reglas de negocio).
- **Creator:** las clases que agrupan datos crean a sus objetos relacionados. Ejemplo: `SocioServiceImpl` crea `Direccion`, `Usuario` y `Socio`.
- **Low Coupling / High Cohesion:** se respetan las capas MVC (controlador, servicio, repositorio, mapper, DTO). Así se logra poco acoplamiento y cohesión alta en cada módulo.
- **Indirection:** se introducen intermediarios para desacoplar. Ejemplo: `NotificationSender` como interfaz para distintos métodos de notificación.
- **Polymorphism:** se usan jerarquías e interfaces para variar comportamientos. Ejemplos: la jerarquía de pagos (`Pago`, `PagoOnline`, `PagoTransferencia`) y la interfaz `NotificationSender`.
- **Pure Fabrication:** se crean clases auxiliares fuera del dominio real para mantener cohesión. Ejemplo: los mapeadores DTO (`SocioMapper`, `FacturaMapper`) y objetos como `PagoCommand`.
- **Protected Variations:** los puntos de cambio se encapsulan. Ejemplos: enums (`TipoPago`, `EstadoFactura`, `EstadoCuota`), configuración de seguridad (`SecurityConfig`) y configuración de MercadoPago (`PagosConfig`).

## 6.2. Patrones GoF: Gang of Four - *Software Desing Patterns*

- **Builder**: se usa para crear objetos complejos de forma fluida, con Lombok y builders explícitos. Ejemplos: `Factura.builder()` y DTOs de pago con `@SuperBuilder`.
- **Strategy**: define una familia de algoritmos intercambiables. Ejemplos: `PasswordEncoder` en seguridad, y la interfaz `NotificationSender` con implementaciones como `ConsoleNotificationSender`.
- **Template Method**: una superclase define el algoritmo y las subclasses redefinen pasos específicos. Ejemplo: `RoleBasedAuthenticationSuccessHandler`, que extiende `SavedRequestAwareAuthenticationSuccessHandler`.
- **Factory Method**: encapsula la creación de objetos. Ejemplos: `PasswordEncoderFactories.cr` y métodos `@Bean` en configuraciones (`PagosConfig`).
- **Chain of Responsibility**: cadena de objetos que procesan una petición. Ejemplo: `SecurityFilterChain` en Spring Security, donde los filtros se aplican secuencialmente.

## 6.3. Oportunidades de Mejora

- Aplicar **Strategy** para procesar pagos según su tipo (efectivo, transferencia, billetera).
- Usar **State** para encapsular las transiciones de estados en facturas y cuotas.
- Implementar un **Adapter** para abstraer la integración con MercadoPago.

## 7. Nueva Funcionalidad: Sistema de Recompensas

Como parte del trabajo, incorporamos un módulo adicional al sistema: un **sistema de recompensas por puntos**. La idea surge de la necesidad de motivar a los socios y promover la fidelización dentro del gimnasio. El funcionamiento es sencillo: los socios acumulan puntos al realizar distintas acciones, como asistir regularmente, pagar

sus cuotas en término o invitar a nuevos socios. Luego, estos puntos pueden canjearse por descuentos en la cuota, productos promocionales o beneficios especiales.

Con esta funcionalidad se busca no solo mejorar la experiencia del usuario, sino también generar un incentivo para que los socios permanezcan activos y colaboren con el crecimiento del gimnasio. Además, el sistema es flexible y podría ampliarse en el futuro con nuevas reglas o tipos de recompensas.

## 7.1. Diagrama de Clases del Módulo

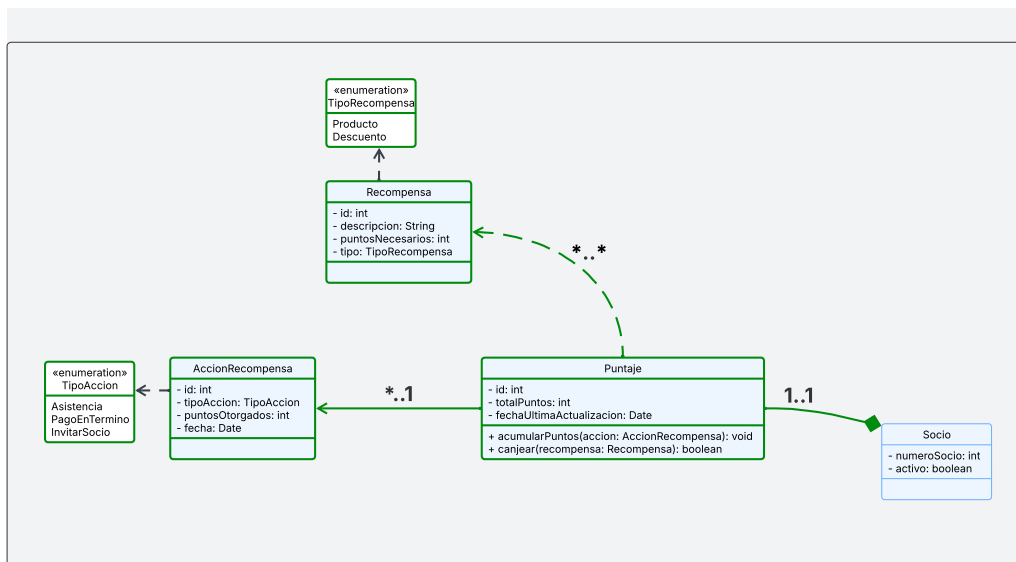


Figura 18: Diagrama de Clases de la Nueva Funcionalidad

**Enlace:** [https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view\\_items=5YJTan\\_uGaTs&invitationId=inv\\_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b](https://lucid.app/lucidchart/75a89fc9-ae9e-469b-a08a-86d133243093/edit?view_items=5YJTan_uGaTs&invitationId=inv_ef5d30df-b4d8-4ee1-bddc-cb6337fbbc2b)

## 8. Relaciones y Justificación

En el módulo de recompensas se definieron las siguientes relaciones principales:

- **Socio – Puntaje:** asociación 1 a 1, cada socio posee un único puntaje acumulado.
- **Puntaje – AccionRecompensa:** asociación 1 a N, un puntaje se forma a partir de múltiples acciones realizadas por el socio.
- **Puntaje – Recompensa:** relación de dependencia, ya que sólo interviene al momento del canje de una recompensa.
- **Enumeraciones:** *TipoRecompensa* y *TipoAccion* funcionan como dependencias de *Recompensa* y *AccionRecompensa*, tipificando su comportamiento.

## 9. Conclusión

Este trabajo nos permitió experimentar con un ciclo completo de desarrollo mediante trabajo en equipo. Una cosa que aprendimos es que los diagramas iniciales sirven de guía, pero después en la implementación cambian varias cosas, y está bien que pase así porque el software evoluciona.

En cuanto a la forma de trabajo, nos dividimos en dos grupos: unos se enfocaron en la implementación y otros en la documentación. De todas formas la división nunca fue total: los que programaban consultaban la documentación y los que documentaban también ayudaban a probar o entender la implementación para basar en ellas la documentación o los futuros diagramas. Eso hizo que el trabajo se mantuviera coordinado y que todos tuviéramos una idea general del sistema.

En resumen, fue un proyecto que nos ayudó a poner en práctica conceptos de la materia (como diagramas UML, requisitos, patrones de diseño, código) y también a organizarnos como equipo de desarrollo por primera vez.