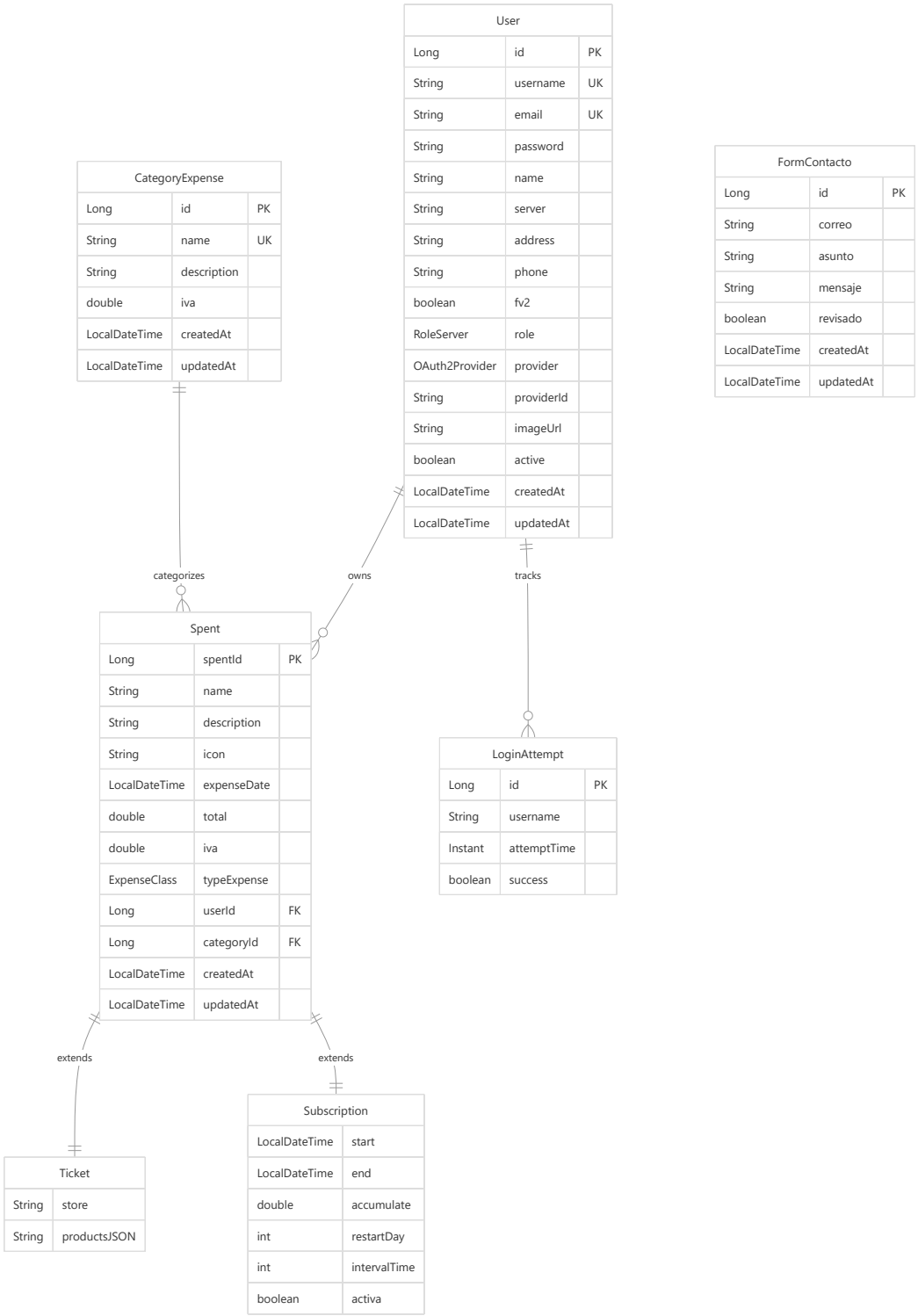
Gestión de datos

Este documento abarca la capa de persistencia de datos del sistema ProyectoGestorGastos, incluyendo las definiciones de entidades JPA, el diseño del esquema de la base de datos y los patrones de acceso a datos. La capa de datos implementa un modelo de dominio integral para el seguimiento de gastos, compatible con tickets procesados por OCR, suscripciones recurrentes, gestión de usuarios y monitorización del sistema.

Para obtener información sobre la capa de servicio que opera en estas entidades, consulte Servicios de backend . Para operaciones de almacenamiento de archivos, consulte Sistema de almacenamiento de archivos .

Modelo de entidad y relaciones

El sistema utiliza entidades JPA con una estructura de relaciones bien definida, centrada en el seguimiento de gastos. Las entidades principales forman una jerarquía que `Spent` sirve de base para todos los tipos de gastos.

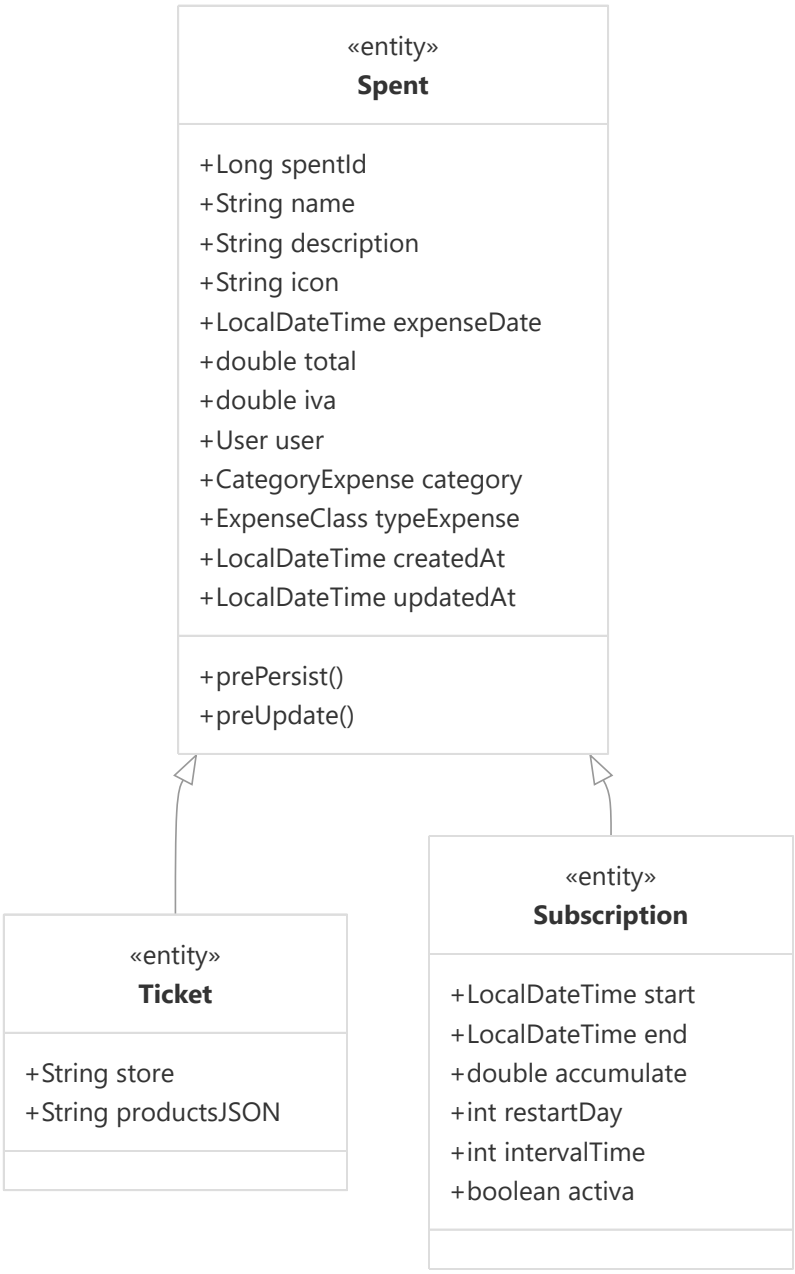


Las relaciones entre entidades implementan un dominio completo de gestión de gastos con:

- **Usuarios** que poseen gastos y tienen seguimiento de inicio de sesión
- **Gastos** (Gastados) categorizados y propiedad de los usuarios
- **Tickets** para recibos procesados por OCR con detalles del producto
- **Suscripciones** para el seguimiento de gastos recurrentes
- **Categorías** para la clasificación de gastos
- **Formularios de contacto** para la comunicación con los usuarios

Estrategia de herencia y jerarquía de entidades

El sistema implementa la herencia de tabla por subclase mediante la `JOINED` estrategia para entidades de gastos. La `Spent` clase actúa como entidad base con `Ticket` y `Subscription` como implementaciones especializadas.



La asignación de herencia se configura con:

- **Tabla base** : `gastos` para `Spent` la entidad

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Spent.java` `12-13`

- **Tabla unida** : `tickets` para `Ticket` entidad

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Ticket.java` `14`

- **Tabla unida** : `subscripciones` para `Subscription` entidad

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Subscription.java` `13`

Este enfoque proporciona:

- Esquema de base de datos normalizado sin columnas duplicadas
- Consultas polimórficas con seguridad de tipos
- Almacenamiento eficiente para atributos de gastos compartidos

Validación y restricciones de datos

Todas las entidades implementan una validación integral utilizando anotaciones de Bean Validation para garantizar la integridad de los datos a nivel de la aplicación.

Entidad	Validaciones clave	Objetivo
`User`	`@UniqueConstraint` en nombre de usuario/correo electrónico	Evitar cuentas duplicadas
`Spent`	`@NotBlank` nombre, `@DecimalMin` total	Asegúrese de que los datos de gastos sean válidos
`Ticket`	`@Size` nombre de la tienda, `@NotBlank` validación	Validar datos procesados mediante OCR
`Subscription`	`@Min/@Max` para rangos de fechas	Garantizar períodos de suscripción válidos
`CategoryExpense`	`@DecimalMin/@DecimalMax` para el IVA	Validar porcentajes de impuestos
`LoginAttempt`	`@NotNull` marca de tiempo, `@NotBlank` nombre de usuario	Requisitos de auditoría de seguridad

Reglas de validación críticas

Restricciones de entidad de usuario:

- El nombre de usuario y el correo electrónico deben ser únicos

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/User.java` `24-27`

- El campo de rol es obligatorio para la autorización

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/User.java` `63-66`

Validación de gastos:

- El importe total debe ser mayor a 0,01

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Spent.java` `62-64`

- Porcentaje de IVA restringido entre 0 y 100

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Spent.java` `70-73`

Reglas de negocio de suscripción:

- El día de reinicio debe ser entre la 1 y la 31

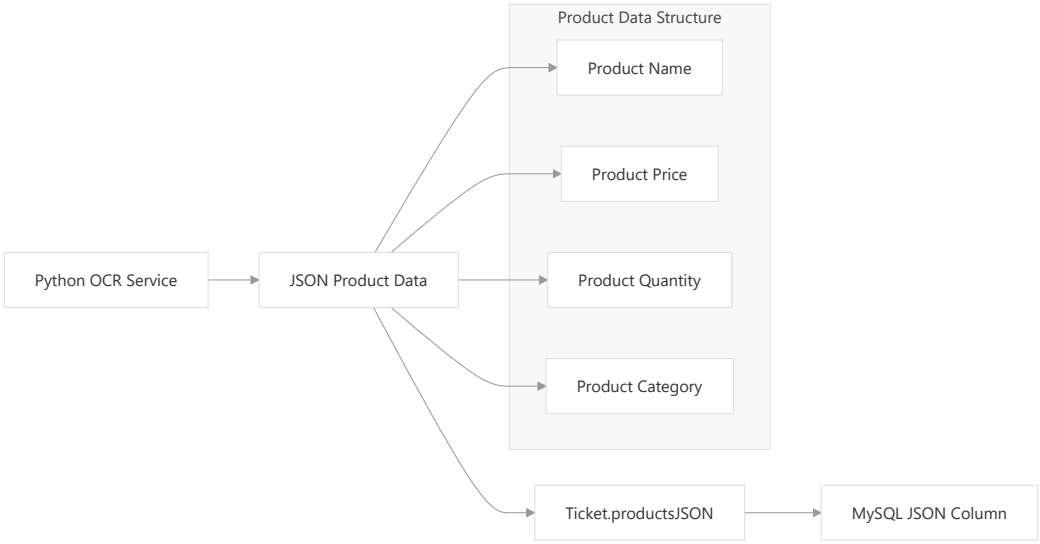
  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Subscription.java` `42-45`

- El tiempo de intervalo debe ser al menos 1

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Subscription.java` 51-53

Integración de OCR y almacenamiento de datos JSON

La `Ticket` entidad incluye almacenamiento especializado para datos de productos procesados con OCR mediante el tipo de columna JSON nativo de MySQL. Este diseño permite la extracción flexible de información de productos de las imágenes de recibos.



Configuración de la columna JSON:

- Definición de columna: `columnDefinition = "JSON"`

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Ticket.java` 35

- Nombre del campo: `productsJSON` para almacenar datos estructurados del producto

  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Ticket.java` 36

- No se pueden anular para garantizar la integridad de los datos

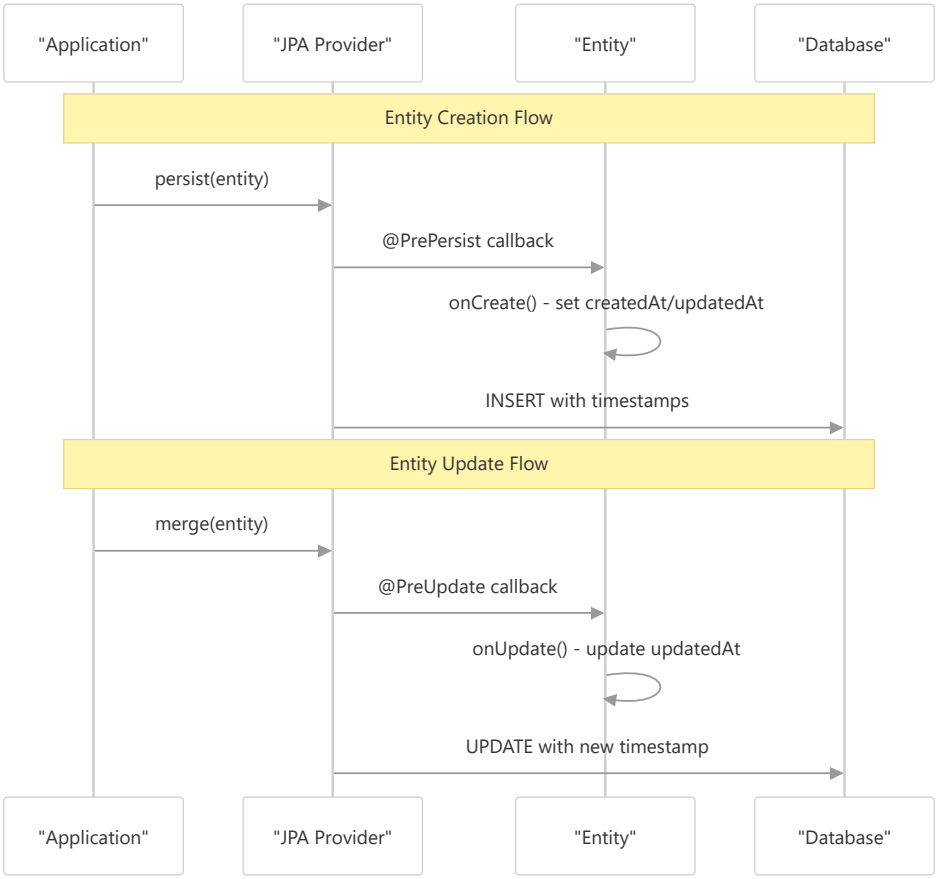
  `ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Ticket.java` 35

Este enfoque proporciona:

- Capacidades de consulta JSON nativas en MySQL
- Esquema flexible para variar estructuras de productos
- Almacenamiento eficiente de datos extraídos mediante OCR
- Integración directa con la salida del servicio OCR de Python

Gestión automática de marcas de tiempo

Todas las entidades implementan el seguimiento automático de marcas de tiempo utilizando devoluciones de llamadas del ciclo de vida de JPA para mantener registros de auditoría y versiones de datos.



Patrón de implementación de marca de tiempo:

Todas las entidades siguen el mismo patrón de gestión de marcas de tiempo:

- **`createdAt`** :Establecer una vez durante **`@PrePersist`**

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/User.java`	`103-107`

- **`updatedAt`** :Actualizado cada **`@PreUpdate`**

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/User.java`	`113-116`

Entidades con gestión de marca de tiempo:

- **`User`** :

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/User.java`	`103-116`

- **`Spent`** :

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/Spent.java`	`113-122`

- **`CategoryExpense`** :

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/CategoryExpense.java`	`72-81`

- **`FormContacto`** :

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/models/FormContacto.java`	`71-80`

Estructura del esquema de base de datos

El esquema de base de datos física implementa el modelo de entidad lógica con indexación adecuada y restricciones de clave externa.

Nombre de la tabla	Clave principal	Claves externas	Restricciones únicas	Objetivo