## ANHEC – Sistema de liquidación de gastos comunes

## 1.- Objetivos

Objetivo General: Diseñar un sistema de administrador de gastos comunes, que permita *gestionar la cobranza, actualizar saldos de deuda, gestión de servicios propios y tercerizados, reportes y relaciones con proveedores*.

1. Su función principal es la liquidación de los gastos propiamente dichos, la carga de cuotas y el procesamiento de los pagos.

Alcances: El sistema de administración de gastos comunes permitirá gestionar de manera eficaz :

* La información de los inmuebles así como las de sus propietarios o inquilinos
* Facilitar el acceso a la información con los niveles de seguridad, acceso y control apropiados
* Control de gastos
* Impresión de documentos
* Envió por eMail de información y documentos
* Gestión de personal
* Gestión de tares
* Gestión de materiales
* Gestión financiera

Objetivos Específicos:

* Gestión administrativa de propiedades inmuebles (realestate)
* Administración de propiedades horizontales (condominio)
* Gestión de gastos (servicios propios, servicios tercerizados, proveedores)
* Gestión de pagos
* Reportes

## 2.- Lista de requerimientos con una breve descripción

* Gestión de usuarios : El sistema debe manejar el control de acceso, el registro de los usuarios, recuperación de contraseña. También debe ser capaz de gestionar el permiso de ejecución y acceso según **rol** del usuario
* Gestión de edificio : Definición de la unidad, sus propiedades (según % mts2), **servicios** fundamentales (electricidad, agua, gas, calefacción, … ). Relación con sus propietarios e inquilinos (**vecino**) . Data del inmueble (planos, ubicación, )
* Gestión de Gastos : Gastos generales e individuales (relación con **proveedores**)
* Gestión de Pagos : Pago de servicios, obras, incidencias (tipo de **incidencia**)
* Gestión de Cobranza : Recaudación, moras, multas, extras
* Gestión de Deudas : Deudas no canceladas con su soporte
* Acuerdos de pago
* Auditoria : El sistema debe permitir el realizar una traza de cualquier acceso, consulta, modificación y gestión de los elementos internos del sistema, registrando la hora, el usuario, el modulo y la acción realizada (r**eporte de auditoria con filtro**)
* Tareas (checkList) : Listas que deben ser cumplidas al iniciar o finalizar (cierre) de un periodo administrativo de una propiedad
* Contabilidad : Modulo para el encuadre de ingresos y egreso, depósitos, transferencias (**consolidación bancaria**)
* Consultas

Algunas características adicionales

* Los edificios se gestionan por periodos
* Los usuarios, proveedores, personal, clientes y propietarios/inquilinos serán importados, creados y gestionados en la base de datos de SuiteCRM
* Las notificaciones a los clientes, proveedores, vendedores, propietarios/inquilinos sean enviadas por email y serán publicadas en el blog (wordpress)
* Los propietarios/inquilinos podrán ingresar al sistema (con una cuenta personal) en la que podrán ver la documentación pertinente a cada uno.
* El sistema tendrá un modulo de gestión general para administrar los usuarios, los inmuebles y en general los elementos del mismo sistema
* El modulo de compras (ANHEC Compras) debe soportar la gestión de comunicación con los proveedores de forma de enviar notificaciones, información, facturas.

## 3.- Análisis Funcional

Area Administrativa

* Administración y control del sistema.
* Gestión de proveedores
* Gestión de cobros, facturas y pagos
* Gestión administrativa del condominio.
* Envío de avisos de cobro por email a los residentes
* Envío de emails con noticias, convocatorias, alertas, etc.
* Publicar informes administrativos.
* Publicar documentos de interés y fotos.
* Publicar mensaje global y foto principal en la página.
* Enviar mensajes privados a los propietarios o residentes

Area Residencial

* Consultar estados de cuenta, avisos de cobro e historial de cualquier mes.
* Notificar pagos.
* Consultar recibos de pago.
* Ver datos del Condominio, Junta directiva y Administradora.
* Ver reportes administrativos.
* Enviar mensajes a la empresa administradora.
* Publicar mensajes para ser vistos por todos los residentes.
* Ver documentos de interés y fotos.

### Paquetes

**Condominios**

- Múltiples condominios.  
- Datos básicos.  
- Junta de condominio.  
- Trabajadores, documentos, fotos y logo.  
- Configuración de la página del condominio.

**Inmuebles**

- Datos del inmueble.  
- Datos del propietario.  
- Datos del inquilino.

**Gastos**

- Cuentas de gastos.  
- Cuotas comunes.  
- Cuotas no comunes.  
- Cuotas extra.  
- Generación de cuotas.  
- Copia, bloqueo, impresión.

**Cobranzas**

- Selección de cuotas por cobrar.  
- Cancelaciones, abonos, anticipos.  
- Múltiples formas de pago.  
- Integrado con caja y bancos.

**Caja y bancos**

- Múltiples cuentas bancarias.  
- Cheques, depósitos, n/débito, n/crédito.  
- Conciliación bancaria.  
- Manejo de caja chica y caja principal.  
- Ingresos, egresos y depósitos de caja.

**Aviso de cobro y estado de cuenta interactivo**

- Enviado al propietario por email.  
- Detalle e impresión del aviso de cobro.  
- Notificación de pago en línea.  
- Lista e impresión de recibos.  
- Historial del inmueble.

**Página del condominio automática**

- Para ser vista por el propietario.  
- Generación automática de la página.  
- Datos del condominio y empresa administradora.  
- Envío de mensaje a la empresa administradora.  
- Visualización de informes, fotos, documentos.

**Control**

- Parametrizar sistema

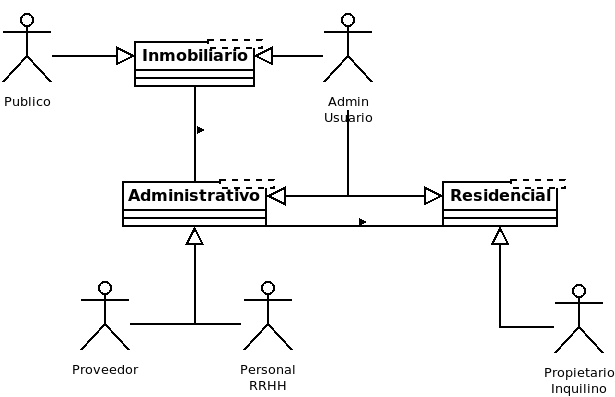
- Historial de movimientos.  
- Intereses por demora. Notas de débito colectivas  
- Ingresos por cobro de cuotas y extraordinarios.  
- Manejo de múltiples fondos.

**Informes**

- Cuadro general de cuotas por cobrar.  
- Avisos de cobro.  
- Cuotas vencidas.  
- Ingresos y Gastos.  
- Historial de movimientos y del inmueble.  
- Gráfico de gastos, Ingresos, morosidad.  
- Movimiento bancario. Conciliaciones.  
- Movimiento de caja.  
- Notificaciones de pago.  
- Recibos.  
- Directorio de residentes.  
- Presentación en formatos pdf, Excel, Html.

### Análisis Funcional (Descriptivo)

El sistema consta de 3 unidades principales que están relacionadas entre si



**Administrativo** : Este modulo se encarga de la gestionar las relaciones que establecen entre las propiedades, las unidades, los propietarios (o inquilinos), los proveedores y las formas contables (ingresos, egreso, deudas, facturas, ….). Es el core del negocio del sistema y es donde la mayor parte de la lógica de negocio sera descrita.

**Residencial** : Este modulo sera responsable de mostrar la información pertinente y según el rol de los propietarios en inquilinos. También se encargara de capturar los requerimientos e información de los propietarios e inquilinos a fin de iniciar los procesos administrativos, p.e. reportar un pago de cuota.

**Inmobiliario** : Este modulo sera responsable de cargar la información de las propiedades a ser ofertadas al publico general de forma, que actuara como una vitrina de exposición de dichas propiedades

Las entidades, actores, relaciones entre dichas entidades y funciones serán:

*Entidades*

* Propiedades
* Unidades
* Facturación
* Rubros
* Orden de trabajo
* Documentación
* Contabilidad
* Administración

*Actores*

* Administradores (Admin) : Son las cuentas que alimentan el sistema, cargando la información y generando las acciones para ser informadas al resto de los actores
* Proveedor : Ente externo que suministra un servicio y insumo necesario en el cumplimiento de una tare
* Personal : (RRHH) Personal propio de la organización bajo nomina o algún régimen administrativo
* Propietario : Dueño de una unidad habitacional o algún tipo de propiedad
* Inquilino : Usufructuante de una unidad habitacional o algún tipo de propiedad
* Publico : Ente externo que consulta información del sistema

***Observación*** : El acceso, autorización y utilización de las funciones del sistema, por parte de los actores, esta controlado por un modelo de Acces Control Role, en el que los actores deben estar registrados en el sistema y tener un nivel de rol para ejecutar dichas acciones.

Relaciones : Los administradores (admin) son los únicos responsables de ejecutar las funciones del sistema según las especificaciones funcionales descritas en este documento y bajo el control de rol

Las relaciones son complejas y numerosas por lo que no seran enumeradas en esta sección. Serán descritas de forma funcional y modeladas mediante UML para describir completamente los alcances d ellas mimas

Funciones : Son las acciones que cada entidad puede ejecutar, como p.e. crear una propiedad (createProperty), Nuevamente son ejecutadas por los actores bajo el control del rol

Al igual que las relaciones, las funciones son complejas y numerosas por lo que no seran enumeradas en esta sección. Serán descritas de forma taxita y modeladas mediante UML y modelos relaciones.

## Clase propiedad

Una “*propiedad”* es la unidad fundamental (singleton) de vivienda.

La propiedad puede estar compuestas por “*unidades”*, que son a su vez las bus-diviciones de dicha propiedad (p.e. propiedad : Edificio, unidad : apartamento)

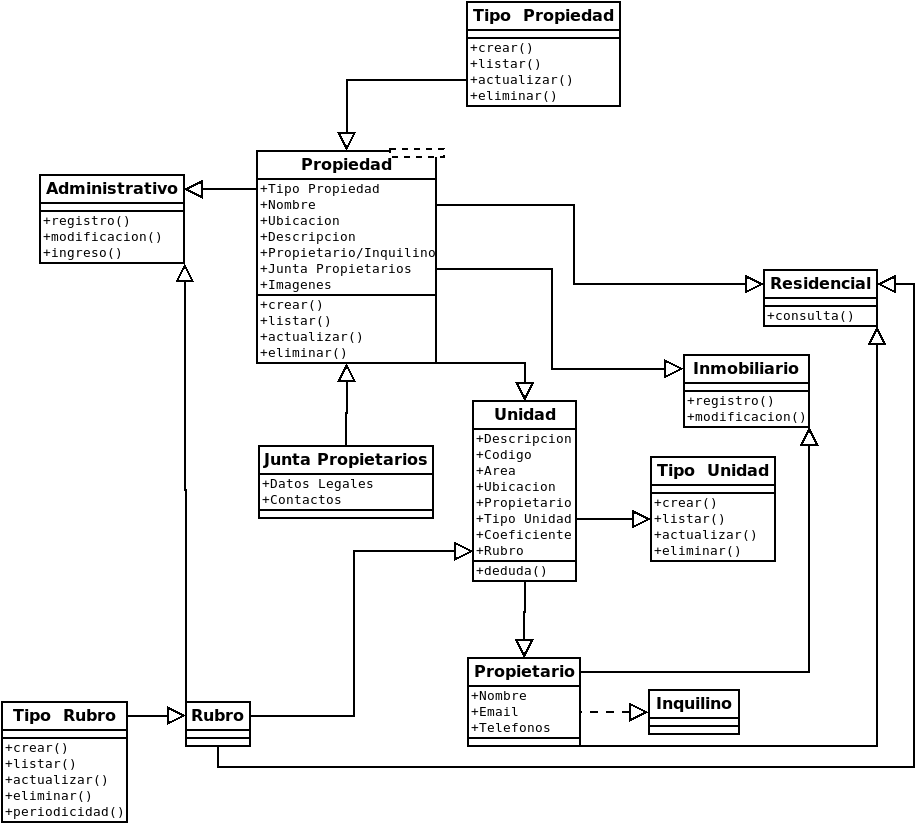
Las propiedades están asociadas a “*tipos de propiedades*” la cual describe que tipo de propiedad es. A su vez los tipos de propiedades son singletons por lo que responden a los métodos primitivos CRUD (crear, listas, actualizar, eliminar)

Una propiedad esta descrita por los siguientes atributos :

* Nombre → nombre de la propiedad. Campo de texto alfanumérico de 30 caracteres de longitud
* Ubicación → dirección física (postal) de la propiedad. Campo de texto de 255 caracteres de longitud
* Descripción → Datos relevantes d ella propiedad. Campo de texto de 255 caracteres de longitud
* Imágenes → lista de archivos almacenados físicamente en el servidor con imágenes de la propiedad. Solo soporta formatos jpg, img, png. Campo de texto alfanumérico de 30 caracteres de longitud que guarda el nombre del archivo – el archivo en si sera almacenado en los assets del sistema (carpeta img)

Una propiedad se relaciona directamente con :

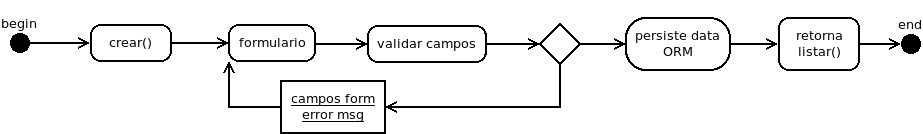
* Junta de propietarios → (junta\_propietarios\_id) : relación 1:1 (one-to-one), tipo asociación
* Unidad → (unidad\_id) : relación 1:n (one-to-mani), tipo asociación
* Tipo de Propiedad → (propiedad\_id) : relacion 1:n (one-to-mani), tipo asociación
* También existen unas relaciones con los módulos inmobiliaria, administrativo y residencia. Estas relaciones no son de tipo cardinal sino de tipo funcional y de flujo de datos, mas relacionadas a la lógica de negocio del sistema

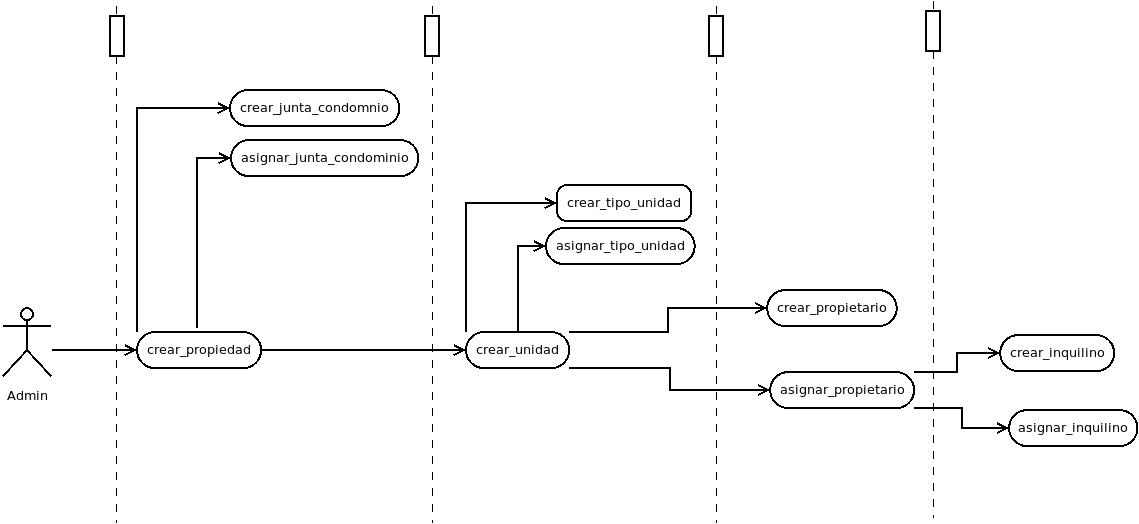


Una propiedad ejecuta los siguientes métodos :

* **crear**() → Registra un objeto propiedad para ser usado por cualquiera de los módulos principales (inmobiliarios, administrativo o residencial)

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **In** | **Out** | **Proceso** |
| crear() | null | * Objeto propiedad * Persiste en la DB el objeto propiedad | 1. Genera un formulario para capturar los atributos 2. Validar formulario (no vació, único) 3. Persistir la data 4. Retorna vista listado propiedades |

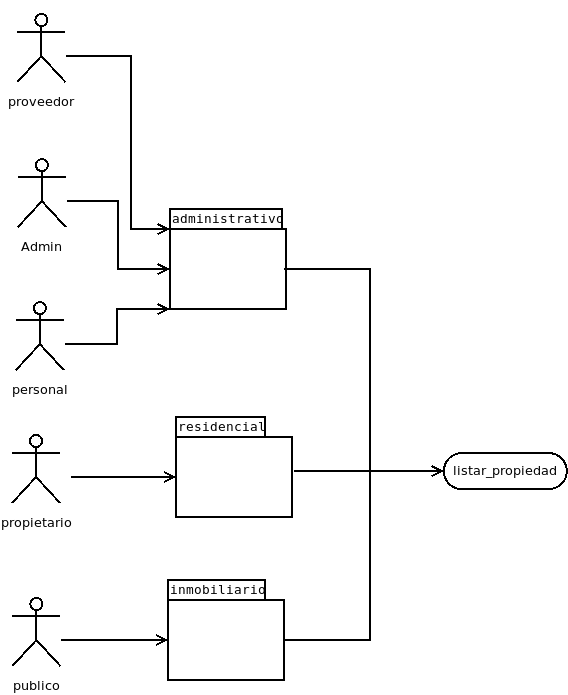




* **listar**() → Lista todos las propiedades.

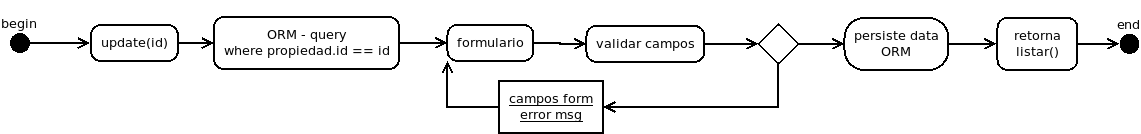
|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **In** | **Out** | **Proceso** |
| listar() | null | * Arreglo de objetos propiedad | 1. Query a la DB (ORM) extrayendo los objetos propiedad filtrado por limit(10) 2. Renderiza la vista listar |

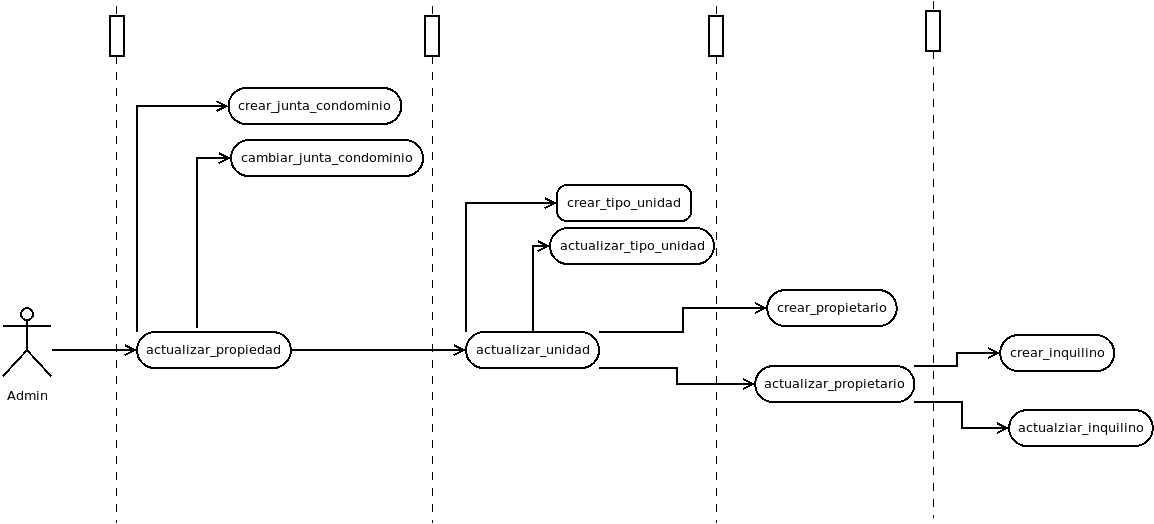




* actualizar(id)→ Modifica los campos descriptivos de una propiedad existente

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **In** | **Out** | **Proceso** |
| update(id) | id :integer | * objeto propiedad | 1. Query a la DB (ORM) extrayendo un objeto de tipo propiedad cuyo el parámetro id 2. Renderiza la data del objeto en el formulario de crear 3. Validar formulario (no vació) 4. Persistir la data 5. Retorna vista listado propiedades |





* eliminar(id) → Deshabilita una propiedad. Para mantener integridad referencial las unidades no se eliminan, sino que se deshabilitan de forma que no se pueden ejecutar ningún método sobre la propiedad que se encuentra en estado deshabilitado

## Clase tipoPropiedad

Un “tipoP*ropiedad”* es una clase atributo de la clase propiedad, con una relación 1:N (ont-to-many)

La clase tipoPropiedad describe los eventuales tipos de propiedad (p.e. Edificio, Casa, Finca, … )

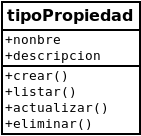
La clase tipoPropiedad son singletons por lo que responden a los métodos primitivos CRUD (crear, listas, actualizar, eliminar)

Un objeto tipoPropiedad esta descrita por los siguientes atributos :

* Nombre → nombre de la propiedad. Campo de texto alfanumérico de 30 caracteres de longitud
* Descripción → Datos relevantes d ella propiedad. Campo de texto de 255 caracteres de longitud

La clase tipoPropiedad se relaciona directamente con :

* Propiedad → (tipoPropiedad\_id) : relación 1:n (one-to-many), tipo asociación

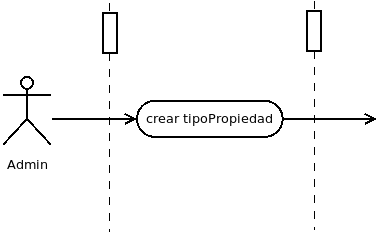


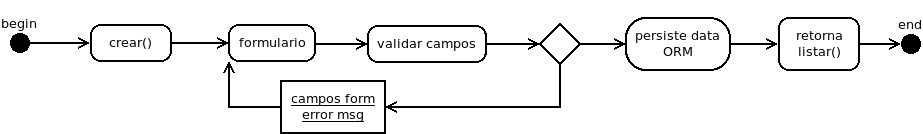
La clase tipoPropiedad y sus metodos solo puede ser invocada por un usuario de rol ***administrador***

La clase tipoPropiedad ejecuta los siguientes métodos :

* **crear**() → Registra un nuevo objeto tipoPropiedad y solo sera invocado por la clase propiedad.

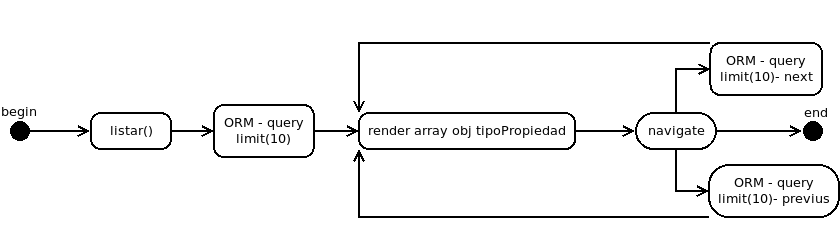
|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **In** | **Out** | **Proceso** |
| crear() | null | * Objeto tipoPropiedad * Persiste en la DB el objeto tipoPropiedad | 1. Genera un formulario para capturar los atributos 2. Validar formulario (no vació, único) 3. Persistir la data 4. Retorna vista tipoPropiedades |





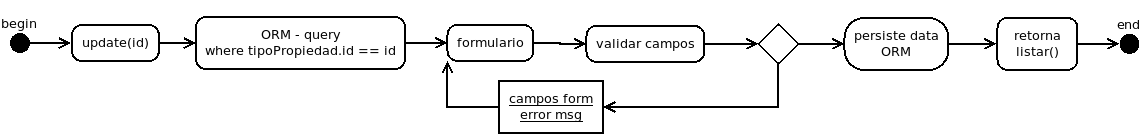
* **listar**() → Lista todos los objetos tipoPropieda.

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **In** | **Out** | **Proceso** |
| listar() | null | * Arreglo de objetos tipoPropiedad | 1. Query a la DB (ORM) extrayendo los objetos de tipoPropiedad filtrado por limit(10) 2. Renderiza la vista listar |



* actualizar(id)→ Modifica los campos descriptivos de objeto tipoPropiedad existente

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **In** | **Out** | **Proceso** |
| update(id) | id :integer | * objeto tipoPropiedad | 1. Query a la DB (ORM) extrayendo un objeto tipoPropiedad asociado al parámetro id 2. Renderiza la data del objeto en el formulario de crear 3. Validar formulario (no vació) 4. Persistir la data 5. Retorna vista listado tipoPropiedades |



* eliminar(id) → Deshabilita un objeto tipoPropiedad. Para mantener integridad referencial las unidades no se eliminan, sino que se deshabilitan de forma que no se pueden ejecutar ningún método sobre la propiedad que se encuentra en estado deshabilitado

## Clase juntaPropietarios

Un objeto “juntaPropietarios*”* es una clase atributo de la clase propiedad, con una relación 1:N (ont-to-many)

La clase juntaPropietarios describe la información de la actual junta de condominios de la propiedad, en caso de aplicar este atributo, por lo que el atributo puede tomar el valor **NULL**

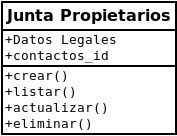
La clase juntaPropietarios son singletons por lo que responden a los métodos primitivos CRUD (crear, listas, actualizar, eliminar)

Un objeto juntaPropietarios esta descrita por los siguientes atributos :

* DatosLegales → nombre descriptivo de la juntaPropietarios. Campo de texto alfanumérico de 80 caracteres de longitud
* Contactos → Lista de objetos “*contactos”* asociado en un relanion N:1 (many-to-one : inversa)

La clase juntaPropietarios se relaciona directamente con :

* Contactos → (contactos\_id) : relación N:1 (many-to-one : inversa), tipo asociación

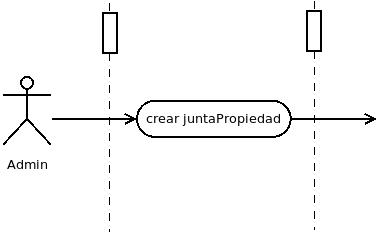


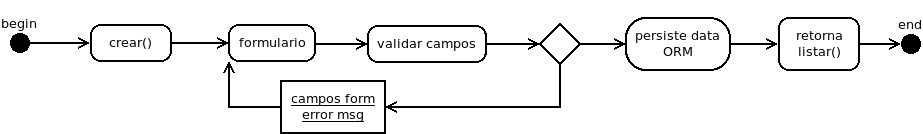
La clase *juntaPropietarios* es relacionada por la clase *propiedad* ypuede tomar el valor NULL en la relación

La clase juntaPropietarios ejecuta los siguientes métodos :

* **crear**() → Registra un nuevo objeto juntaPropietarios y solo sera invocado por la clase propiedad.

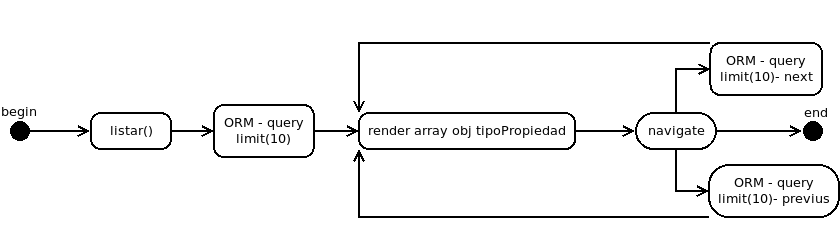
|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **In** | **Out** | **Proceso** |
| crear() | null | * Objeto juntaPropietarios * Persiste en la DB el objeto juntaPropietarios | 1. Genera un formulario para capturar los atributos 2. Validar formulario (no vació, único) 3. Persistir la data 4. Retorna vista juntaPropietarios |





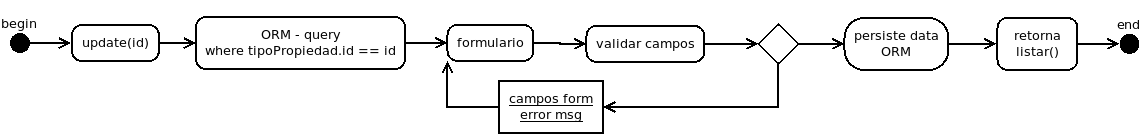
* **listar**() → Lista todos los objetos juntaPropietarios.

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **In** | **Out** | **Proceso** |
| listar() | null | * Arreglo de objetos juntaPropietarios | 1. Query a la DB (ORM) extrayendo los objetos de juntaPropietarios filtrado por limit(10) 2. Renderiza la vista listar |



* actualizar(id)→ Modifica los campos descriptivos de objeto tipoPropiedad existente

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **In** | **Out** | **Proceso** |
| update(id) | id :integer | * objeto tipoPropiedad | 1. Query a la DB (ORM) extrayendo un objeto tipoPropiedad asociado al parámetro id 2. Renderiza la data del objeto en el formulario de crear 3. Validar formulario (no vació) 4. Persistir la data 5. Retorna vista listado tipoPropiedades |



* eliminar(id) → Deshabilita un objeto tipoPropiedad. Para mantener integridad referencial las unidades no se eliminan, sino que se deshabilitan de forma que no se pueden ejecutar ningún método sobre la propiedad que se encuentra en estado deshabilitado