



# Integrador Parte N°1

## *Software para Gimnasio*

---

### Grupo 2: ElvaStudio

- Trinidad Perea
- Perla Valerio
- Giovanni Azurduy
- Adriano Fabris



← QR al repositorio GitHub:  
código + informe con links hacia  
la documentación

INDICE	Introducción	1
	Caso de Uso Crítico	2
	Diseño del Software	3
	Modelo de Datos	4
	Diagrama de Paquetes	5
	Requisitos F y No Funcionales	6
	Patrones de Diseño Aplicados	7
	Software en Funcionamiento	8
	Nueva Funcionalidad	9
	Conclusión	10

# Introducción

---

Desarrollo de un sistema de gestión para el gimnasio “Sport” (Mendoza). Administra socios, usuarios, cuotas, facturas, deudas y rutinas.

Se agregó un módulo de recompensas para motivar la fidelización.

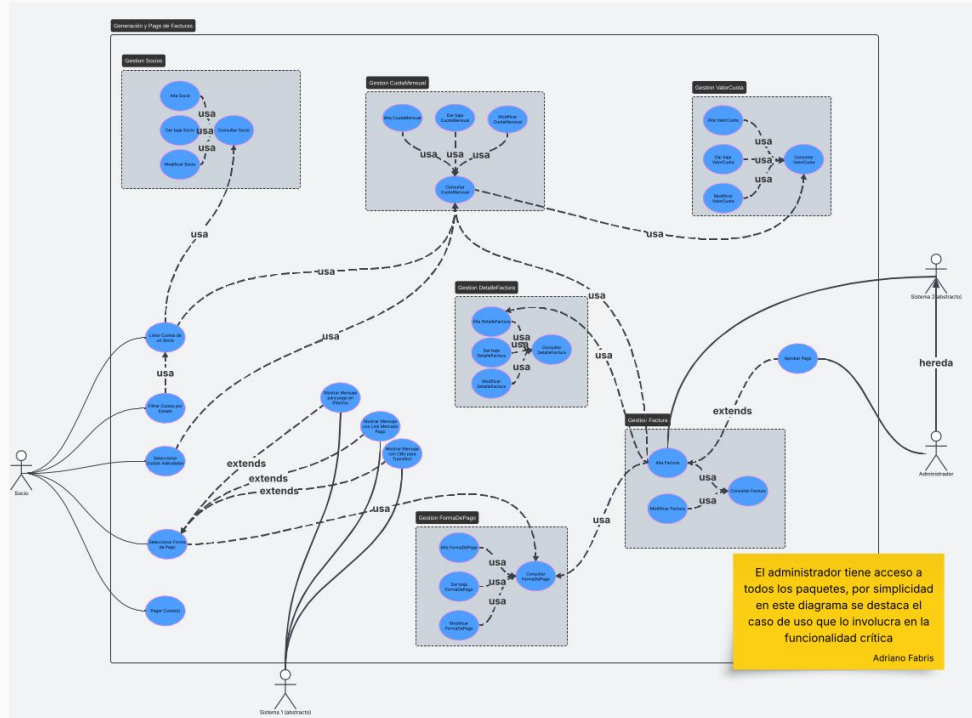
# Caso de Uso Crítico

*“Pago de Cuotas con Generación de Factura y  
Método de Pago”*

---

# Diagrama de Caso de Uso

Veámoslo parte por parte, [en vivo y en directo](#)



# 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	<p>(A1) El socio puede filtrarlas según su estado: adeudada/pagada</p> <p>(A3.1) Si el método de pago es en efectivo el sistema muestra: "Acérquese a mesa de entrada"</p> <p>(A3.2) Si el método de pago es por transferencia el sistema muestra el CBU destino del gimnasio</p> <p>(A3.3) Si el método de pago es por Mercado Pago el sistema muestra un link de pago</p>
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	-

# Prototipado de la UI

---



## Caso de uso crítico

### Crear Facturas

Buscar Socio

N	Apellido	Nombre	Dni	Estado	Acciones
1	Perez	Juan	34.231.213	Activo	<input type="button" value="Seleccionar"/>
	Lopez	Ana	23.123.123	Activo	<input type="button" value="Seleccionar"/>

Socio: 123

Nombre Apellido: Juan perez

Pendiente

Estado

N	Periodo	Vencimiento	Monto	Estado
<input checked="" type="checkbox"/>	2025-09	2025-09-30	\$12.000	Pendiente
	2025-08	2025-08-31	\$9.500	Pagado

Factura # (Pago Pendiente) | Fecha: 2025-09-14 | Socio: Juan Perez(#123)  
numReferencia: 1234

Items (Detalle)

-Cuota 2025-08	\$11.000
----------------	----------

Total

\$ 11.000

Confirmar Cobro en Efectivo

**Total: \$11.000**

Confirmar Transferencia

**Total: \$11.000**

Buscar Facturas

Caso de uso crítico

Buscar Facturas

Filtro: Estado: Pendiente Socio:

N	Fecha	Socio	Dni	Estado	Monto	Acciones	
	2025-09-15	Juan Perez	39.321.123	Pendiente	\$11.000	<span>eliminar</span>	<span>ver</span>
	2025-08-12	Juan Perez	23.234.311	Pagado	\$9.500	<span>eliminar</span>	<span>ver</span>

Factura # (Pago Pendiente) | Fecha: 2025-09-14 | Socio: Juan Perez(#123)  
numReferencia: 1234-20250915-2AD

Items (Detalle)

-Cuota 2025-08	\$11.000
----------------	----------

Total \$ 11.000

Cobrar Efectivo Aprobar Transferencia Volver

Confirmar Cobro en Efectivo

Total: \$11.000

Confirmar Cancelar

Confirmar Transferencia

Total: \$11.000

Confirmar Cancelar

## Caso de uso crítico

Socio: 1234 | Nombre: Juan Perez

### Buscar Facturas

Filtro: Estado: Pendiente ~

N	Fecha	Socio	Dni	Estado	Monto	Acciones	
	2025-09-15	Juan Perez	39.321.123	Pendiente	\$11.000	Transferencia	MercadoPago
	2025-08-12	Juan Perez	39.321.123	Pagado	\$9.500	Transferencia	MercadoPago

Reedireccion a MercadoPago

► Realice una TRANSFERENCIA BANCARIA con los siguientes datos:

- Alias/CBU: GYM.ELVA.PAGOS
- Banco: Banco X
- Monto exacto: \$11.000

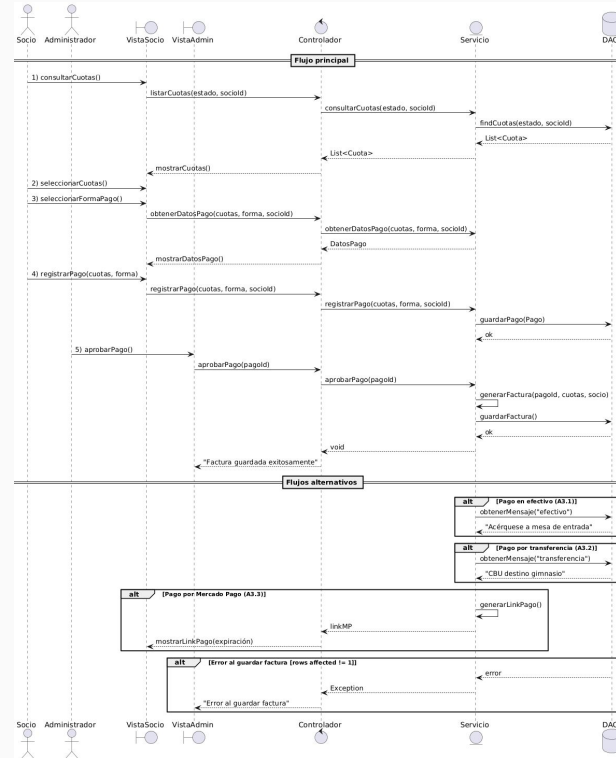
Adjuntar Comprobante

Volver

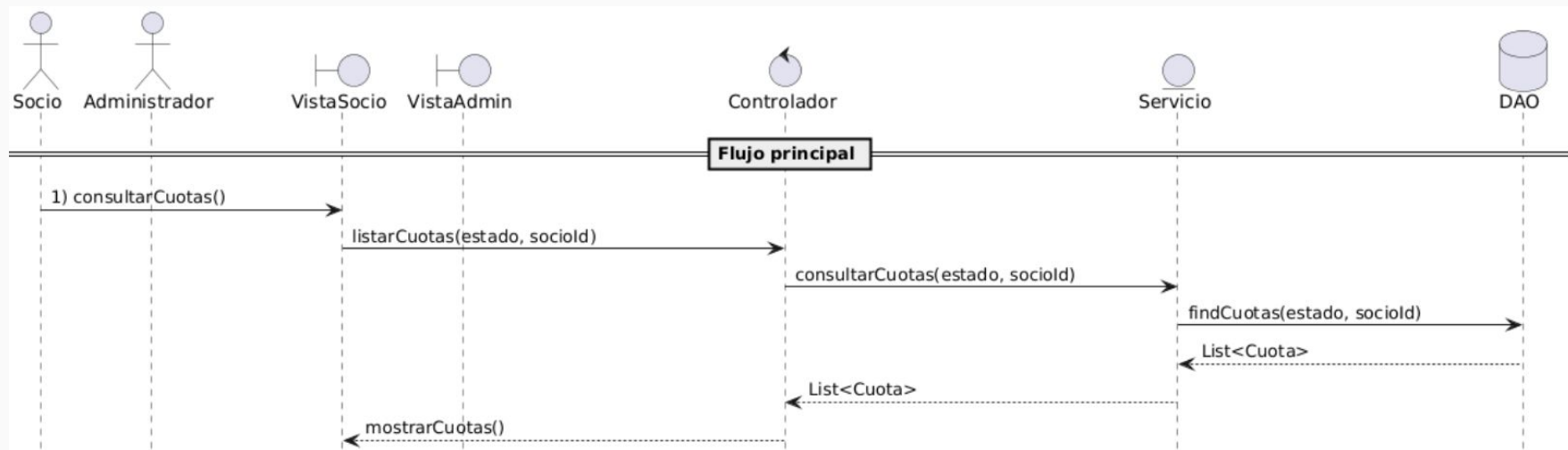
Ver si agregamos este boton, ya que requerira guardar los comprobantes y que el operador pueda visualizarlos

# Diagrama de Secuencia

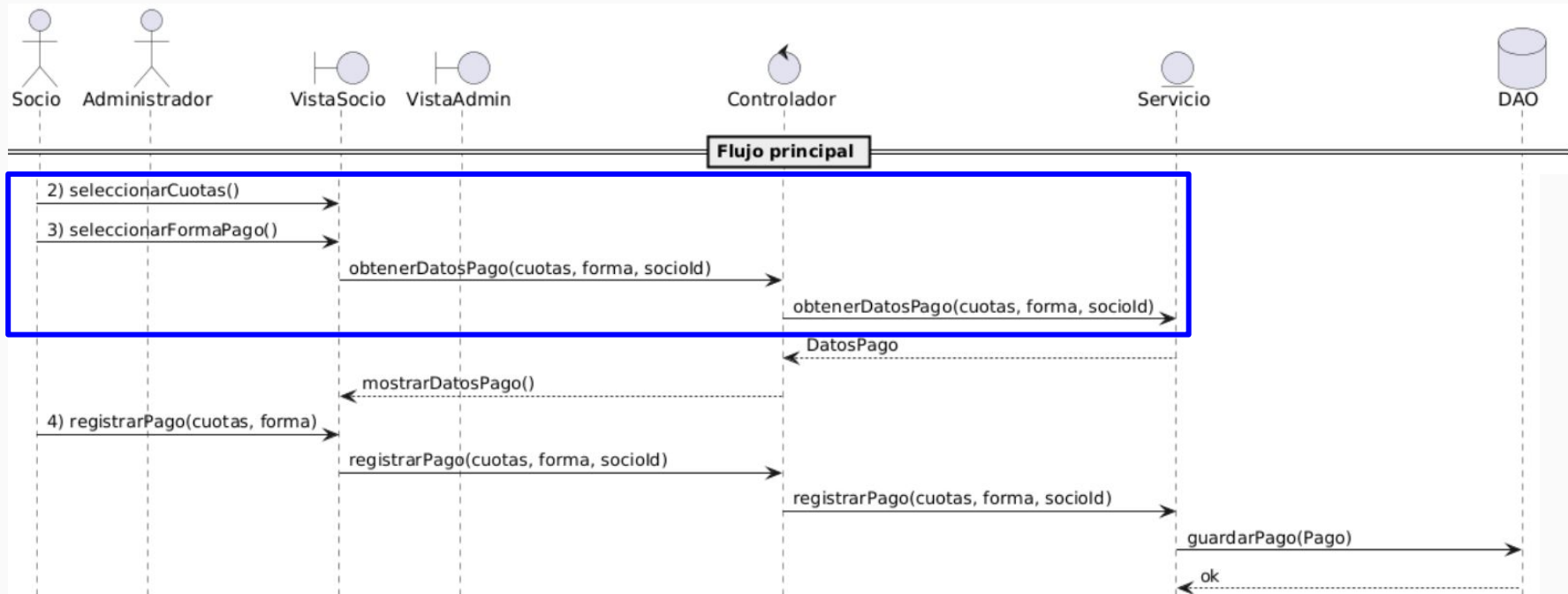
De nuevo, veámoslo de a partes...



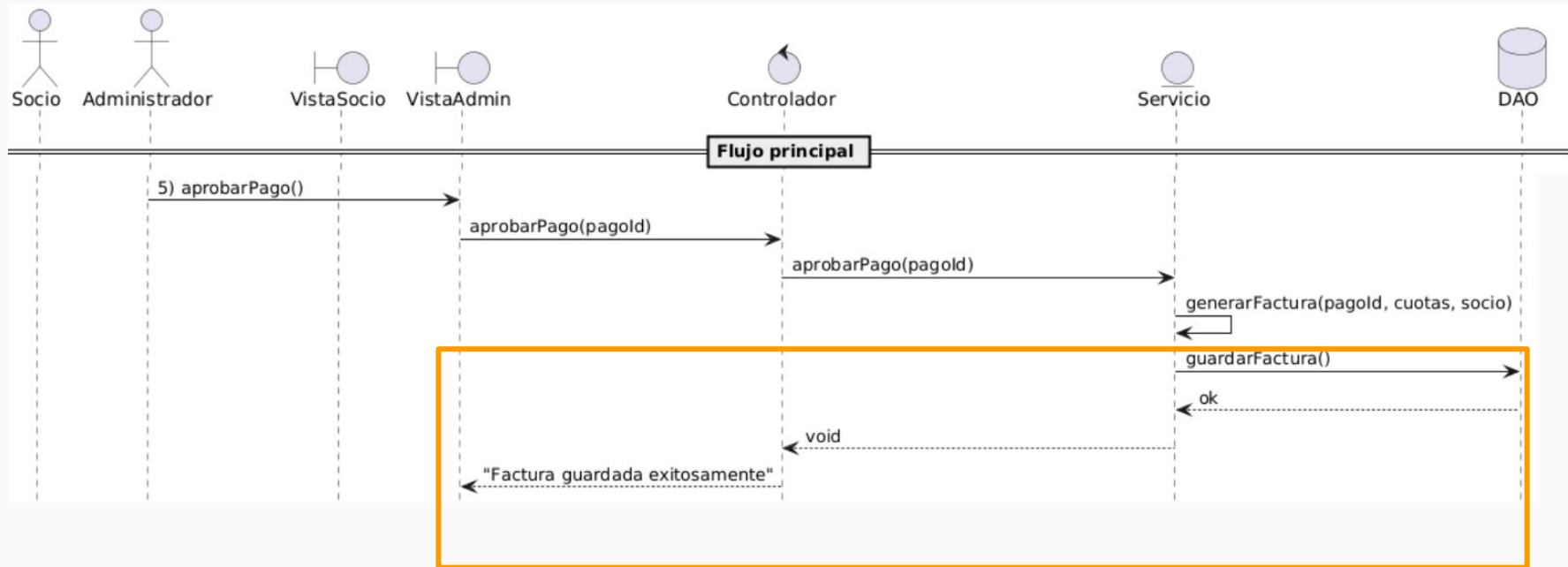
## Caso de uso crítico



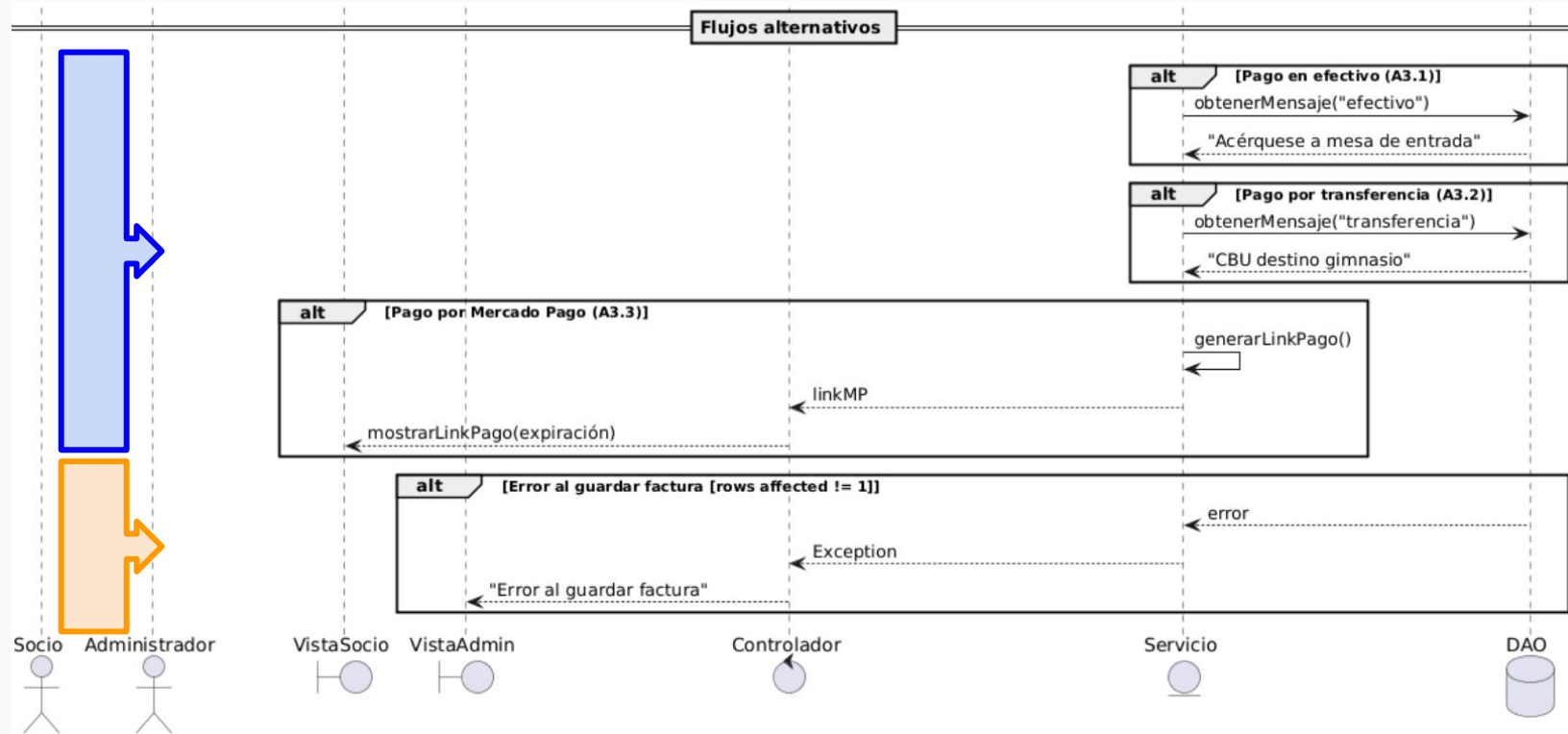
## Caso de uso crítico



## Caso de uso crítico



## Caso de uso crítico





# Diseño del Software

---

A modo de diferenciar las Entidades de sus Servicios y por cuestiones de legibilidad, separamos el diagrama en 2 capas:

1) Entidad

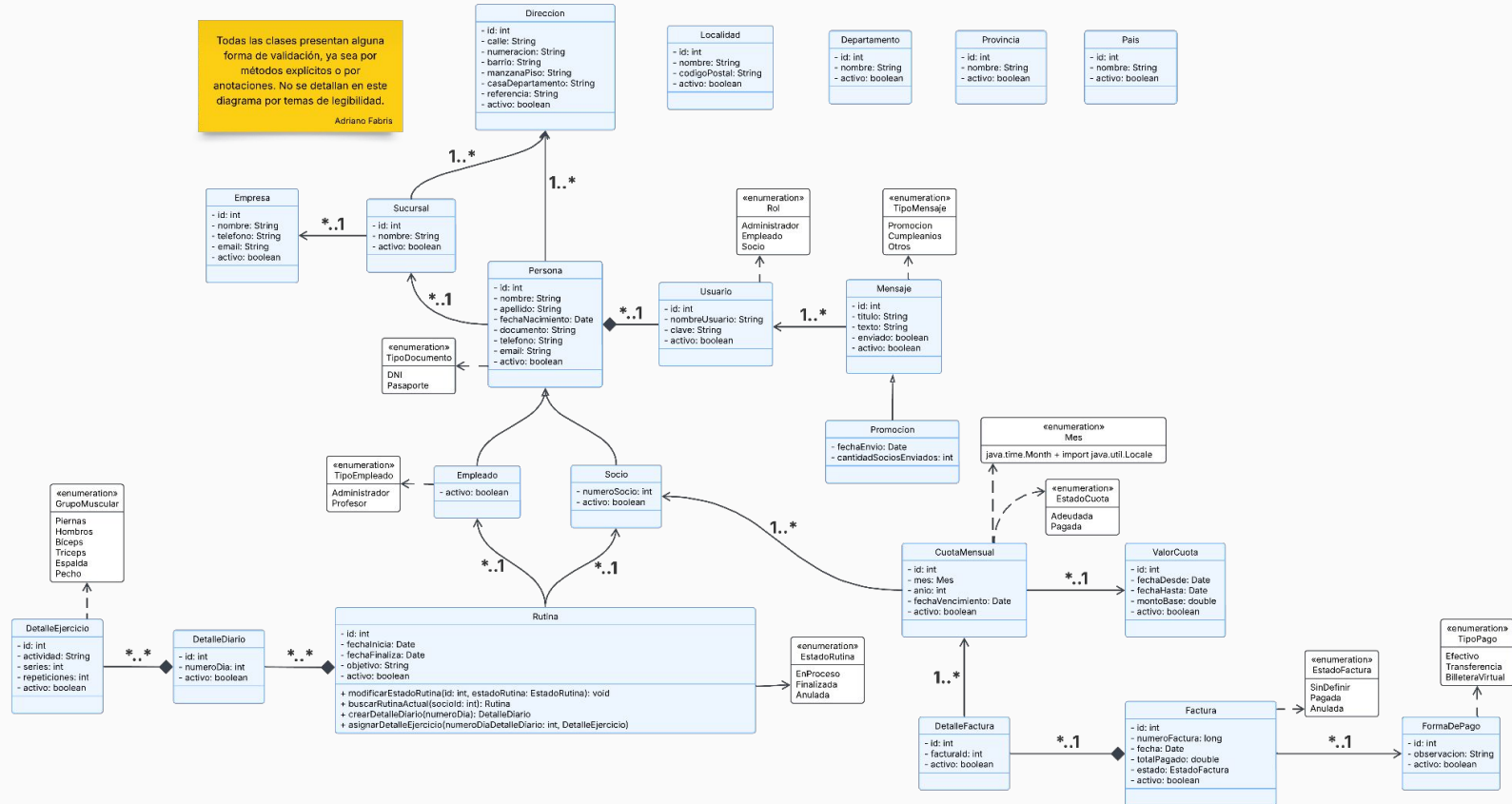
2) DAO

---

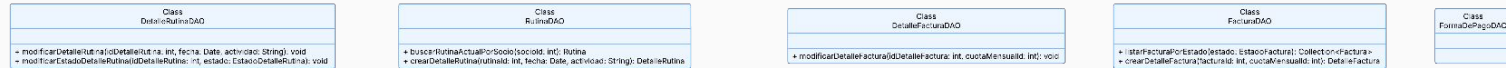
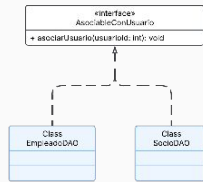
# Entidad

Todas las clases presentan alguna forma de validación, ya sea por métodos explícitos o por anotaciones. No se detallan en este diagrama por temas de legibilidad.

Adriano Fabris



# DAO



Abstract Class  
\*GenericDAO

- + crear(entity: T): T
- + actualizar(entity: T): T
- + eliminar(entity: T): void
- + buscarPorId(id: Int): T
- + listar(entity: T): Collection<T>
- + listarActivos(): Collection<T>

Todas las clases  
(nombre\_claseDAO)  
extienden de la clase  
abstracta GenericDAO

Adriano Fabris

# DAO

Class
DireccionDAO
+ buscarDireccionPorCalleNumeracion(calle: String, numeracion: String): Direccion

Class
LocalidadDAO
+ buscarLocalidadPorNombre(nombre: String): Localidad + buscarLocalidadesPorCodigoPostal(codigoPostal: String, active: boolean): Localidad + listarLocalidadesPorDepartamento(departamentoId: Int, active: boolean): Collection<Localidad>

Class
DepartamentoDAO
+ buscarDepartamentoPorNombre(nombre: String): Departamento + listarDepartamentosPorProvincia(provinciaId: Int, active: boolean): Collection<Departamento>

Class
ProvinciaDAO
+ buscarProvinciaPorNombre(nombre: String): Provincia + listarProvinciasPorPaís(paísId: Int, active: boolean): Collection<Provincia>

Class
PaísDAO
+ buscarPaísPorNombre(nombre: String): País

Class
EmpresaDAO
+ buscarEmpresaPorNombre(nombre: String): Empresa

Class
SucursalDAO
+ buscarSucursalPorNombre(nombre: String): Sucursal + buscarSucursalesPorNombreYEmpresa(nombre: String, empresaId: Int): Sucursal

Class
PersonaDAO

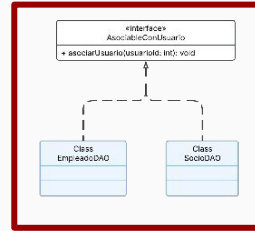
Class
UsuarioDAO
+ buscarPorNombreUsuario(nombreUsuario: String): Usuario + login(nombreUsuario: String, clave: String): Usuario + modificarCuenta(id: Int, claveNuevo: String, nombreCuenta: String): void

Class
MensajeDAO
+ enviarMensaje(id: Int): void

Class
PromocionDAO

Abstract Class
*GenericDAO
+ crear(entity: T): T + actualizar(entity: T): T + eliminarPorId(id: Int): void + buscarPorId(id: Int): T + listarPorId(): Collection<T> + listarActivos(): Collection<T>

Todas las clases  
(nombre\_claseDAO)  
extienden de la clase  
abstracta GenericDAO  
Adriano Fabris



Class
CuentaMensajeDAO
+ listarCuentaMensajePorEstado(estado: EstadoCuenta): Collection<CuentaMensaje> + listarCuentaMensajePorFecha(desde: Date, fechaHasta: Date): Collection<CuentaMensaje>

Class
ValorCuentaDAO
+ buscarValorCuentaPorFecha(fecha: Date): ValorCuenta

Class
DetalleRutinaDAO
+ modificarDetalleRutina(idDetalleRutina: Int, fecha: Date, activo: String): void + modificarEstadoDetalleRutina(idDetalleRutina: Int, estado: EstadoDetalleRutina): void

Class
RutinaDAO
+ buscarRutinaActualPorSocio(socioId: Int): Rutina + crearDetalleRutina(idRutina: Int, fecha: Date, activo: String): DetalleRutina

Class
DetalleFacturaDAO
+ modificarDetalleFactura(idDetalleFactura: Int, cuentaMensajeId: Int): void

Class
FacturaDAO
+ listarFacturaPorEstado(estado: EstadoFactura): Collection<Factura> + crearDetalleFactura(idFactura: Int, cuentaMensajeId: Int): DetalleFactura

Class
FormaDePagoDAO

## DAO

Abstract Class  
\*GenericDAO

- + crear(entity: T): T
- + actualizar(entity: T): T
- + eliminarPorId(ID: int): void
- + buscarPorId(ID: int): T
- + listarTodos(): Collection<T>
- + listarActivos(): Collection<T>

Todas las clases  
(*nombre\_claseDAO*)  
extienden de la clase  
abstracta GenericDAO

Adriano Fabris

Class  
FormaDePagoDAO

«interface»  
AsociableConUsuario

- + asociarUsuario(usuarioId: int): void

Class  
EmpleadoDAO

Class  
SocioDAO

Class  
UsuarioDAO

- + buscarPorNombreUsuario(nombreUsuario: String): Usuario
- + login(nombreUsuario: String, clave: String): Usuario
- + modificarClave(id: int, claveActual: String, nuevaClave: String): void

# Modelo de Datos

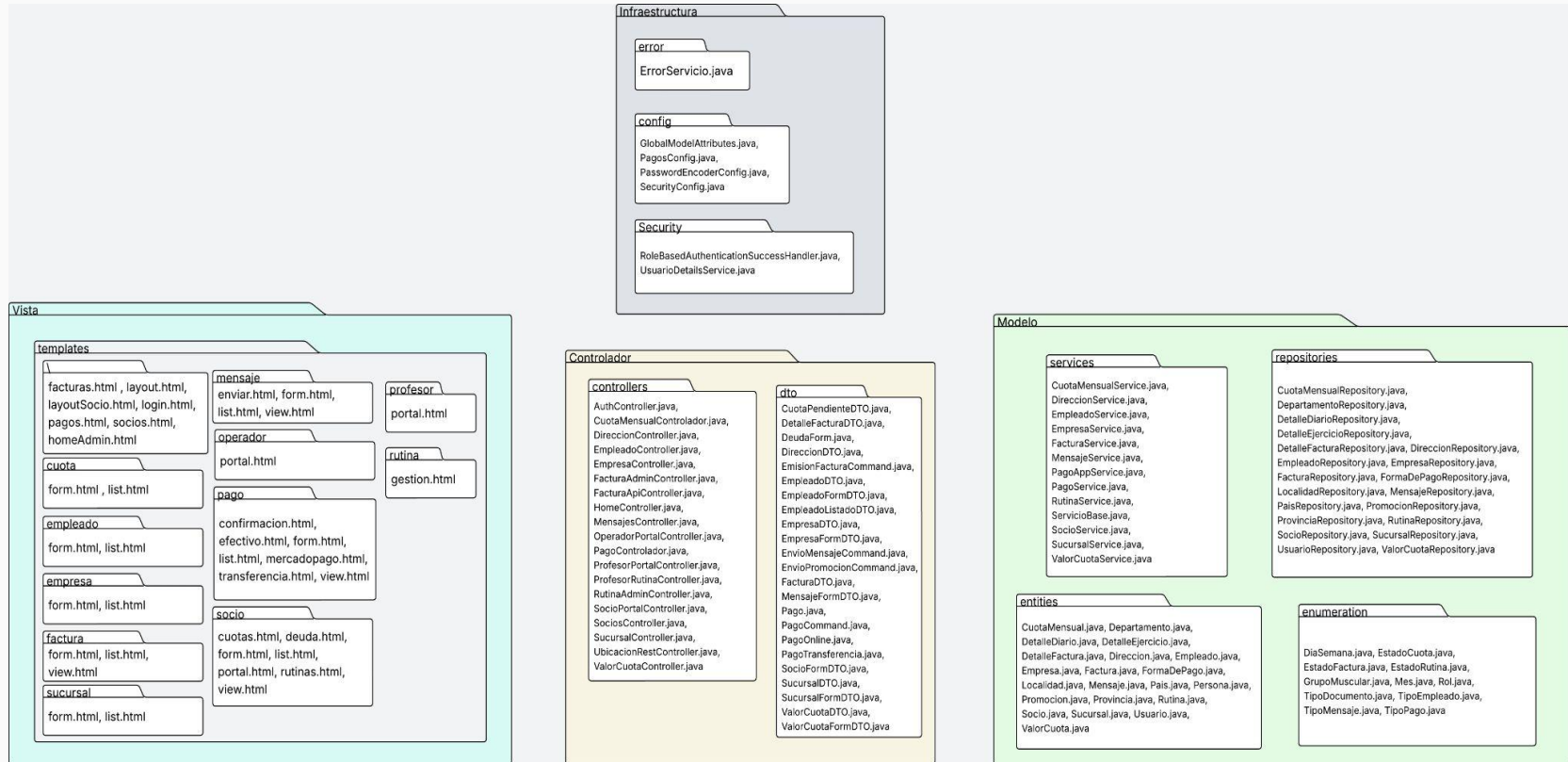
---

# Diagrama de Paquetes

---



# Proyecto



# Proyecto

## Vista

### templates

facturas.html , layout.html,  
layoutSocio.html, login.html,  
pagos.html, socios.html,  
homeAdmin.html

### cuota

form.html , list.html

### empleado

form.html, list.html

### empresa

form.html, list.html

### factura

form.html, list.html,  
view.html

### sucursal

form.html, list.html

### mensaje

enviar.html, form.html,  
list.html, view.html

### operador

portal.html

### pago

confirmacion.html,  
efectivo.html, form.html,  
list.html, mercadopago.html,  
transferencia.html, view.html

### socio

cuotas.html, deuda.html,  
form.html, list.html,  
portal.html, rutinas.html,  
view.html

### profesor

portal.html

### rutina

gestion.html

## Controlador

### controllers

AuthController.java,  
CuotaMensualControlador.java,  
DireccionController.java,  
EmpleadoController.java,  
EmpresaController.java,  
FacturaAdminController.java,  
FacturaApiController.java,  
HomeController.java,  
MensajesController.java,  
OperadorPortalController.java,  
PagoControlador.java,  
ProfesorPortalController.java,  
ProfesorRutinaController.java,  
RutinaAdminController.java,  
SocioPortalController.java,  
SociosController.java,  
SucursalController.java,  
UbicacionRestController.java,  
ValorCuotaController.java

### dto

CuotaPendienteDTO.java,  
DetalleFacturaDTO.java,  
DeudaForm.java,  
DireccionDTO.java,  
EmisionFacturaCommand.java,  
EmpleadoDTO.java,  
EmpleadoFormDTO.java,  
EmpleadoListadoDTO.java,  
EmpresaDTO.java,  
EmpresaFormDTO.java,  
EnvioMensajeCommand.java,  
EnvioPromocionCommand.java,  
FacturaDTO.java,  
MensajeFormDTO.java,  
Pago.java,  
PagoCommand.java,  
PagoOnline.java,  
PagoTransferencia.java,  
SocioFormDTO.java,  
SucursalDTO.java,  
SucursalFormDTO.java,  
ValorCuotaDTO.java,  
ValorCuotaFormDTO.java

# Proyecto

## Modelo

### services

CuotaMensualService.java,  
DireccionService.java,  
EmpleadoService.java,  
EmpresaService.java,  
FacturaService.java,  
MensajeService.java,  
PagoAppService.java,  
PagoService.java,  
RutinaService.java,  
ServicioBase.java,  
SocioService.java,  
SucursalService.java,  
ValorCuotaService.java

### repositories

CuotaMensualRepository.java,  
DepartamentoRepository.java,  
DetalleDiarioRepository.java,  
DetalleEjercicioRepository.java,  
DetalleFacturaRepository.java, DireccionRepository.java,  
EmpleadoRepository.java, EmpresaRepository.java,  
FacturaRepository.java, FormaDePagoRepository.java,  
LocalidadRepository.java, MensajeRepository.java,  
PaisRepository.java, PromocionRepository.java,  
ProvinciaRepository.java, RutinaRepository.java,  
SocioRepository.java, SucursalRepository.java,  
UsuarioRepository.java, ValorCuotaRepository.java

### entities

CuotaMensual.java, Departamento.java,  
DetalleDiario.java, DetalleEjercicio.java,  
DetalleFactura.java, Direccion.java, Empleado.java,  
Empresa.java, Factura.java, FormaDePago.java,  
Localidad.java, Mensaje.java, Pais.java, Persona.java,  
Promocion.java, Provincia.java, Rutina.java,  
Socio.java, Sucursal.java, Usuario.java,  
ValorCuota.java

### enumeration

DiaSemana.java, EstadoCuota.java,  
EstadoFactura.java, EstadoRutina.java,  
GrupoMuscular.java, Mes.java, Rol.java,  
TipoDocumento.java, TipoEmpleado.java,  
TipoMensaje.java, TipoPago.java

## Infraestructura

### error

ErrorServicio.java

### config

GlobalModelAttributes.java,  
PagosConfig.java,  
PasswordEncoderConfig.java,  
SecurityConfig.java

### Security

RoleBasedAuthenticationSuccessHandler.java,  
UsuarioDetailsService.java

# Requisitos Funcionales

- 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: Gestión 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.

# Requisitos No Funcionales

- Usabilidad:

**Interfaz intuitiva:** Diseño claro y consistente para operadores y socios.

**Compatibilidad multiplataforma:** Accesible desde computadoras de escritorio y dispositivos móviles.

- Seguridad:

**Autenticación y autorización:** Acceso protegido con roles (administrador, operador, socio).

**Protección de datos:** Cifrado de contraseña.

- 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).

- Portabilidad:

**Compatibilidad con distintas bases de datos:** Uso de ORM para facilitar migraciones.

- Mantenibilidad y Escalabilidad:

**Código modular:** Uso de arquitectura en capas (por ejemplo, MVC con Spring Boot) para facilitar futuras ampliaciones.

La jerarquía de clases **Usuario, Socio y Empleado** permite incorporar fácilmente nuevos tipos de usuarios si el gimnasio amplía sus servicios o personal.

# Patrones de Diseño

---

- + Capas
- + MVC
- + Inyección de dependencias
- + DTO
- + DAO

# Patrones GRASP (1/2)

**Experto en información** → cada clase maneja sus propios datos (**Factura**, **CuotaMensual**).

**Controlador** → **PagoControlador**, **FacturaControlador** median entre UI y lógica.

**Creador** → **PersonaService** crea **Socio** al conocer la info necesaria.

**Baja acoplamiento / Alta cohesión** → capas bien separadas



## Patrones GRASP (2/2)

- Polimorfismo** → herencia (**Persona** abstracta → **Socio**, **Empleado**).  
enums (**EstadoCuota** cada tipo puede usarse en situaciones diferentes)
- Indireccion** → servicios median entre controladores y repositorios.  
(el controlador no se comunica directamente en la base de datos)

# Patrones GoF

**Builder** → creación de objetos con Lombok (`PagoOnline.builder()`).

# Software en Funcionamiento

---

# Nueva Funcionalidad

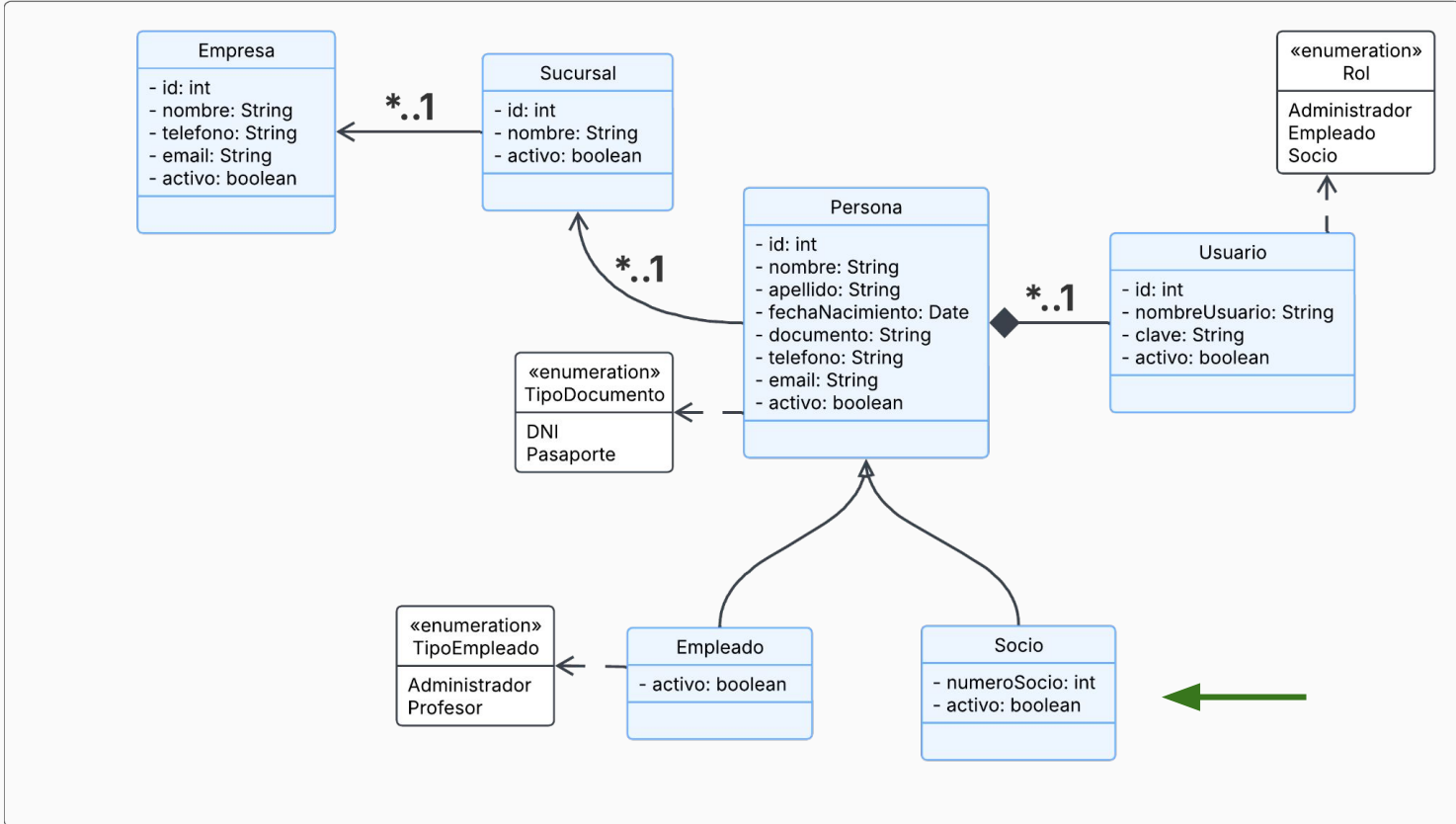
---

Socios acumulan puntos por acciones (asistencia, pago en término, referidos) y los canjean por beneficios.

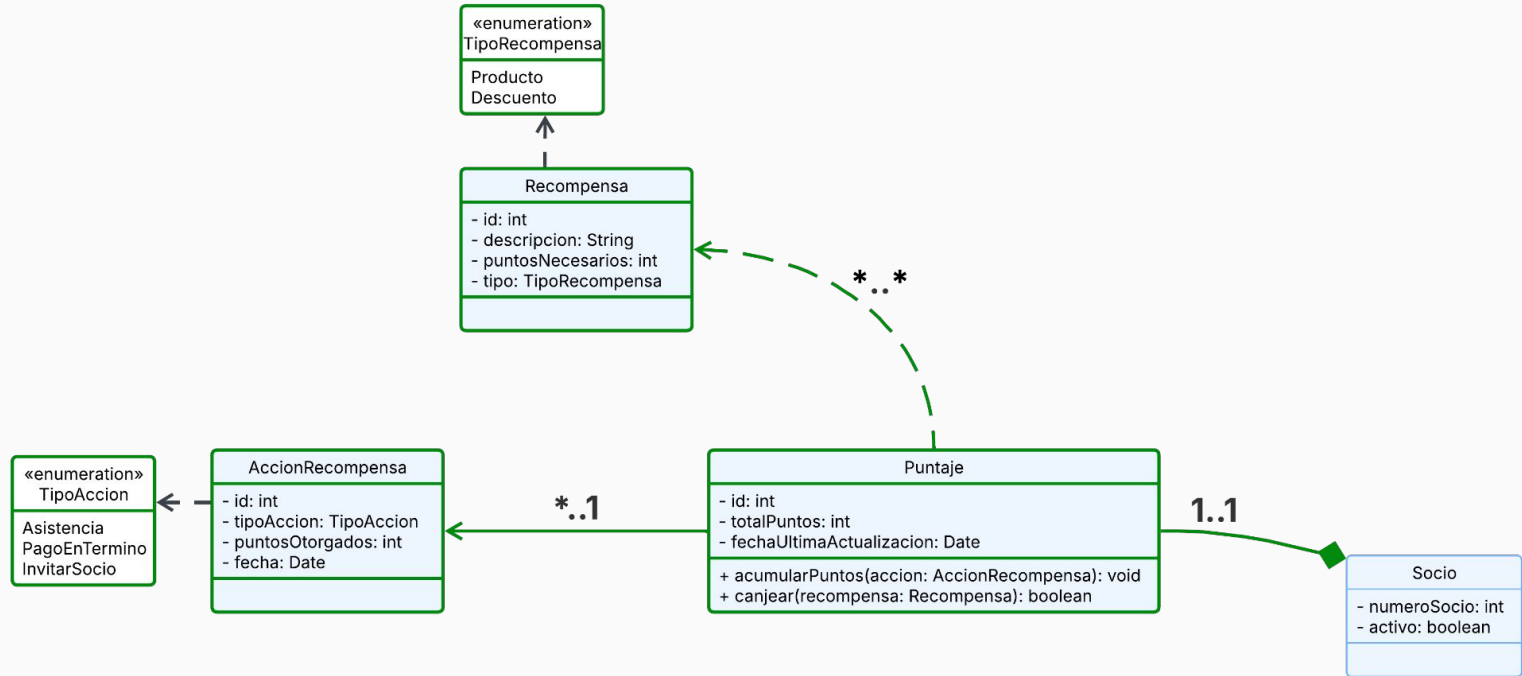
---

Ejemplo: Mes del Amigo

## Funcionalidad



## Funcionalidad



# Conclusión

---



Trabajo sinérgico entre  
documentación e  
implementación.

División de tareas +  
Comunicación constante

*¿Preguntas?*

*¡Muchas  
Gracias!*



*¡Muchas Gracias!*

```
System.out.print  
("Muchas  
Gracias");
```

# Fin

---