

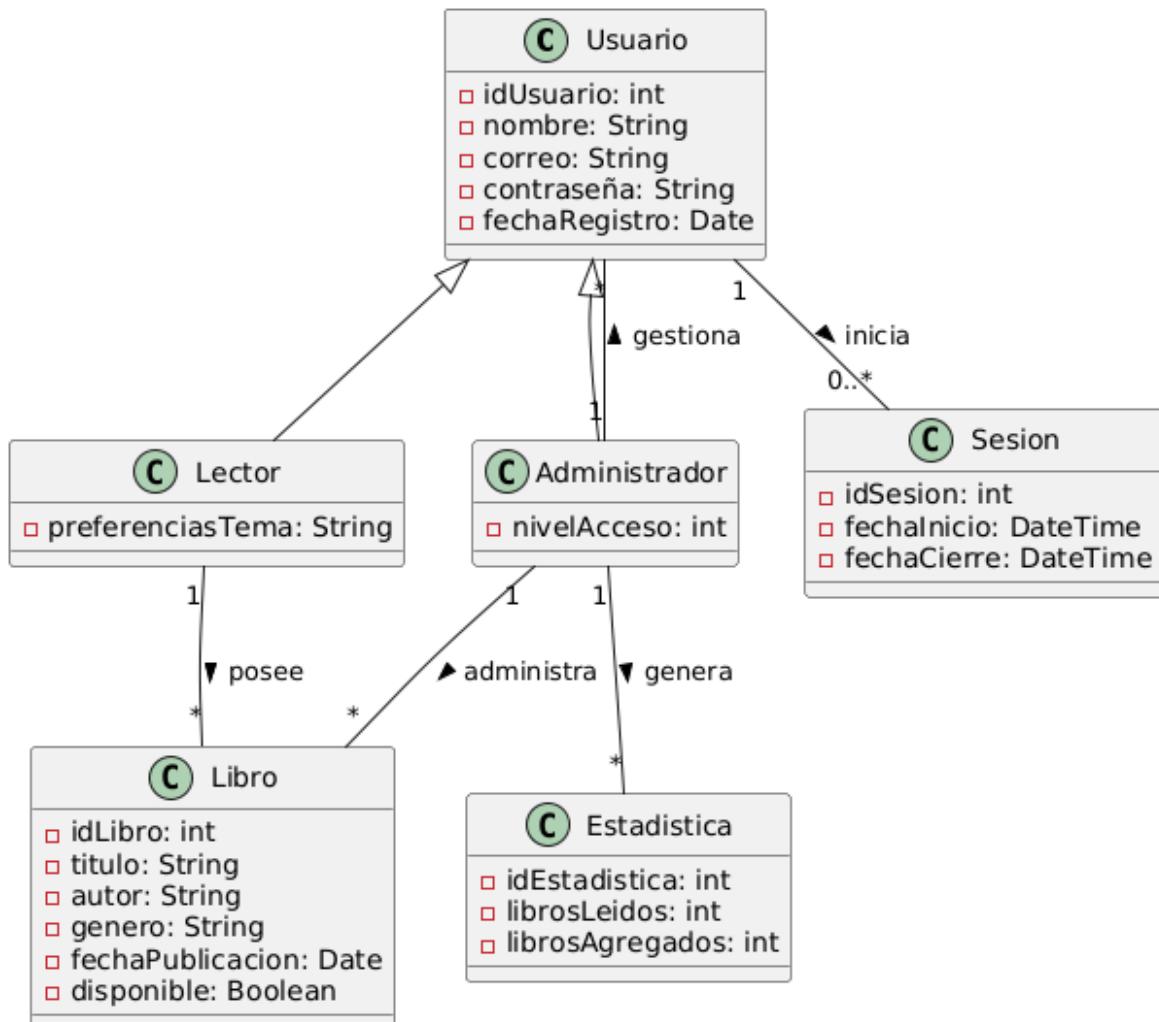
## Ejercicio 1

### Diagrama de clases

El diagrama de clases conceptuales busca representar las entidades principales del sistema **StoryVerse** de forma abstracta, enfocándose en los conceptos del dominio sin entrar aún en detalles de implementación (como frameworks, base de datos o servicios web).

Este diagrama constituye el primer acercamiento estructurado al problema y define:

- Las **clases conceptuales** (ideas centrales del sistema).
- Los **atributos relevantes** que caracterizan a cada clase.
- Las **relaciones** entre clases (asociaciones, agregaciones, composiciones, generalizaciones).
- Las **cardinalidades** (cuántas instancias de una clase se relacionan con otras).



### Notas sobre el diseño:

- Usuario es una **superclase** de Lector y Administrador.
- Lector tiene una **asociación múltiple** a Libro, indicando su **biblioteca personal**.

- Administrador puede gestionar tanto Usuarios como Libros y generar Estadísticas.
- Sesión está asociada directamente con Usuario, permitiendo múltiples sesiones históricas.
- Se representan adecuadamente las **cardinalidades**.

### Modelo de Dominio

El **modelo de dominio** representa el conjunto de conceptos fundamentales que estructuran el sistema, así como las **reglas de negocio** y **restricciones** que aseguran la integridad del mismo.

#### Reglas de negocio por entidad:

- **Usuario:**
  - Cada usuario debe registrarse con un correo electrónico único.
  - No puede existir un usuario sin contraseña cifrada ni nombre completo.
  - Un usuario no puede ser simultáneamente un Lector y un Administrador.
- **Lector:**
  - Solo puede gestionar su propia biblioteca de libros.
  - Puede configurar sus preferencias de tema visual.
- **Administrador:**
  - Puede dar de alta, modificar o eliminar libros de StoryVerse.
  - Puede bloquear usuarios en caso de incumplimiento de normas.
  - Tiene acceso a reportes estadísticos agregados.
- **Libro:**
  - Cada libro debe contar obligatoriamente con título, autor y fecha de publicación válidos.
  - La disponibilidad debe actualizarse cuando el administrador lo retire del catálogo.
- **Sesión:**
  - Cada sesión debe registrar hora de inicio.
  - No se permite registrar el cierre de sesión antes de su inicio.
- **Estadística:**
  - Se actualiza automáticamente cada vez que un administrador realiza acciones de gestión de libros o usuarios.

#### Invariantes del sistema:

- Todo correo registrado debe ser único en la plataforma.
- Ninguna contraseña debe almacenarse en texto plano.
- No puede haber sesiones abiertas simultáneamente para un mismo usuario si ya existe una activa.

#### Restricciones adicionales:

- Las fechas de publicación de libros no pueden ser posteriores a la fecha actual.
- Los nombres de los libros deben ser únicos dentro del sistema para evitar ambigüedades.
- El nivel de acceso del administrador debe ser mayor a cero.

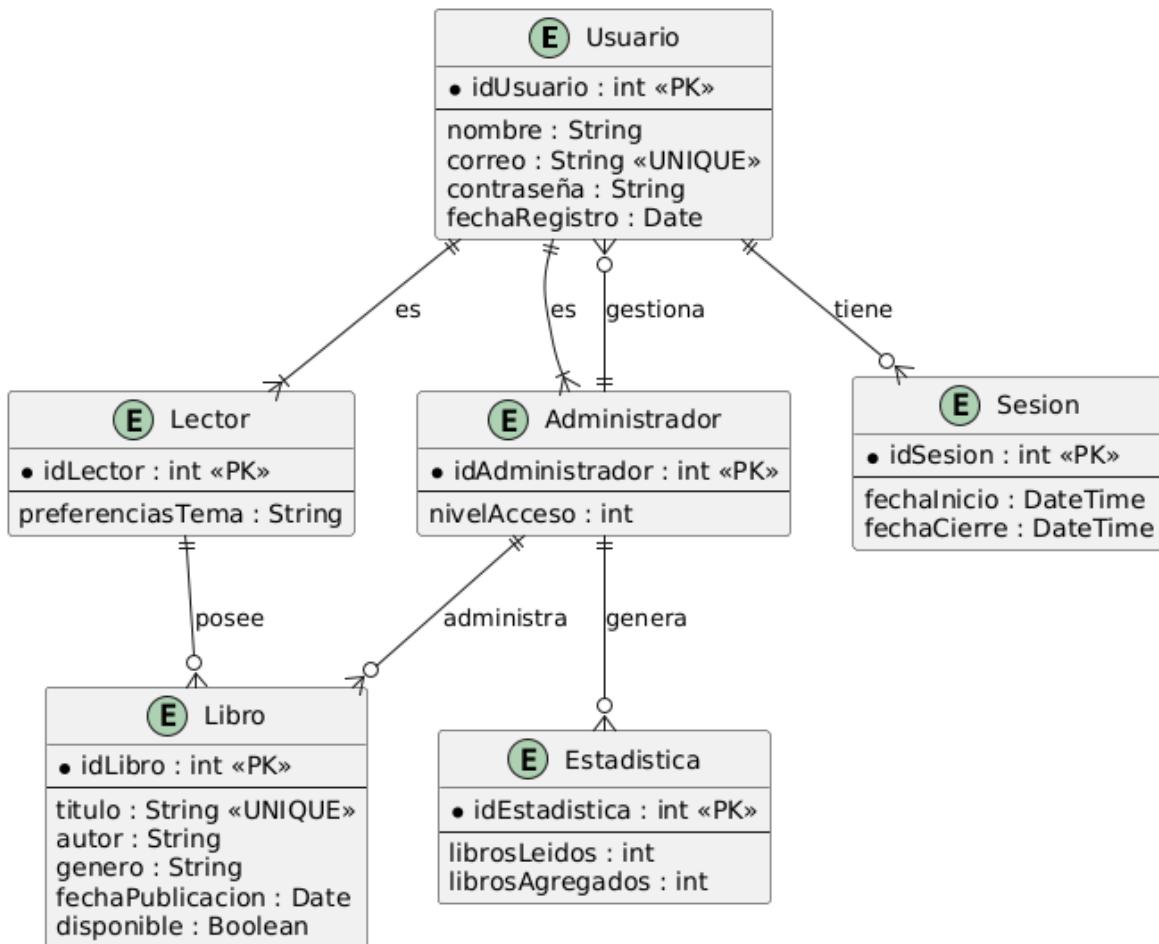
## Diccionario de datos

Entidad	Descripción	Atributos	Tipo de dato	Restricciones	Relaciones
<b>Usuario</b>	Representa a cualquier usuario de StoryVerse.	idUsuario	int	PK, obligatorio	Sesión (1:N), Subclases Lector/Administrador
		nombre	String	Obligatorio	
		correo	String	Único, obligatorio	
		contraseña	String	Obligatorio, cifrado	
		fechaRegistro	Date	Obligatorio	
<b>Lector</b>	Usuario orientado a la lectura de libros.	preferenciasTema	String	Opcional	Relación con Libro (N:N)
<b>Administrador</b>	Usuario con permisos especiales de gestión.	nivelAcceso	int	Obligatorio (>0)	Relación con Usuario, Estadística con Libro,
<b>Libro</b>	Objeto de lectura digital en el sistema.	idLibro	int	PK, obligatorio	Relación con Lector, Administrador
		titulo	String	Único, obligatorio	
		autor	String	Obligatorio	
		genero	String	Opcional	
		fechaPublicacion	Date	Obligatorio, no futura	
		disponible	Boolean	Obligatorio	
<b>Sesion</b>	Registro de acceso del usuario.	idSesion	int	PK, obligatorio	Relación con Usuario
		fechaInicio	DateTime	Obligatorio	
		fechaCierre	DateTime	Opcional	
<b>Estadistica</b>	Reporte de actividades del administrador.	idEstadistica	int	PK, obligatorio	Relación con Administrador
		librosLeidos	int	Obligatorio	
		librosAgregados	int	Obligatorio	

## Modelo Entidad Relación

El diagrama E-R de **StoryVerse** refleja cómo se estructurará la base de datos a partir de las entidades conceptuales:

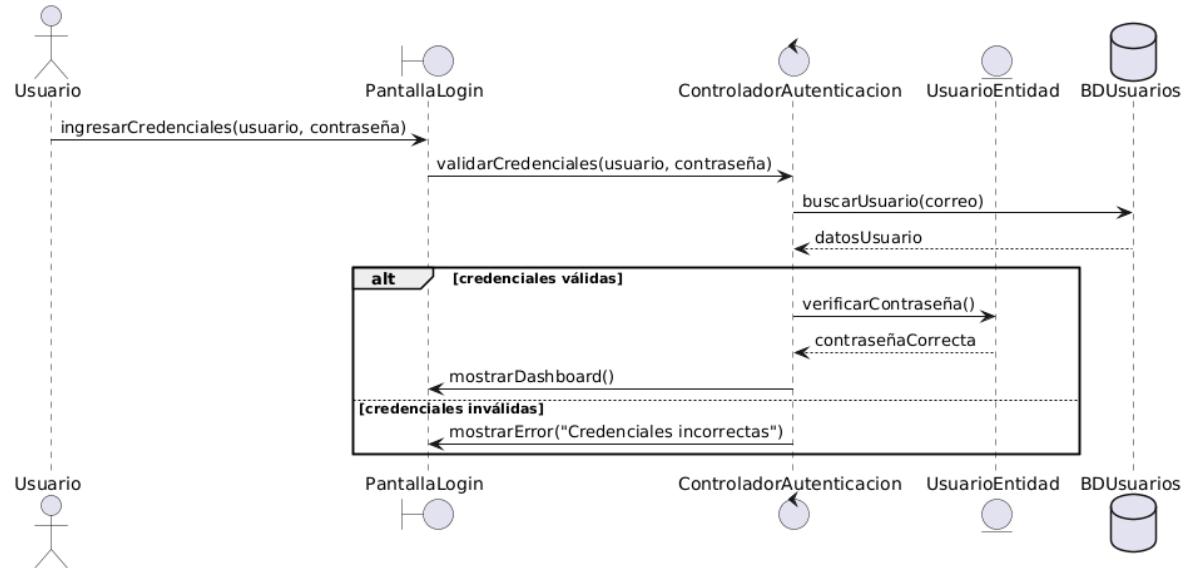
- Cada entidad conceptual se transforma en una tabla.
- Las relaciones de herencia se representan por medio de claves foráneas (foreign keys).
- Se definen explícitamente las claves primarias y relaciones N:N o 1:N según el caso.



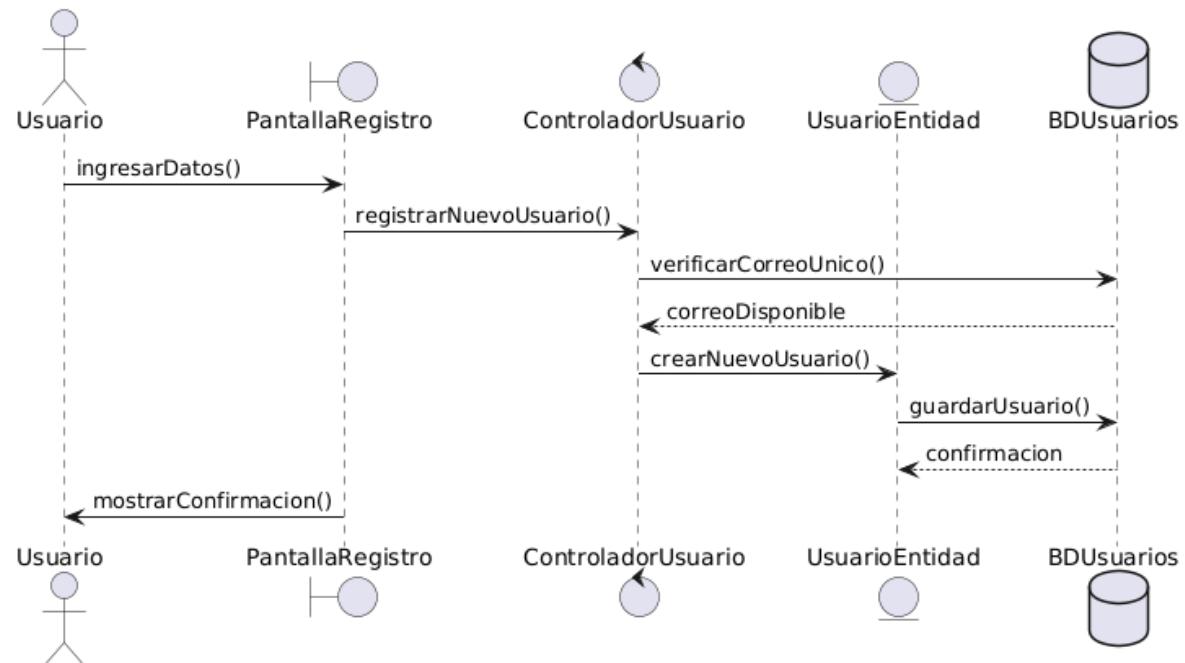
## Ejercicio 2

### Diagramas de secuencia

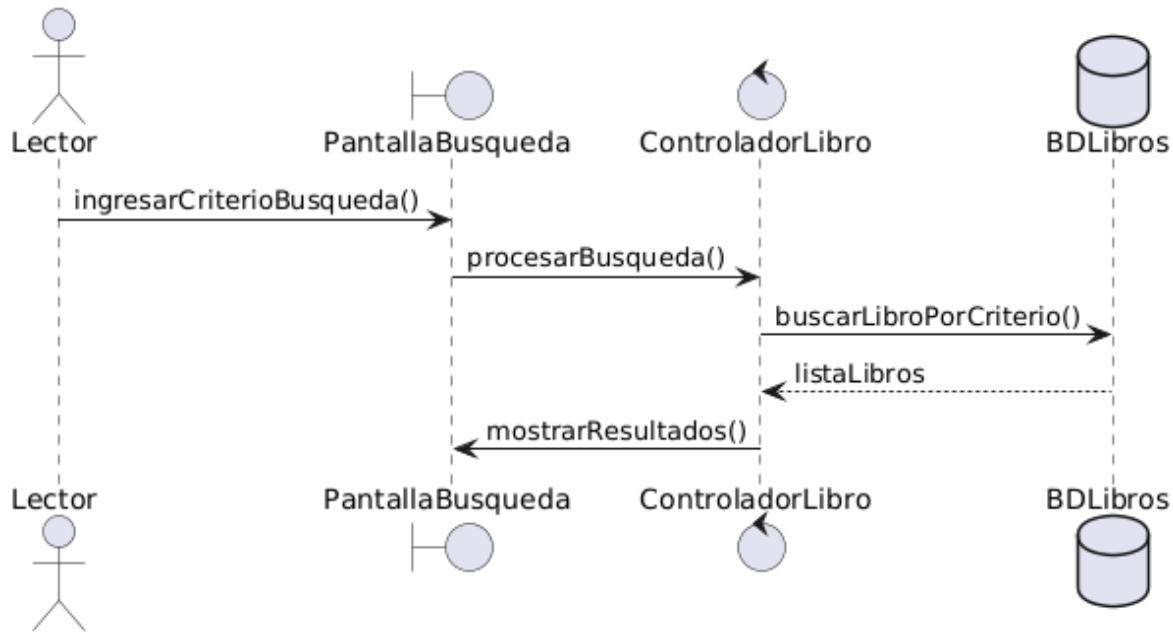
#### Inicio de sesión



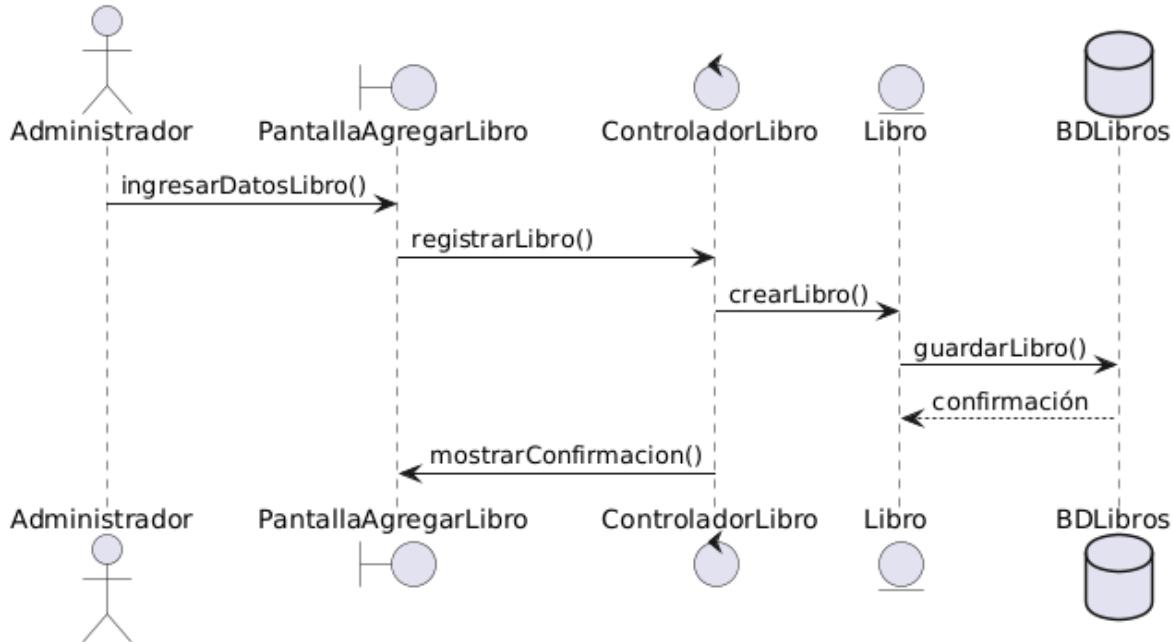
#### Registrarse



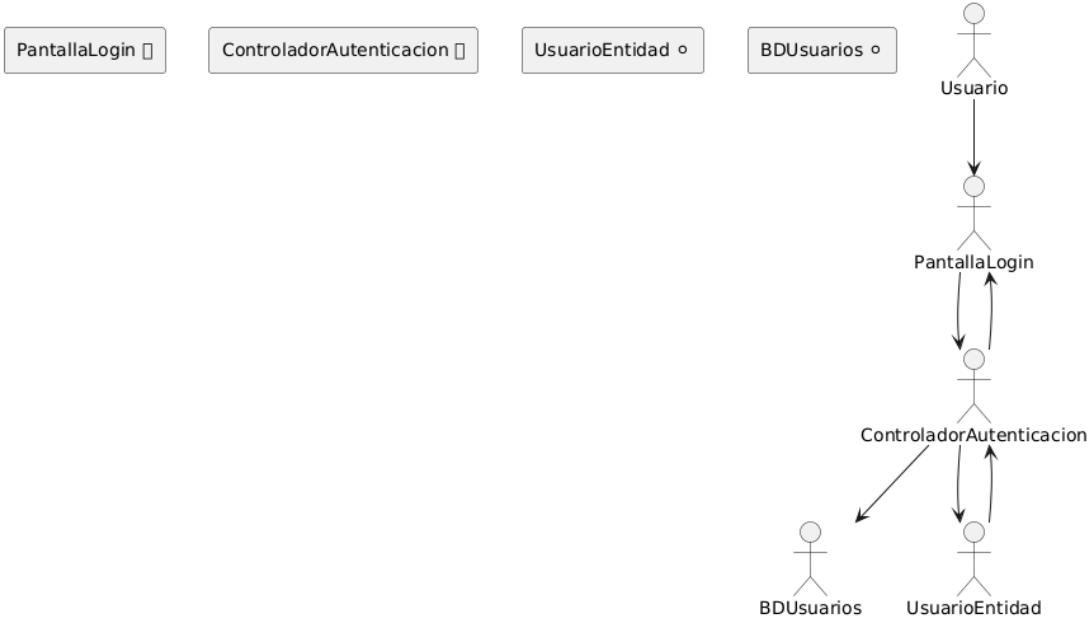
#### Buscar libro



### Agregar libro

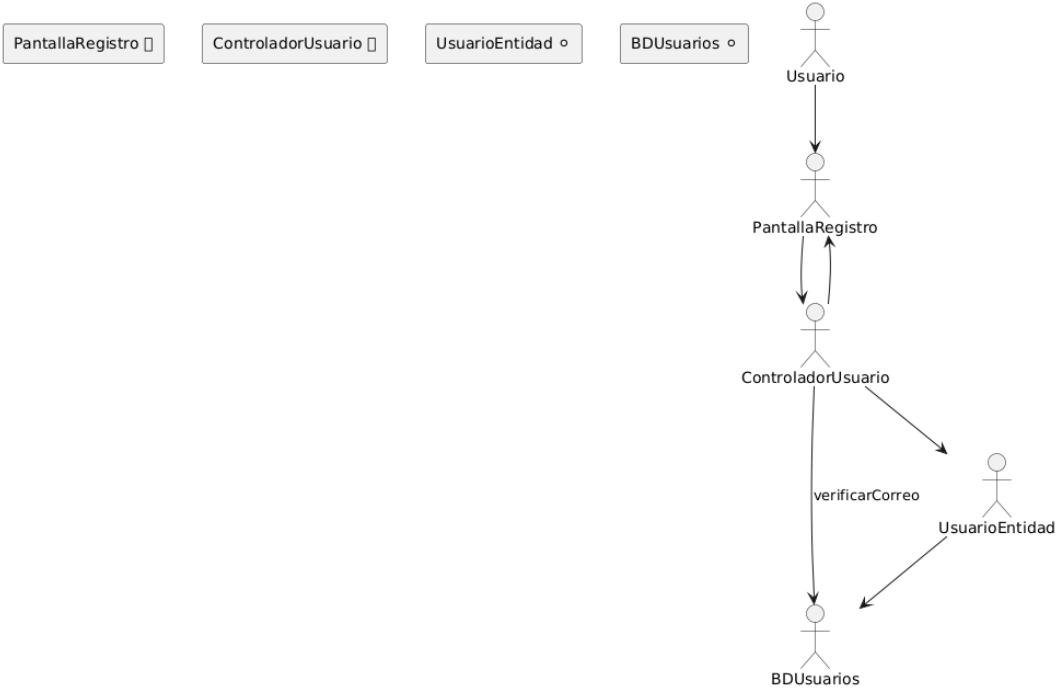


## Diagrama de robustez



### Convenciones:

- ◆ Objeto de interfaz (boundary)
- ◆ Objeto de control
- ◆ Objeto de entidad

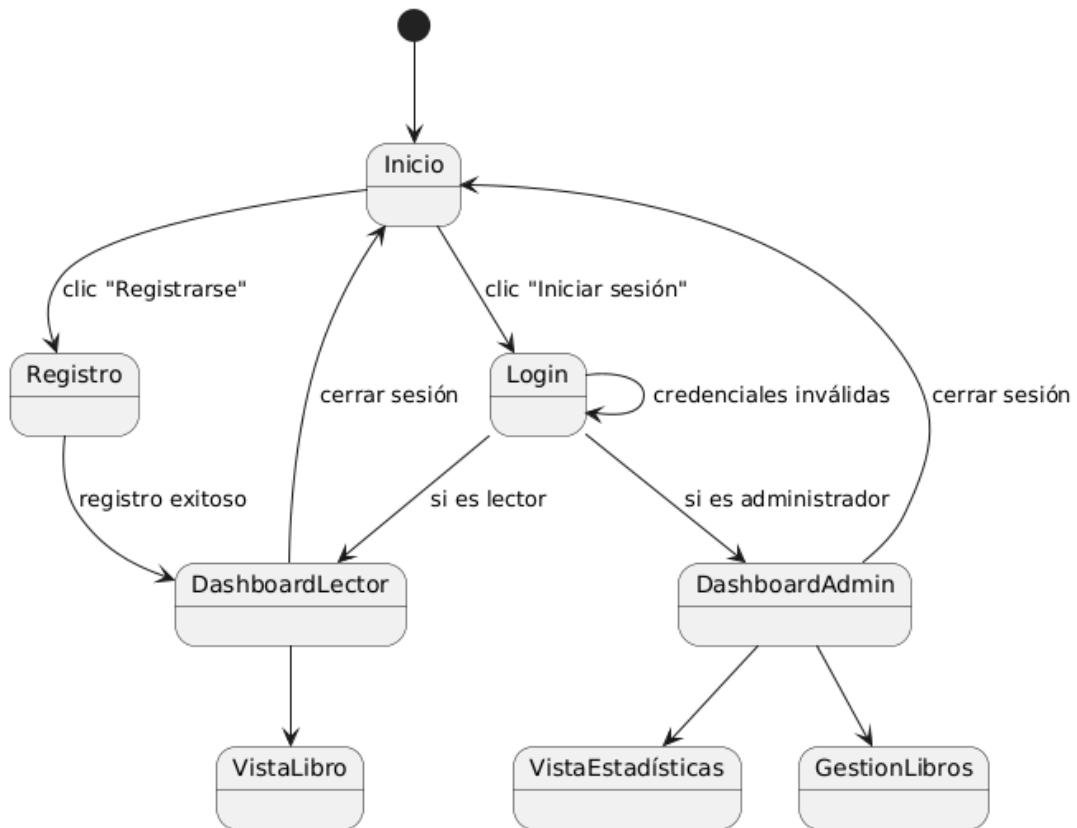


## Modelo de Interfaz y Navegación

### 3.1. Diseño esquemático de pantallas principales

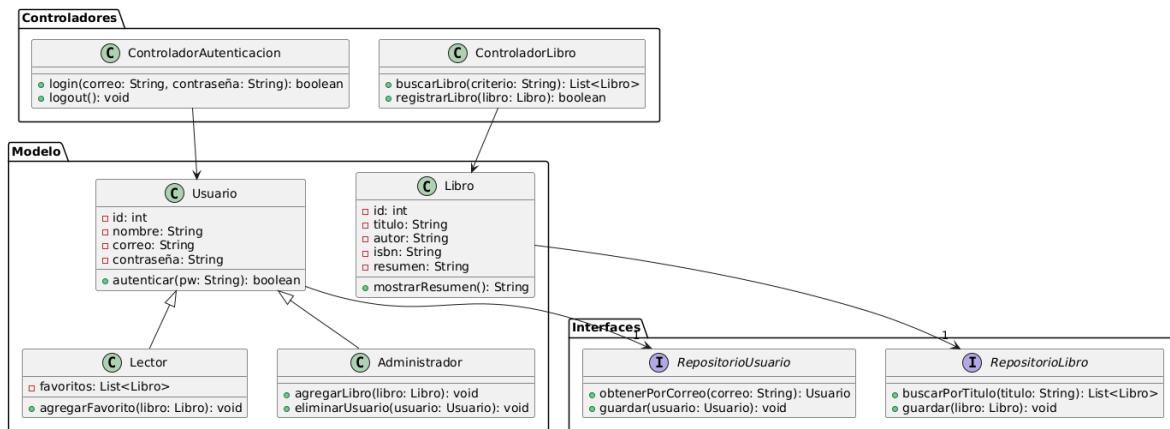
Pantalla	Componentes
Inicio	Botones: "Iniciar sesión", "Registrarse", acceso público al catálogo
Login	Formulario de usuario/contraseña, botón "Iniciar sesión", mensajes de error
Registro	Formulario de nombre, correo, contraseña, botón "Registrarse"
Dashboard Lector	Menú lateral, búsqueda de libros, acceso a libros guardados
Dashboard Admin	Menú de gestión (usuarios/libros), acceso a estadísticas, botón cerrar sesión
Vista Libro	Título, autor, resumen, botón "Agregar a favoritos"
Vista Estadísticas	Gráficas de uso, contadores de libros leídos, usuarios activos

## Modelo de navegación



## Ejercicio 3

### Diagrama de clases de diseño



### Patrón de Diseño

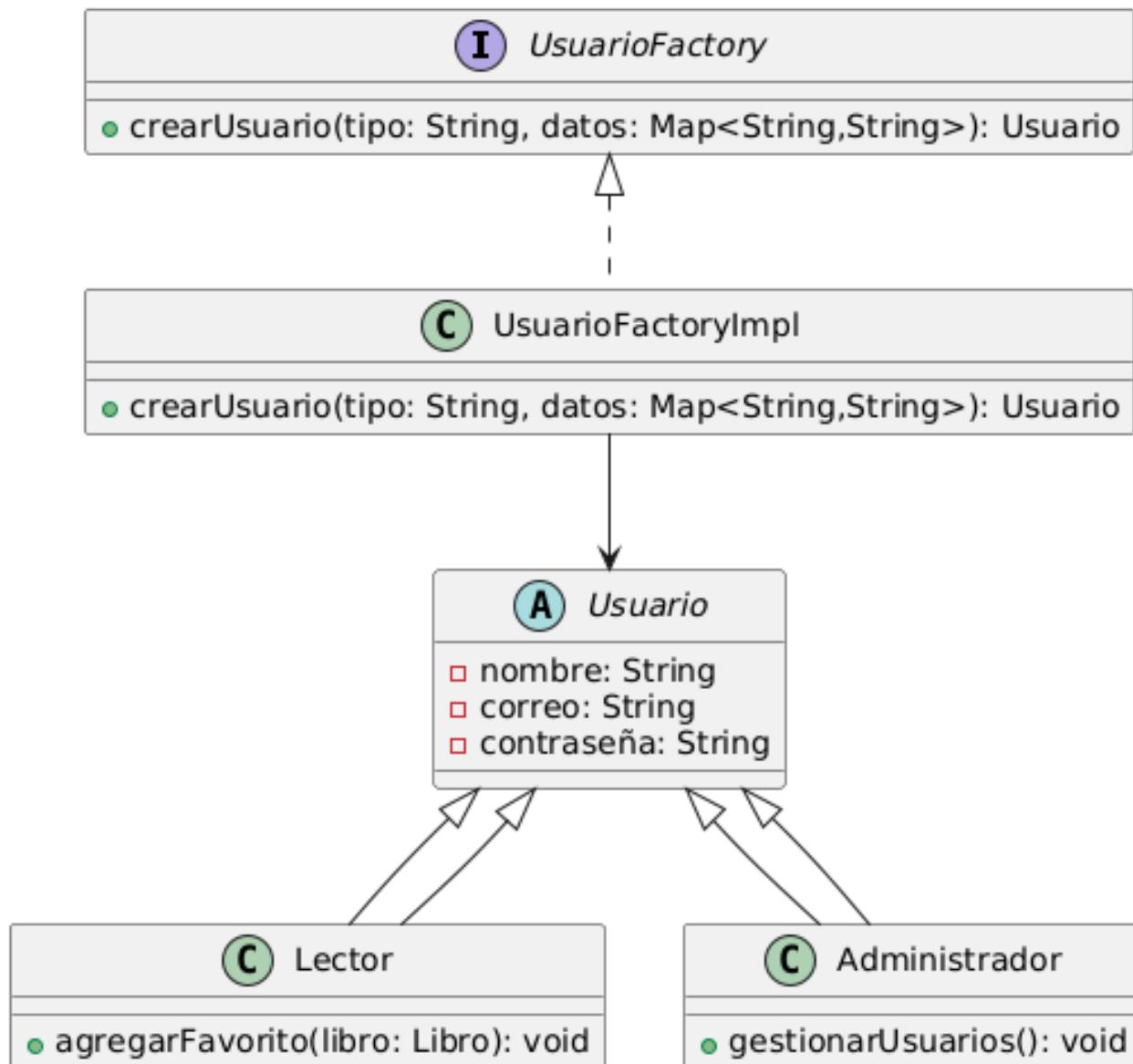
#### Patrón Seleccionado: Factory Method

#### Justificación:

Se utiliza para encapsular la creación de objetos **Usuario** según su tipo (**Lector**, **Administrador**), especialmente durante el inicio de sesión o el registro. Mejora la extensibilidad y mantiene la apertura al cambio (OCP - SOLID).

#### Ventajas en StoryVerse:

- Abstracción del tipo concreto de usuario.
- Facilita pruebas unitarias y mantenimiento.
- Permite integrar nuevos tipos de usuarios sin modificar código base.



## Modelo de Implementación

### Arquitectura Técnica del Sistema:

**Tipo de arquitectura:** MVC distribuido con componentes desacoplados

**Estilo arquitectónico:** Arquitectura por capas

### Tecnologías utilizadas:

- Backend: Spring Boot
- Frontend: Thymeleaf + Bootstrap
- Base de datos: MySQL
- Seguridad: Spring Security
- Comunicación: RESTful API

## Estructura de Componentes (por paquetes):

Paquete	Responsabilidad
modelo	Entidades de dominio (Usuario, Libro, etc.)
controladores	Coordinación de la lógica y servicios del sistema
repositorios	Interfaces para acceso a datos (JPA, MySQL)
servicios	Lógica de negocio reusable y transaccional
seguridad	Control de acceso y manejo de sesiones
interfaces	Plantillas HTML, controladores REST, UI front-end

## Consideraciones de seguridad, rendimiento y escalabilidad:

- Autenticación segura con BCrypt.
- Cache de consultas frecuentes.
- Uso de paginación y filtros para rendimiento.
- Diseño orientado a microservicios futuros.

## Diagrama de despliegue

