

Extracción de los sustantivos en la descripción del problema.

Clase/Objeto potencial	Categoría
Administración de Fincas	Unidad Organizativa
Comunidad	Unidad organizacional
Código Comunidad	Atributo
Dirección Comunidad	Atributo
Saldo Comunidad	Atributo
Usuario	Entidad externa o Rol
Vecino	Entidad externa o Rol
Trabajador	Entidad externa o Rol
Codito Inmueble	Atributo
Inmueble	Cosa
Cuotas pendientes	Atributo
Saldo Inmueble	Atributo
Tipo Inmueble	Atributo (Enumerado)
Nombre Trabajador	Atributo
DNI Trabajador	Atributo
Cargo Trabajador	Atributo
Nombre Usuario	Atributo
Dni Usuario	Atributo
Dirección Usuario	Atributo
Codigo Inmuebles	Atributos
Ccc	Atributo
Facturas	Cosa
Estado pago	Atributo
Nombre Empresa	Atributo
CIF	Atributo
Descripción Factura	Atributo
Fecha Factura	Atributo
Importe Factura	Atributo
Reunión (Junta)	Unidad Organizativa
Actas	Cosa
Fecha Acta	Atributo

Código Vecinos	Atributo
Orden Día	Atributo
Estado Acta	Atributo
Empresas	Entidad externa Rol

Selección de sustantivos como objetos/clases del sistema.

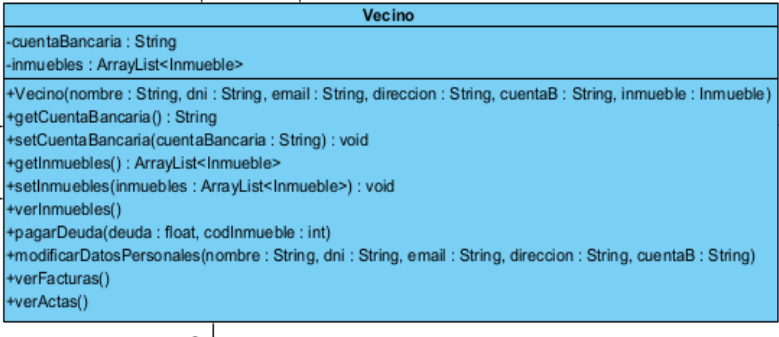
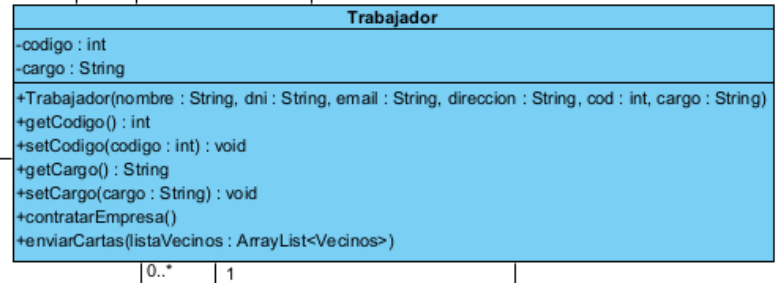
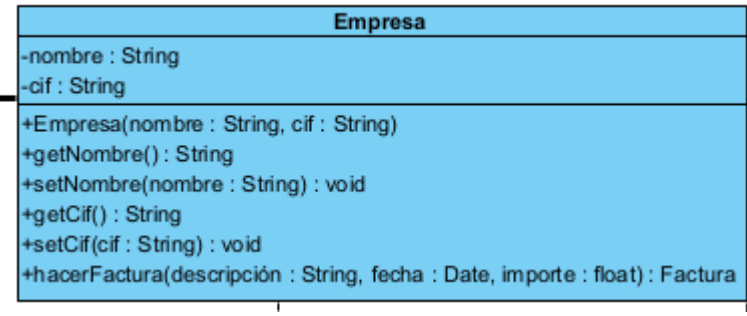
1. La información de la clase es necesaria para que el sistema funcione.
2. La clase posee un conjunto de atributos que podemos encontrar en cualquier ocurrencia de sus objetos. Si sólo aparece un atributo normalmente se rechazará y será añadido como atributo de otra clase.
3. La clase tiene un conjunto de operaciones identificables que pueden cambiar el valor de sus atributos y son comunes en cualquiera de sus objetos.
4. Es una entidad externa que consume o produce información esencial para la producción de cualquier solución en el sistema.

Clase/Objeto potencial	Criterios Aplicables
Administración de Fincas	1,2,3,4
Comunidad	1,2,3,4
Usuario	2,3,4
Vecino	2,3,4
Trabajador	2,3,4
Inmueble	1,2,3
Facturas	1,2,3
Actas	1,2,3
Empresas	2,3,4
Tipo Inmueble	3
Junta	1,4,3

Obtención de los atributos de los objetos.

Clase/Objeto potencial	Atributos
AdministacionDeFincas	Comunidades, trabajadores
Comunidad	código, dirección, vecinos, saldo, facturas, codigoAdministrador,juntas
Usuario	nombre, dni, email, dirección, codigosInmueble, cuentaBancaria
Vecino	CodigoInmueble
Trabajador	CodigoAdmnistrador, nombre, dni, cargo, administrador
Inmueble	CuotasPendientes, Saldo, CodigoInmueble, tipoInmueble
Facturas	NombreEmpresa, cif, descripción, fecha, importe, estado
Junta	Fecha, vecinosAsistieron, ordenDelDía, votación, codigoComunidad, trabajo
Empresas	NombreEmpresa, cif

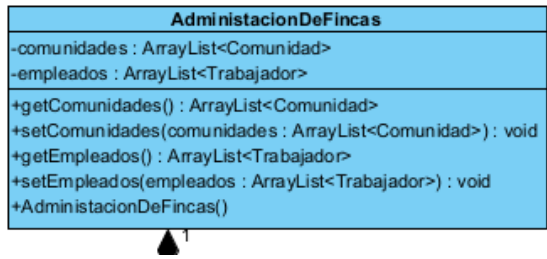
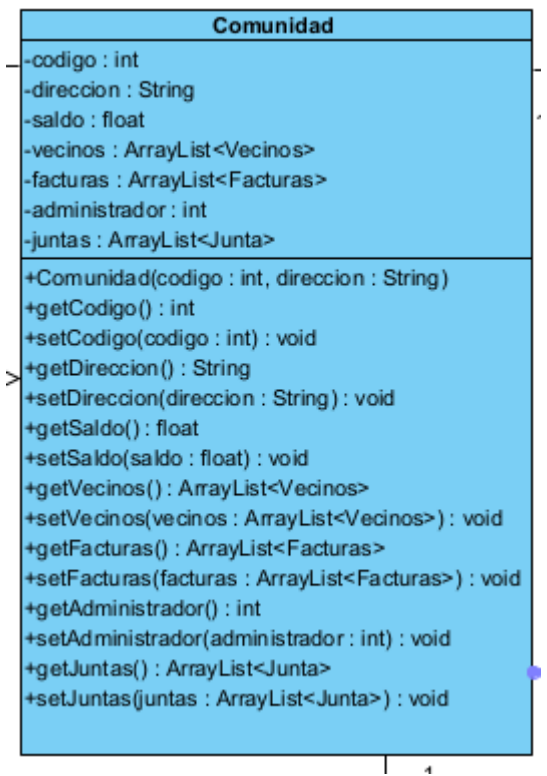
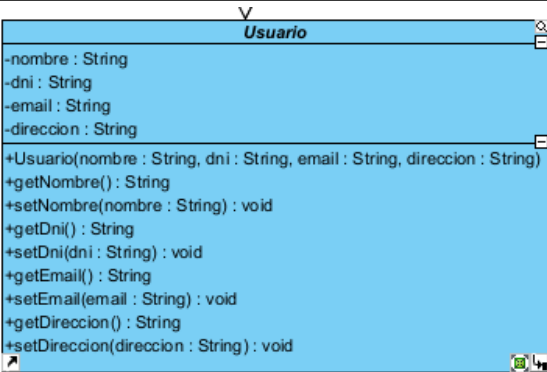
Obtención de los métodos.

Clase/Objeto potencial	Métodos	
Comunidad		
Usuario		
Vecino	verInmuebles() pagarDeuda() convocaReunion() modificarDatosPersonales() verFacturas() verActas()	 <pre> classDiagram class Vecino { -cuentaBancaria : String -inmuebles : ArrayList<Inmueble> +Vecino(nombre : String, dni : String, email : String, direccion : String, cuentaB : String, inmueble : Inmueble) +getCuentaBancaria() : String +setCuentaBancaria(cuentaBancaria : String) : void +getInmuebles() : ArrayList<Inmueble> +setInmuebles(inmuebles : ArrayList<Inmueble>) : void +verInmuebles() +pagarDeuda(deuda : float, codInmueble : int) +modificarDatosPersonales(nombre : String, dni : String, email : String, direccion : String, cuentaB : String) +verFacturas() +verActas() } </pre>
Trabajador	EnviarCartas() contratarServicio()	 <pre> classDiagram class Trabajador { -codigo : int -cargo : String +Trabajador(nombre : String, dni : String, email : String, direccion : String, cod : int, cargo : String) +getCodigo() : int +setCodigo(codigo : int) : void +getCargo() : String +setCargo(cargo : String) : void +contratarEmpresa() +enviarCartas(listaVecinos : ArrayList<Vecinos>) } </pre>
Inmueble		
Facturas		
Junta		
Empresas	HacerFactura()	 <pre> classDiagram class Empresa { -nombre : String -cif : String +Empresa(nombre : String, cif : String) +getNombre() : String +setNombre(nombre : String) : void +getCif() : String +setCif(cif : String) : void +hacerFactura(descripción : String, fecha : Date, importe : float) : Factura } </pre>
TipoInmueble	Enumerador	

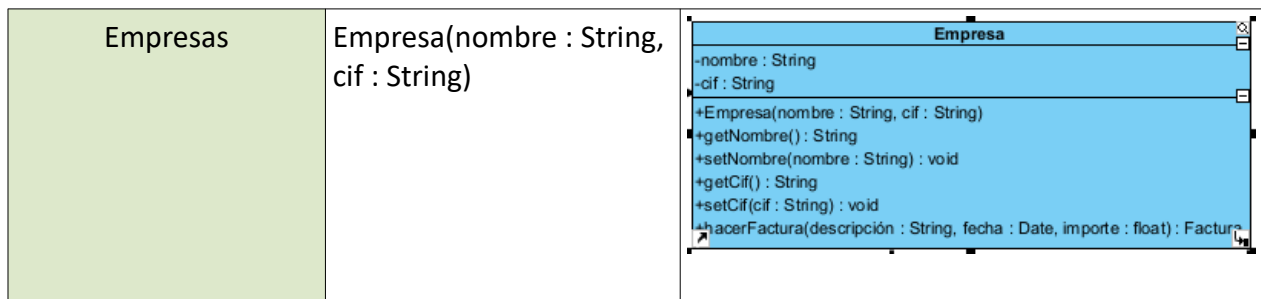
Obtención de las relaciones.

Clases que se relacionan	Explicación
AdministacionDeFincas – Trabajador	La relación que tiene con Trabajador es de tipo Composición ya que si la Empresa Administración de Fincas desaparece evidentemente desaparecen sus trabajadores.
AdministacionDeFincas - Comunidad	La relación con la clase Comunidad es de asociación y tiene dice que una administración puede administrar una o muchas comunidades
Comunidad – Facturas	La relación entre con Facturas es que 1 comunidad puede tener cero o muchas facturas, he decidido que sea cero ya que en el momento de creación de la comunidad no tienen facturas, pero lógicamente en cualquier momento las pueden tener
Facturas - Empresas	La relación es que una factura siempre va a pertenecer a una Empresa por eso la relación es de agregación y es de tipo 1 a n es decir un empresa va poder emitir tantas facturas como trabajos genere.
Comunidad - Junta	La relación con Junta es de composición ya que la junta de comunidad depende de la comunidad si esta desaparece desaparece la Junta
Comunidad – Vecino	La relación de comunidad con vecino es de asociación en la cual en una comunidad tiene que tener al menos un vecino.
Vecino - Inmueble	Un vecino puede tener 1 o mas inmueble
Vecino - Junta	La relación con Junta es que 1 vecino puede ir a ninguna reunión o a las que quiera
Vecino – Vecino (Presidente)	Esta relación indica que un vecino puede ser presidente
Comunidad - Trabajador	Esta relación de agregación es xq un trabajador tiene que ser el administrador de la comunidad o de varias.
Trabajador – Trabajador (administrador)	Esta relación indica que un trabajador puede ser administrador
Trabajador – Junta	Relación de 1 a n es decir que un trabajador tiene que asistir a la junta, pero a tantas juntas como comunidades tenga asignadas
Trabajador - Empresa	Un trabajador puede contratar los servicios de un empresa
Usuario – Vecino	Vecino hereda de Usuario
Usuario - Trabajador	Trabajador hereda de Usuario

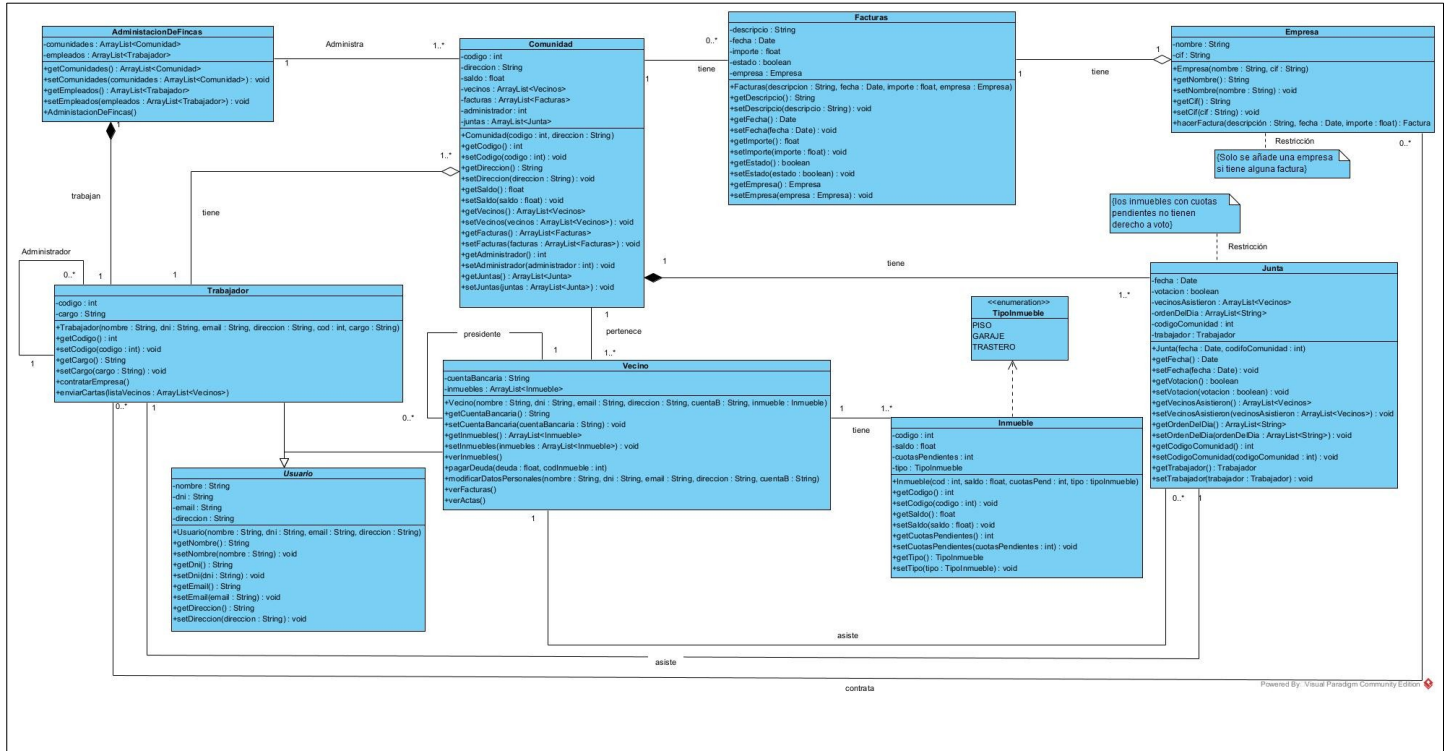
Añadir getters, setters y constructores.

Clase/Objeto	Constructor	Explicación
AdministracinDeFincas	AdministacionDeFincas()	 <pre> classDiagram class AdministacionDeFincas { -comunidades : ArrayList<Comunidad> -empleados : ArrayList<Trabajador> +getComunidades() : ArrayList<Comunidad> +setComunidades(comunidades : ArrayList<Comunidad>) : void +getEmpleados() : ArrayList<Trabajador> +setEmpleados(empleados : ArrayList<Trabajador>) : void +AdministacionDeFincas() } </pre>
Comunidad	Comunidad(codigo : int, direccion : String)	 <pre> classDiagram class Comunidad { -codigo : int -direccion : String -saldo : float -vecinos : ArrayList<Vecinos> -facturas : ArrayList<Facturas> -administrador : int -juntas : ArrayList<Junta> +Comunidad(codigo : int, direccion : String) +getCodigo() : int +setCodigo(codigo : int) : void +getDireccion() : String +setDireccion(direccion : String) : void +getSaldo() : float +setSaldo(saldo : float) : void +getVecinos() : ArrayList<Vecinos> +setVecinos(vecinos : ArrayList<Vecinos>) : void +getFacturas() : ArrayList<Facturas> +setFacturas(facturas : ArrayList<Facturas>) : void +getAdministrador() : int +setAdministrador(administrador : int) : void +getJuntas() : ArrayList<Junta> +setJuntas(juntas : ArrayList<Junta>) : void } </pre>
Usuario	Usuario(nombre : String, dni : String, email : String, direccion : String)	 <pre> classDiagram class Usuario { -nombre : String -dni : String -email : String -direccion : String +Usuario(nombre : String, dni : String, email : String, direccion : String) +getNombre() : String +setNombre(nombre : String) : void +getDni() : String +setDni(dni : String) : void +getEmail() : String +setEmail(email : String) : void +getDireccion() : String +setDireccion(direccion : String) : void } </pre>

Vecino	Vecino(nombre : String, dni : String, email : String, direccion : String, cuentaB : String, inmueble : Inmueble)	<pre> class Vecino { -cuentaBancaria : String -inmuebles : ArrayList<Inmueble> +Vecino(nombre : String, dni : String, email : String, direccion : String, cuentaB : String, inmueble : Inmueble) +getCuentaBancaria() : String +setCuentaBancaria(cuentaBancaria : String) : void +getInmuebles() : ArrayList<Inmueble> +setInmuebles(inmuebles : ArrayList<Inmueble>) : void +verInmuebles() +pagarDeuda(deuda : float, codInmueble : int) +modificarDatosPersonales(nombre : String, dni : String, email : String, direccion : String, cuentaB : String) +verFacturas() +verActas() } </pre>
Trabajador	Trabajador(nombre : String, dni : String, email : String, direccion : String, cod : int, cargo : String)	<pre> class Trabajador { -codigo : int -cargo : String +Trabajador(nombre : String, dni : String, email : String, direccion : String, cod : int, cargo : String) +getCodigo() : int +setCodigo(codigo : int) : void +getCargo() : String +setCargo(cargo : String) : void +contratarEmpresa() +enviarCartas(listaVecinos : ArrayList<Vecinos>) } </pre>
Inmueble	Inmueble(cod : int, saldo : float, cuotasPend : int, tipo : tipoInmueble)	<pre> class Inmueble { -codigo : int -saldo : float -cuotasPendientes : int -tipo : TipoInmueble +Inmueble(cod : int, saldo : float, cuotasPend : int, tipo : tipoInmueble) +getCodigo() : int +setCodigo(codigo : int) : void +getSaldo() : float +setSaldo(saldo : float) : void +getCuentasPendientes() : int +setCuotasPendientes(cuotasPendientes : int) : void +getTipo() : TipoInmueble +setTipo(tipo : TipoInmueble) : void } </pre>
Facturas	Facturas(descripcion : String, fecha : Date, importe : float, empresa : Empresa)	<pre> class Facturas { -descripcion : String -fecha : Date -importe : float -estado : boolean -empresa : Empresa +Facturas(descripcion : String, fecha : Date, importe : float, empresa : Empresa) +getDescripcion() : String +setDescripcion(descripcion : String) : void +getFecha() : Date +setFecha(fecha : Date) : void +getImporte() : float +setImporte(importe : float) : void +getEstado() : boolean +setEstado(estado : boolean) : void +getEmpresa() : Empresa +setEmpresa(empresa : Empresa) : void } </pre>
Junta	Junta(fecha : Date, codifoComunidad : int)	<pre> class Junta { -fecha : Date -votacion : boolean -vecinosAsistieron : ArrayList<Vecinos> -ordenDelDia : ArrayList<String> -codigoComunidad : int -trabajador : Trabajador +Junta(fecha : Date, codifoComunidad : int) +getFecha() : Date +setFecha(fecha : Date) : void +getVotacion() : boolean +setVotacion(votacion : boolean) : void +getVecinosAsistieron() : ArrayList<Vecinos> +setVecinosAsistieron(vecinosAsistieron : ArrayList<Vecinos>) : void +getOrdenDelDia() : ArrayList<String> +setOrdenDelDia(ordenDelDia : ArrayList<String>) : void +getCodigoComunidad() : int +setCodigoComunidad(codifoComunidad : int) : void +getTrabajador() : Trabajador +setTrabajador(trabajador : Trabajador) : void } </pre>



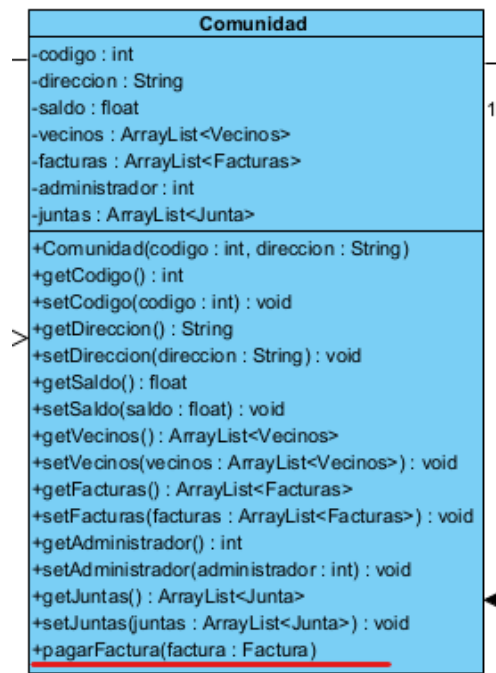
Primera versión del Diagrama



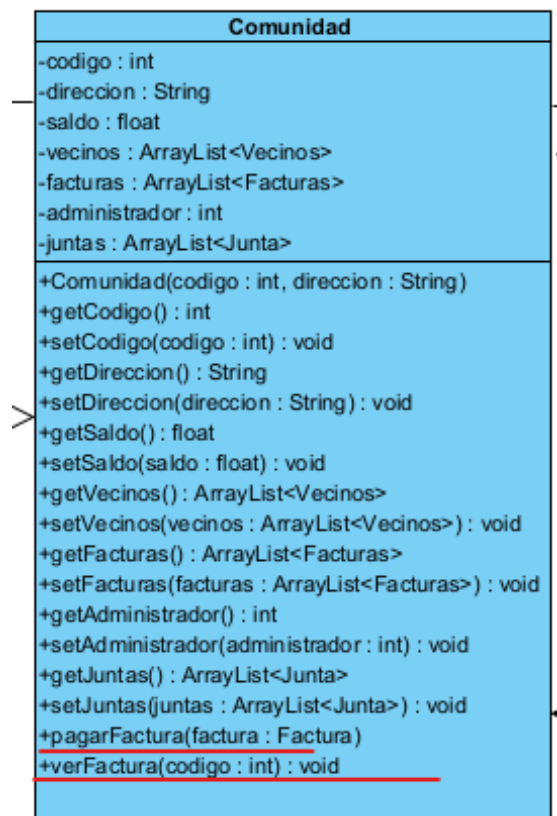
Primer refinamiento.

Una vez terminada la primera versión comenzamos a repasar el diagrama y el funcionamiento de este y comprobamos que nos faltan varias cosas y otras que tal y como fueron creadas inicialmente no tienen funcionalidad o dicha funcionalidad sería complicada de realizar.

La comunidad tiene que cuando una factura es pagada indicar que se ha pagado cambiando el check a true, por lo tanto añadimos en a clase comunidad el método para poder pagar una Factura.



Otra observación que vemos es que los usuarios tienen que tener la posibilidad de ver las Facturas y para ello tienen un método creado, dicho métodos no están en el sitio correcto ya que verdaderamente se crean y almacenan en la clase Comunidad, por lo tanto para poder acceder a esos datos tienen que estar en la clase Comunidad y así se tienen acceso a esos datos por lo tanto trasladamos los métodos verFacturas y a la clase Comunidad.



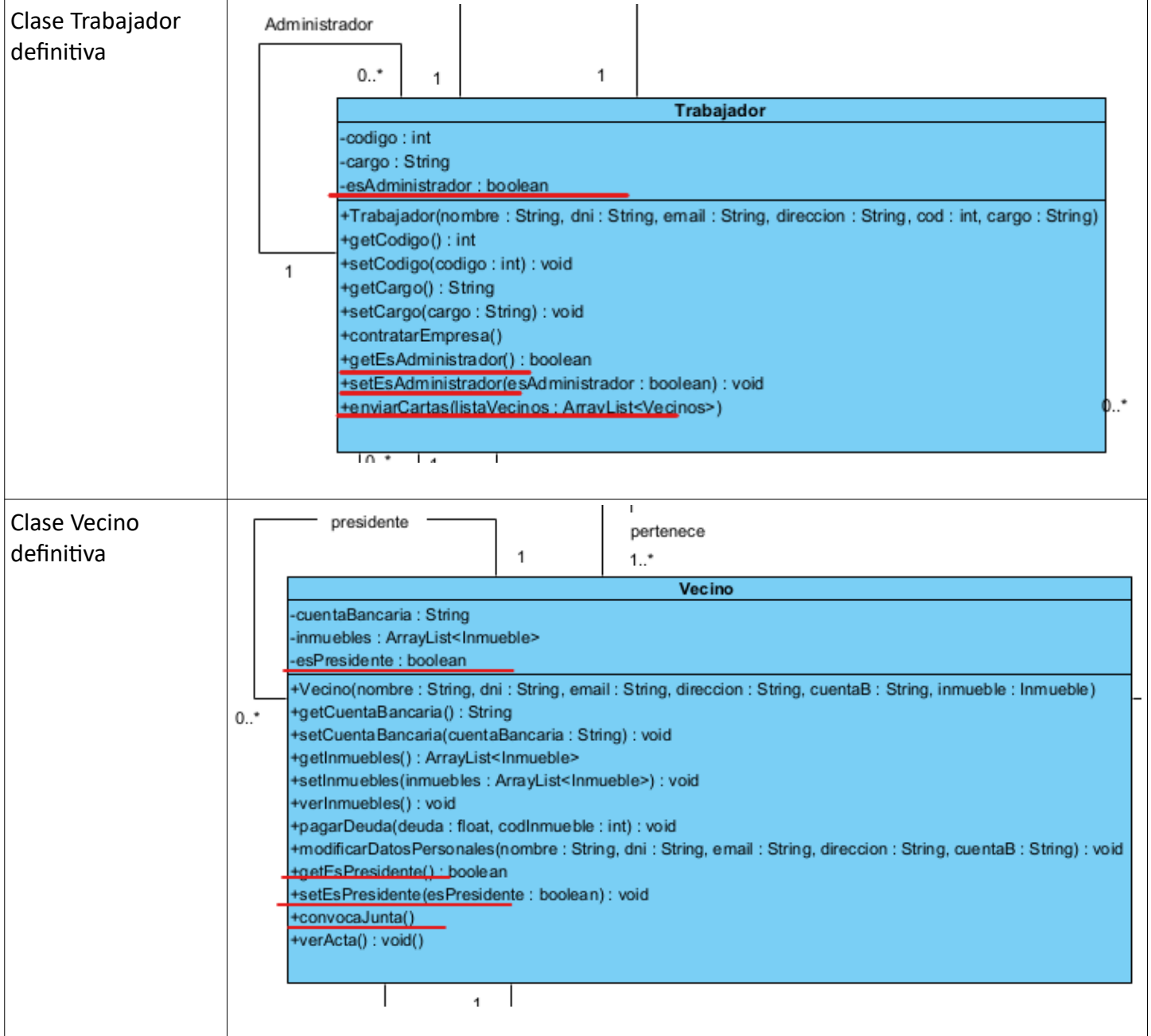
En facturas vemos que para buscar una factura no se tiene un dato que identifique una factura de forma única, por lo tanto añadimos un atributo CodigoFactura y modificamos el constructor y creamos los getter y setter correspondientes.

Facturas
-descripcion : String -fecha : Date -importe : float -estado : boolean -empresa : Empresa -codigoFactura : int
+Facturas(coid : int, descripcion : String, fecha : Date, importe : float, empresa : Empresa) +getDescripcion() : String +setDescripcion(descripcion : String) : void +getFecha() : Date +setFecha(fecha : Date) : void +getImporte() : float +setImporte(importe : float) : void +getEstado() : boolean +setEstado(estado : boolean) : void +getEmpresa() : Empresa +setEmpresa(empresa : Empresa) : void +getCodigoFactura() : int +setCodigoFactura(codigoFactura : int) : void

En la clase Junta tenemos que añadir un metodo que Genere un Acta y por ello creamos un método llamado GenerarActa.

Junta
-fecha : Date -votacion : boolean -vecinosAsistieron : ArrayList<Vecinos> -ordenDelDia : ArrayList<String> -codigoComunidad : int -trabajador : Trabajador
+Junta(fecha : Date, codifoComunidad : int) +getFecha() : Date +setFecha(fecha : Date) : void +getVotacion() : boolean +setVotacion(votacion : boolean) : void +getVecinosAsistieron() : ArrayList<Vecinos> +setVecinosAsistieron(vecinosAsistieron : ArrayList<Vecinos>) : void +getOrdenDelDia() : ArrayList<String> +setOrdenDelDia(ordenDelDia : ArrayList<String>) : void +getCodigoComunidad() : int +setCodigoComunidad(codigoComunidad : int) : void +getTrabajador() : Trabajador +setTrabajador(trabajador : Trabajador) : void +GenerarActa() : void

Repasando el texto inicial vemos que se nos ha pasado añadir una serie de funcionalidades o métodos para el Vecino que es presidente y el Trabajador que es administrador por lo tanto lo primero que añadimos a ambos es un atributo a modo de check que dice si dicho usuario es presidente y si un trabajador es administrador y así puede usar unos métodos únicos sin que estos generen una excepción. A los vecinos añadimos el método para convocar a la junta si es Presidente y al Trabajador añadimos el método envarcarta si es Administtrador



Otra modificación que se ha realizado es el cambiar la relación entre comunidad y Junta, ya que era de composición y si se quita una comunidad bien es cierto que se tendrían que quitar las juntas, pero como el modelo que se ha creado es el objeto junta quien guarda todos los datos de dichas reuniones y es el que genera las actas, pues para no duplicar datos creando un objeto ACTA pues declaramos esta relación de agregación y así no se destruirían los datos.

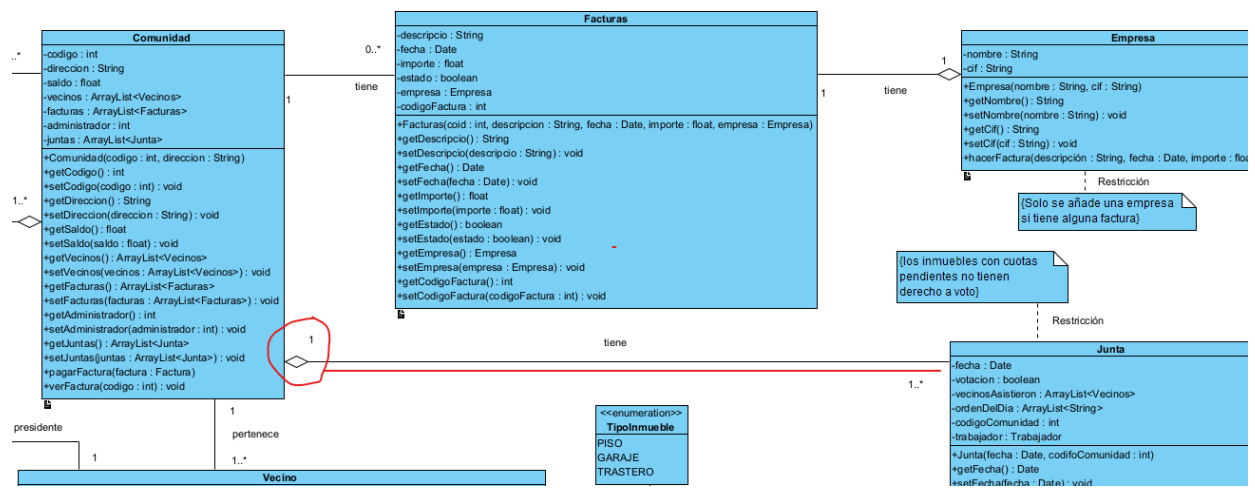
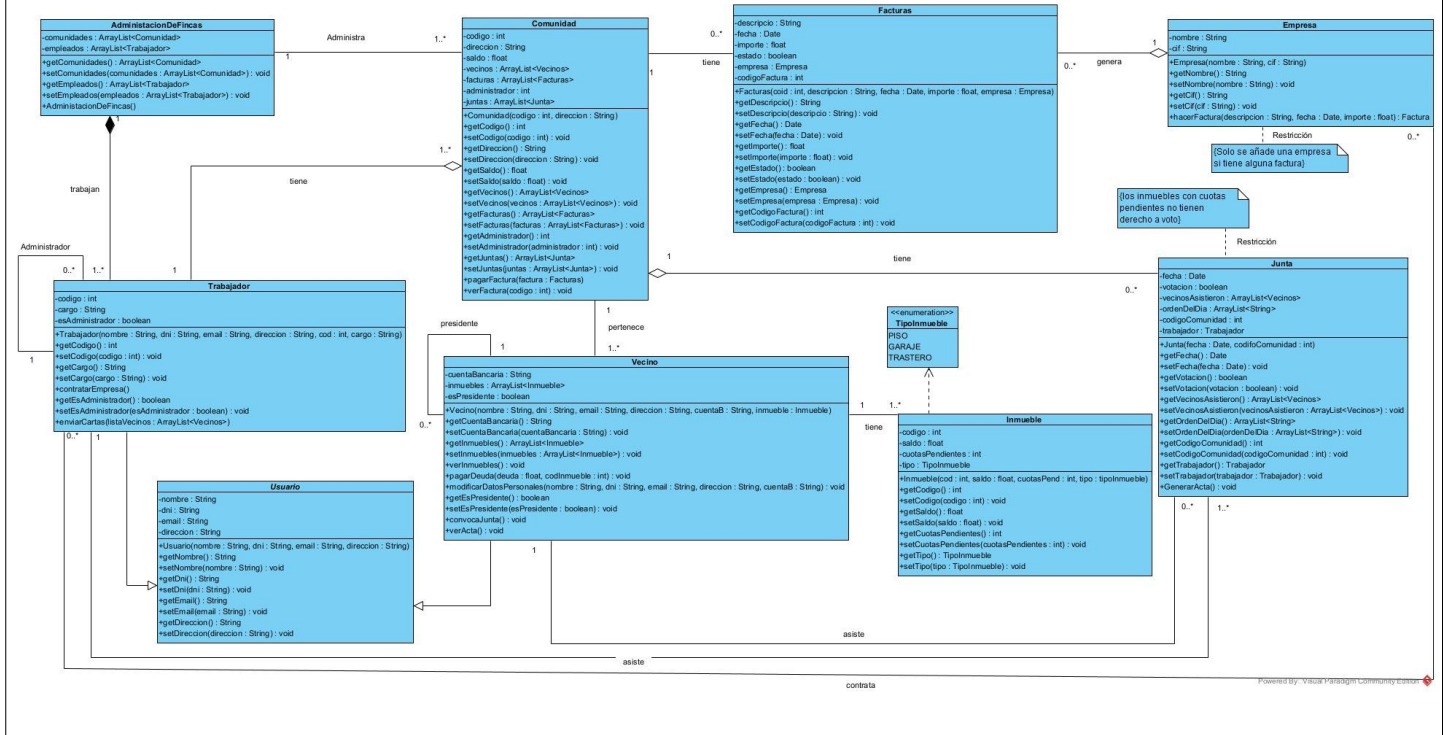
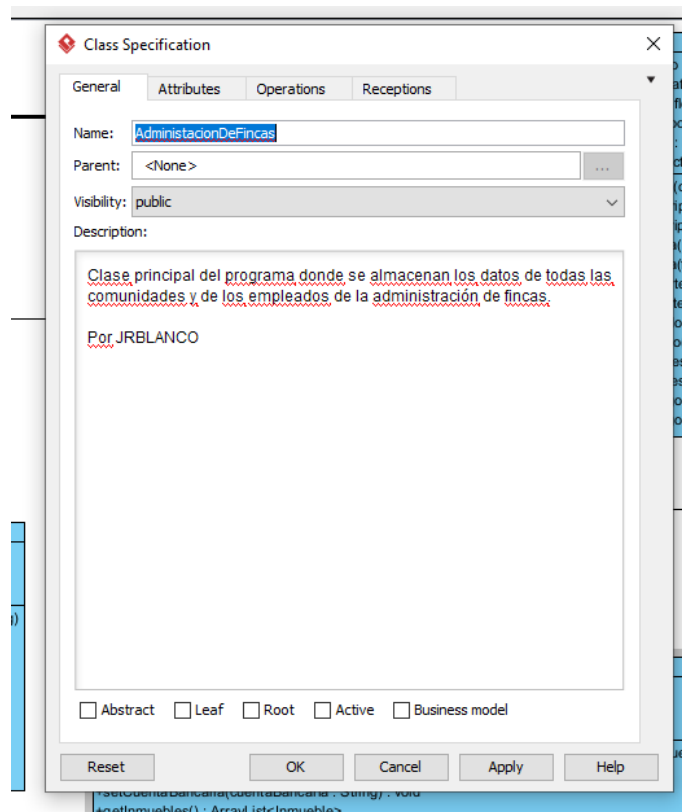


Diagrama final y es el que se adjunta en formato vpp

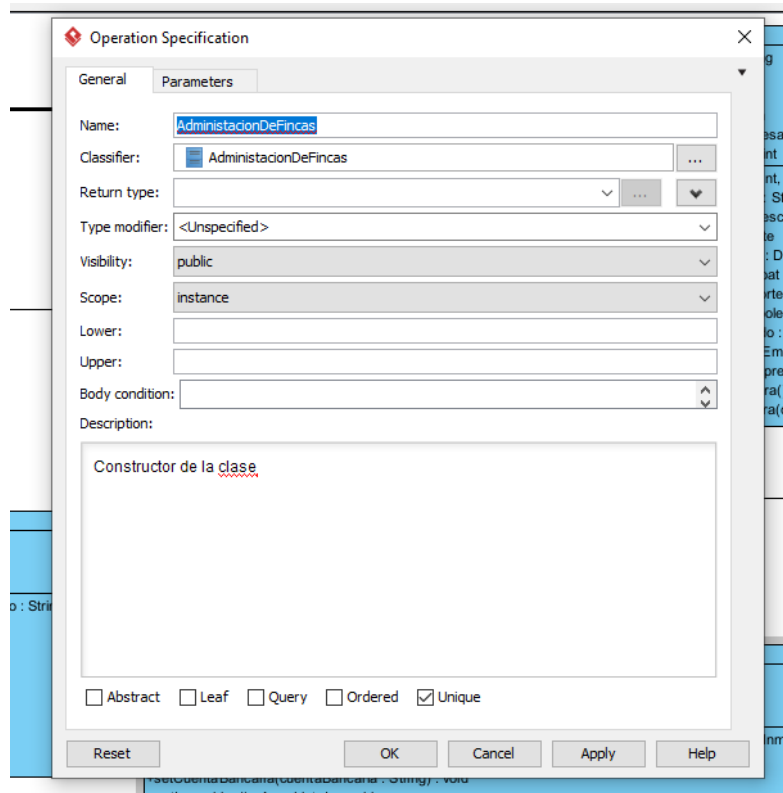


Documentación.

Para la documentación editamos en el VP y añadimos la que es cada clase para que así luego cuando se genera el código crea el javadoc.



Y también vamos método por método añadiendo comentarios para que cuando se genere el código se creen los comentarios de javadoc



Y al generar el código con el netbeans podemos comprobar que genera los comentarios javadoc

```
import java.util.ArrayList;

/**
 * Clase principal del programa donde se almacenan los datos de todas las comunidades
 * y de los empleados de la administración de fincas.
 */
// Por JRBLANCO
public class AdministracionDeFincas {

    private ArrayList<Comunidad> comunidades;
    private ArrayList<Trabajador> empleados;

    public ArrayList<Comunidad> getComunidades() {
        return this.comunidades;
    }

    /**
     * Constructor de la clase
     */
    public AdministracionDeFincas() {
        // TODO - implement AdministracionDeFincas.AdministracionDeFincas
        throw new UnsupportedOperationException();
    }

    /**
     *
     */
}
```

Clase	Descripción
AdministraciónDeFincas	Clase principal del programa donde se almacenan los datos de todas las comunidades y de los empleados de la administración de fincas.
Comunidad	Clase para cada comunidad de vecinos. El constructor de la clase los tiene que añadir un código y una dirección y posteriormente se van añadiendo los vecinos, facturas y las juntas.
Factura	Clase Facturas, para almacenar los datos de las facturas que emiten las empresas a cada comunidad por los servicios contratados.
Empresa	Clase Empresa almacena los datos de una empresa, es decir su nombre y cif
Trabajador	Clase Trabajador es para almacenar los datos de los trabajadores. Tiene la particularidad que también vale para identificar a un trabajador que sea administrador de alguna o varias comunidades y en función de si es administrador o no tiene una serie de métodos activos.
Vecino	Clase Vecino, para almacenar los datos de los vecinos y los inmuebles que este posee, tiene la particularidad que también identifica si un vecino es o no es presidente y si lo es da acceso a ciertas funciones que solo el presidente puede hacer.

Usuario	Clase Abstracta Usuario es la plantilla usada para los datos personales de los vecinos y los trabajadores
Inmueble	Clase Inmueble utilizada para almacenar los datos de cada tipo de inmueble
Tipoinmueble	Enumerador de Tipoinmueble, utilizado para diferenciar los diferentes inmuebles que tienen los vecinos, ya sea un piso, un garaje o un trastero.
Junta	Clase Junta, utilizada para guardar todos los datos de una junta de vecinos y que con esos datos genera las actas,