Módulo Profesional - Entornos de Desarrollo.

Diseño orientado a objetos. Elaboración de diagramas estructurales. Tarea para ED05.

José Francisco Márquez Díaz. 1 de abril de 2022.

Contenido

Enunciado

Se desea modelar una aplicación que permita gestionar la administración de una comunidad de vecinos. La empresa de administración de fincas puede gestionar varias comunidades cada una de ellas con un código, dirección, vecinos, saldo, facturas, y un código para el administrador responsable.

Descripción exacta del problema: "Comunidades de vecinos"

Los usuarios del sistema pueden ser los vecinos y los trabajadores de la empresa de administración de fincas. De los vecinos nos interesa su código de inmueble. Del inmueble nos interesa saber si tiene o no cuotas pendientes (si es o no moroso) que se visualiza con el saldo total el código del inmueble y el tipo (piso, garaje o trastero). Un mismo vecino puede ser propietario de varios inmuebles, por lo que se deben poder asignar varios inmuebles a un mismo vecino. De los trabajadores nos interesa conocer su nombre, su DNI, y su cargo. Existe un vecino concreto que es el presidente y existe un trabajador concreto que es el administrador.

Los vecinos pueden registrarse en el sitio web. Cuando un usuario se hace cliente debe proporcionar los siguientes datos: nombre completo, DNI, correo electrónico y dirección además de los códigos de los inmuebles de los que es propietario, así como el número de cuenta bancaria. Los vecinos pueden visualizar sus pisos y pagar su deuda siempre. Así mismo puede modificar sus datos personales.

Los usuarios (vecinos y trabajadores) navegan por la web para ver las facturas de la comunidad, que pueden estar pagadas o pendientes de pago. De las facturas nos interesa el nombre de la empresa, el CIF, descripción, fecha e importe.

Por otro lado, los usuarios deben poder visualizar las actas de las reuniones de vecinos, donde aparecerá la fecha, los códigos de los vecinos que han asistido, y los ítems del orden del día sabiendo si han sido aprobados o rechazados en junta, cada acta corresponde con una comunidad concreta.

Los trabajadores pueden solicitar servicios a empresas para que realicen un servicio en la comunidad. Dichas empresas generarán

facturas que podrán pagar con el saldo de la comunidad. No se da de alta en el sistema ninguna empresa que no haya generado factura.

Las reuniones de vecinos las convoca un vecino que es presidente de la comunidad que avisa al administrador de fincas que es quien convoca oficialmente la reunión mandando una carta a cada vecino. sería un tipo de empleado encargado de realizar el conteo de vecinos que asisten a la comunidad y contar el número de votos a favor y en contra de cada punto del orden del día. Los pisos con cuotas pendientes (morosos) no tienen derecho a voto en la reunión.

Acrónimo: Código de Identificación Fiscal

Tu tarea consiste en elaborar el diagrama de clases, así como la documentación para el análisis de una aplicación que implemente la gestión por internet de la comunidad de vecinos con la aplicación Visual Paradigm. Debido a las restricciones de la aplicación con el tiempo de evaluación, no es obligatorio generar el código de la aplicación, pero si debes importar el proyecto creado VP-UML en un proyecto de NetBeans. Para documentar el proceso deberás entregar un documento de texto con los siguientes puntos:

- **Extracción de los sustantivos en la descripción del problema.**
- **Selección de sustantivos como objetos/clases del sistema.**
- **Obtención de los atributos de los objetos.**
- **Obtención de los métodos.**
- **Obtención de las relaciones.**
- **Añadir getters, setters y constructores.**
- **Primer refinamiento.**
- **Documentación.**

1. Extracción de los sustantivos de la descripción del problema:

Clase/objeto potencial	Categoría
Comunidad de vecinos	Unidad organizacional
Empresa	Unidad organizacional
Finca	Cosa
Código	Atributo
Dirección	Cosa
Vecino	Entidad externa o rol
Saldo	Atributo
Factura	Cosa
Administrador	Entidad externa o rol
Usuario	Entidad externa o rol
Trabajador	Entidad externa o rol
Código inmueble	Atributo
Cuota	Atributo
Saldo total	Atributo
Inmueble	Cosa
Tipo inmueble	Atributo
Propietario	Entidad externa o rol
Nombre	Atributo
DNI	Atributo
Cargo	Atributo
Presidente	Entidad externa o rol
Cliente	Entidad externa o rol
Correo electrónico	Atributo
Cuenta bancaria	Atributo
Piso	Cosa
Factura	Cosa
Nombre empresa	Atributo
CIF	Atributo
Descripción	Atributo
Fecha	Atributo
Importe	Atributo
Acta	Cosa
Reunión de vecinos	Cosa
Ítem del día	Cosa
Servicio	Cosa
Carta	Cosa
Cuota pendiente	Atributo

Nota: Los refinamientos propuestos aparecen reflejados en las tablas.

2. Selección de sustantivos como objetos/clases del sistema:

Los criterios que hay que seguir para determinar si los sustantivos, que aparecen en el enunciado, son clases:

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
Comunidad	1, 2,
Usuario	1, 2, 3,
Vecino	1, 2, 3, 4
Trabajador	1, 2, 3, 4
Administrador	1, 2, 3, 4
Presidente	1, 3, 4
Acta	1, 2
Ítem del día	1, 2, 3
Factura	2, 3,
Dirección	2
Inmueble	1, 2, 3,
Empresa	2, 3,

3. Tabla de relación de las clases u objetos con sus atributos:

Clase/objeto potencial	Atributos
Comunidad	Código, saldo.
Usuario	Nombre, apellidos, DNI, correo electrónico, número de cuenta bancaria.
Vecino	Código de inmueble
Trabajador	Cargo.
Administrador	Código Administrador
Presidente	
Acta	Fecha, códigosVecinos.

Ítem del día	Aprobado.
Factura	NombreEmpresa, fecha, descripción, CIF, importe.
Dirección	Tipo de vía, vía, número, piso, planta, letra, código postal, municipio, provincia.
Inmueble	Código, tipo.
Empresa	NombreEmpresa, CIF.

4. Tabla de clases u objetos del sistema con sus posibles métodos.

Clase/objeto potencial	Métodos
Comunidad	
Usuario	hacerseCliente(nombre : String, apellidos : String, DNI : String, correo electrónico : String, …) : void verFactura(): Factura visualizarActa() : Acta
Vecino	registroWeb() : Boolean visualizarPiso() : String pagarDeuda() : Boolean modificarDatos() : Boolean
Trabajador	solicitarServicio() : Boolean
Administrador	pagarFactura() : int notificaReunión() : Boolean darEmpresaAlta(factura : Factura) : Boolean
Presidente	convocaReunión(): void
Acta	
Ítem del día	aprobarÍtem(): void
Factura	estadoFactura() : int
Dirección	
Inmueble	hayPendientes() : Boolean
Empresa	generarFactura(NombreEmpresa : String, fecha : Date, descripción : String, CIF : String, importe : int) : Factura

5. Obtención de las relaciones.

- La relación entre Vecino e Inmueble es una asociación. En esta relación existe multiplicidad, ya que podemos decir que un vecino puede ser propietario de uno o más inmuebles y que un inmueble pertenece a un único vecino (obviamos la posibilidad de multitutilaridad por representar de forma más didáctica la relación).

- La relación entre Comunidad y Dirección, así como la relación entre Usuario y Dirección son relaciones de asociación. En este caso no existe multiplicidad, ya que una comunidad/Usuario tiene una única dirección y una dirección está asociada a una única Comunidad/Usuario.

- La relación entre Empresa y Factura es una asociación. En esta relación existe multiplicidad, ya que una empresa puede generar una o muchas facturas y que una factura es generada por una única empresa.

- La relación entre Empresa y Trabajador es una asociación. En esta relación existe multiplicidad, ya que un trabajador puede solicitar servicios a una empresa y una empresa puede proporcionar servicios a un trabajador.

- La relación entre Empresa y Comunidad es una asociación. En esta relación existe multiplicidad, ya que una comunidad puede recibir facturas de una empresa y una empresa puede generar facturas a una comunidad.

- La relación entre Usuario y Comunidad es una agregación. En esta relación existe multiplicidad, ya que una comunidad estará conformada por muchos usuarios y un usuario podrá pertenecer a muchas comunidades (por ejemplo, un vecino con varios inmuebles o un trabajador que presta servicios a varias comunidades).

- La relación entre Administrador y Factura es una asociación. En esta relación existe multiplicidad, ya que un administrador puede pagar muchas facturas y una factura solo puede ser pagada por un administrador.

- La relación entre Acta y Comunidad es una asociación. En esta relación existe multiplicidad, ya que una comunidad puede tener muchas actas de reuniones y un acta de reunión solo puede pertenecer a una comunidad.

- La relación entre Acta e Ítem del día es una composición. En este caso, no tiene sentido un ítem del día si no está asociado a un acta. Así pues, un acta está formada por ítems del día. En esta relación existe multiplicidad, ya que un acta estará conformada por varios ítems del día y un ítem del día solo podrá pertenecer a un acta.

- La relación entre Usuario y Vecino y la relación entre Usuario y Trabajador son generalizaciones/especializaciones ya que un vecino/trabajador es un usuario que además de compartir los atributos y métodos de este, se especializa presentando atributos y métodos propios. O, dicho de otro modo, un usuario es una generalización de vecino y trabajador.

- La relación entre Vecino y Presidente y la relación entre Trabajador y Administrador son generalizaciones/especializaciones ya que un presidente/administrador es un vecino/trabajador respectivamente, que además de compartir los atributos y métodos de estos, se especializan presentando atributos y métodos propios. O, dicho de otro modo, un vecino es una generalización de presidente y un administrador es una generalización de trabajador.

6. Añadir getters, setters y constructores.

Se han añadido los métodos Getter y Setter necesarios para consultar y modificar los atributos de clase privados. Además, se incluyen los constructores requeriidos para crear objetos de la clase.

Ver diagrama adjunto creado en Visual Paradigm.

7. Primer refinamiento.

Como refinamiento se ha decidido sacar el atributo dirección a una entidad propia. A esta se le han añadido los atributos clásicos de una dirección (Tipo

de vía, vía, número, piso, planta, letra, código postal, municipio, provincia). De este modo, es posible una mejor gestión de las direcciones.

La entidad dirección se relaciona tanto con la entidad Comunidad como con la entidad Usuario.

8. Documentación adicional.

A continuación, se detalla la información relativa a cada clase, atributo y método de la aplicación:

Comunidad: Es el núcleo central del sistema. Relaciona a los propietarios, trabajadores, empresas de servicios e inmuebles. Sus atributos son el código de la comunidad y el saldo del que dispone para el pago de las facturas.

Usuario: Generalización para agrupar las características comunes de vecinos y trabajadores. Además, en el momento en que un usuario contenga los datos necesarios, representará a un cliente. De un usuario interesa conocer su nombre, apellidos, DNI, correo electrónico y número de cuenta bancaria. La clase Usuario posee los siguientes métodos:

- **hacerseCliente**(nombre : String, apellidos : String, DNI : String, correo electrónico : String, …) : void
  Este método permite que un Usuario (vecino o trabajador), se convierta en un cliente.
- **verFactura**(): Factura
  Permite visualizar una factura.
- **visualizarActa**() : Acta
  Permite visualizar el acta de una reunión.

Vecino: Es un tipo de Usuario. Representa a un propietario de inmuebles. Por lo que debe proporcionar los códigos de sus bienes inmuebles. Los métodos de esta clase son:

- **registroWeb**() : Boolean
  Permite registrarse en la aplicación para gestionar las propiedades
- **visualizarPiso**() : String
  Permite visualizar el estado de un piso.
- **pagarDeuda**() : Boolean
  Permite pagar deudas de un piso.
- **modificarDatos**() : Boolean
  Permite modificar los datos personales.

Trabajador: Es un tipo de Usuario. Representa a un empleado de la comunidad de vecinos. Necesitamos conocer su cargo. Pueden solicitar servicios a otras empresas para realizar trabajos en la comunidad. Los métodos de esta clase son:

- **solicitarServicio**() : Boolean
  Permite contratar servicios de empresas externas.

Administrador: Es un tipo de trabajador. Administra la gestión de la comunidad. Posee un código de administrador. Se encarga de notificar las reuniones de vecinos, pagar las facturas y dar de alta a nuevas empresas de servicios. Los métodos de esta clase son:

- **pagarFactura**() : int
  Permite pagar una factura generada por una empresa de servicios.
- **notificaReunión**() : Boolean
  Permite notificar a los vecinos las reuniones.
- **darEmpresaAlta**(factura : Factura) : Boolean
  Permite dar de alta en la comunidad a una empresa de servicios.

Presidente: Es un tipo de vecino. No posee atributos propios (solo los heredados de vecino). Se encarga de convocar las reuniones de vecinos. Los métodos de esta clase son:

- **convocaReunión**(): void
  Permite convocar una reunión de vecinos. Se notificará la reunión al administrador que será el encargado de hacer llegar dicha información a los vecinos.

Acta: Representa los contenidos de las reuniones de vecinos de cada comunidad. Sus atributos principales son la fecha de celebración y el listado de los códigos de vecinos que asisten. Está compuesta por los ítems del día.

Ítem del día: Un conjunto de ítems compone un acta. Su atributo principal es aprobado, que indica si el ítem ha sido aprobado en reunión por la mayoría de los vecinos. Los principales métodos de la clase son:

- **aprobarÍtem**(): void
  Permite aprobar un ítem si ha alcanzado mayoría de votos a favor en la votación.

Factura: Representan facturas de servicios sobre la comunidad, generadas por empresas externas. Sus atributos principales son: nombreEmpresa, fecha, descripción, CIF e importe. Los métodos principales de la clase son:

- **estadoFactura**() : int
  Permite ver el estado de una factura.

Dirección: Representa los datos de una dirección. Sus atributos son: Tipo de vía, vía, número, piso, planta, letra, código postal, municipio, provincia. Está relacionada con Comunidad y con Usuario.

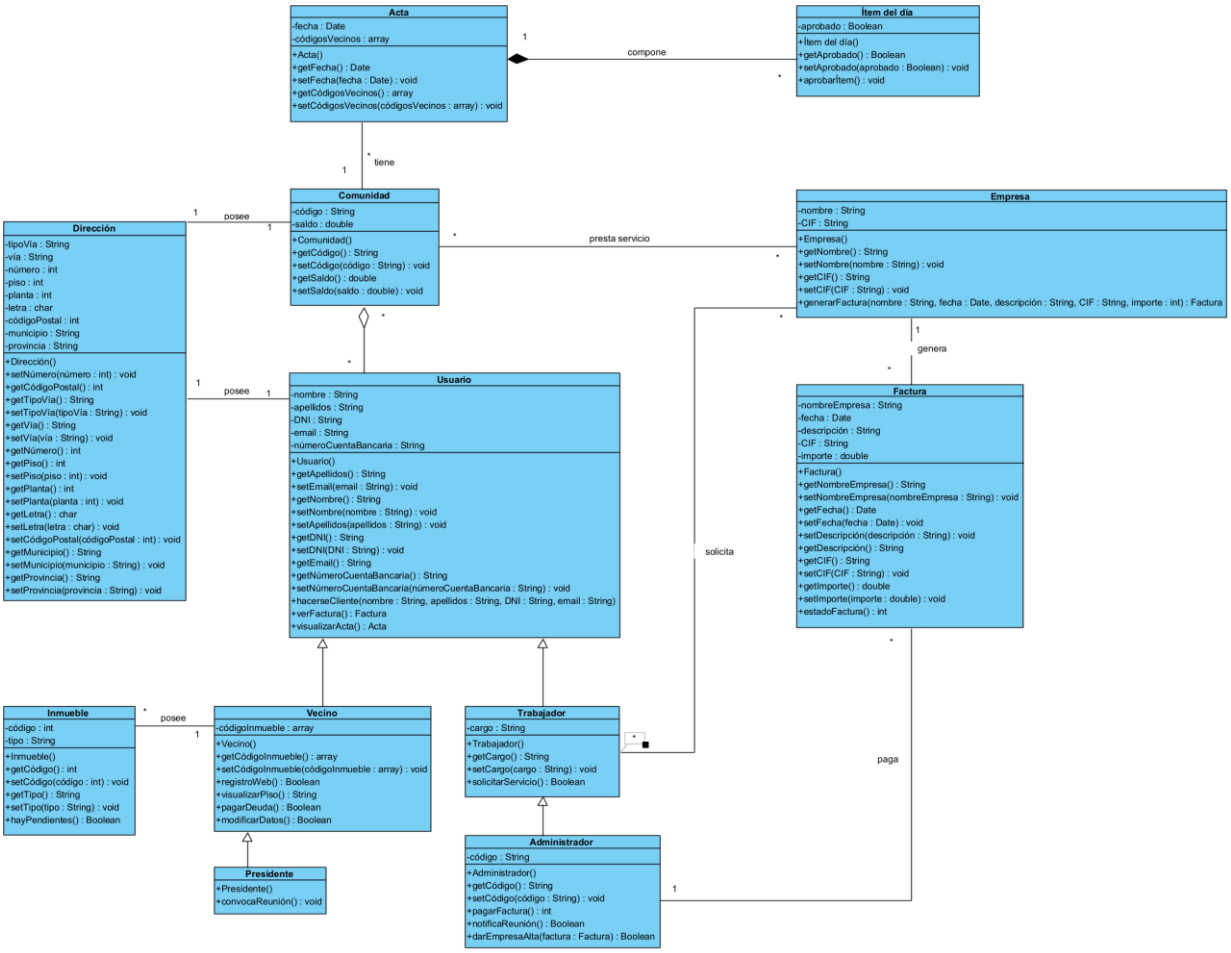
Inmueble: Representa una propiedad perteneciente a un vecino. Sus atributos principales son el código del inmueble y su tipo. Los métodos principales de la clase son:

- **hayPendientes**() : Boolean
  Permite conocer si existen cuotas pendientes de pago.

Empresa: Representa empresas de servicios externas a la comunidad. Podrán registrarse y pasar a prestar servicios. Sus atributos principales son: NombreEmpresa y CIF. Los métodos principales de la clase son:

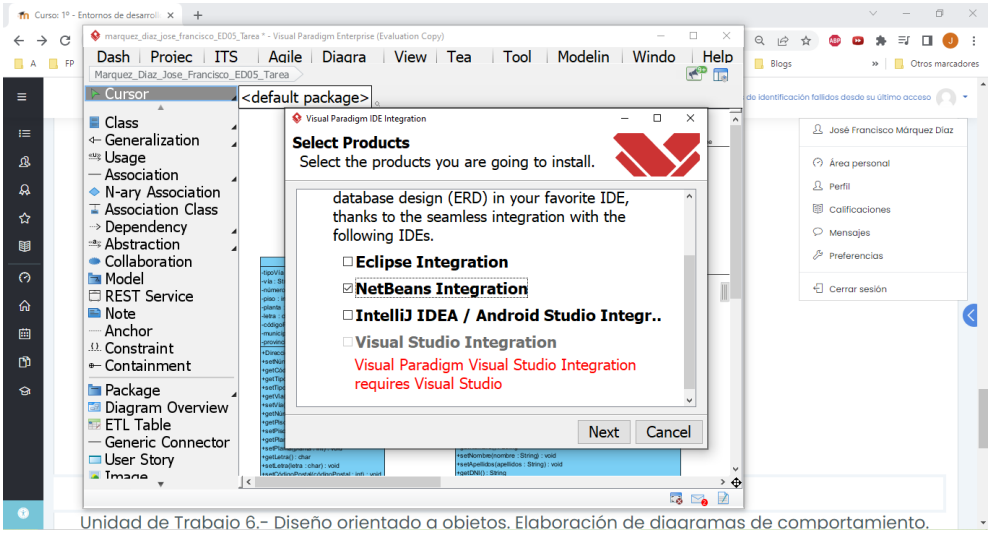
- **generarFactura**(NombreEmpresa String, fecha : Date, descripción : String, CIF : String, importe : int) : Factura
  Permite generar una factura con motivo de un servicio prestado a la comunidad de vecinos.

Imagen del diagrama creado con Visual Paradigm Enterprise

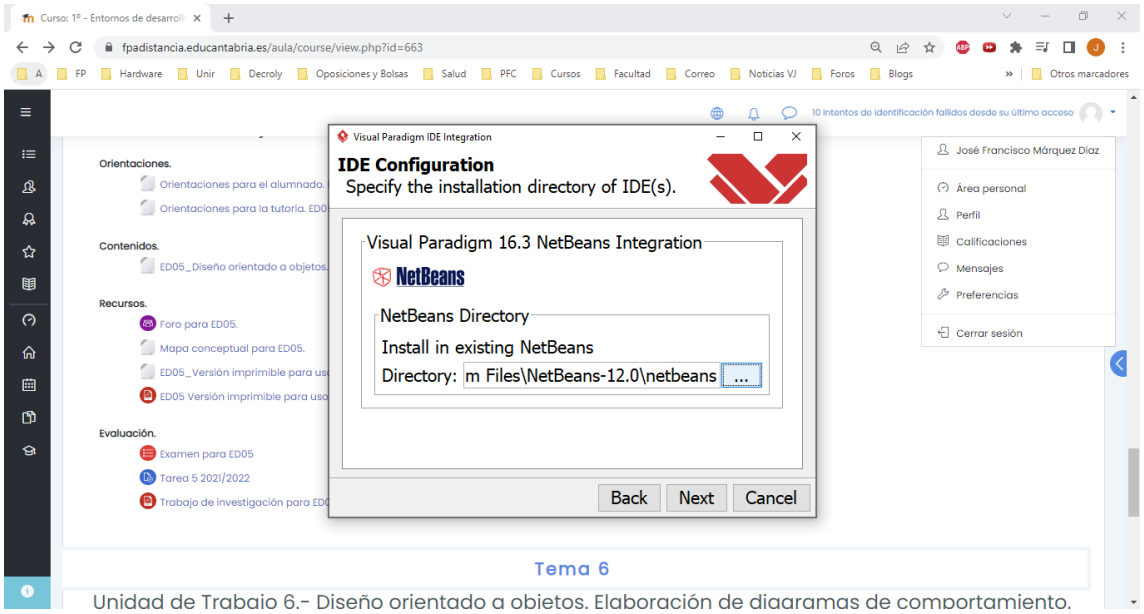


Incluimos el proyecto en NetBeans.

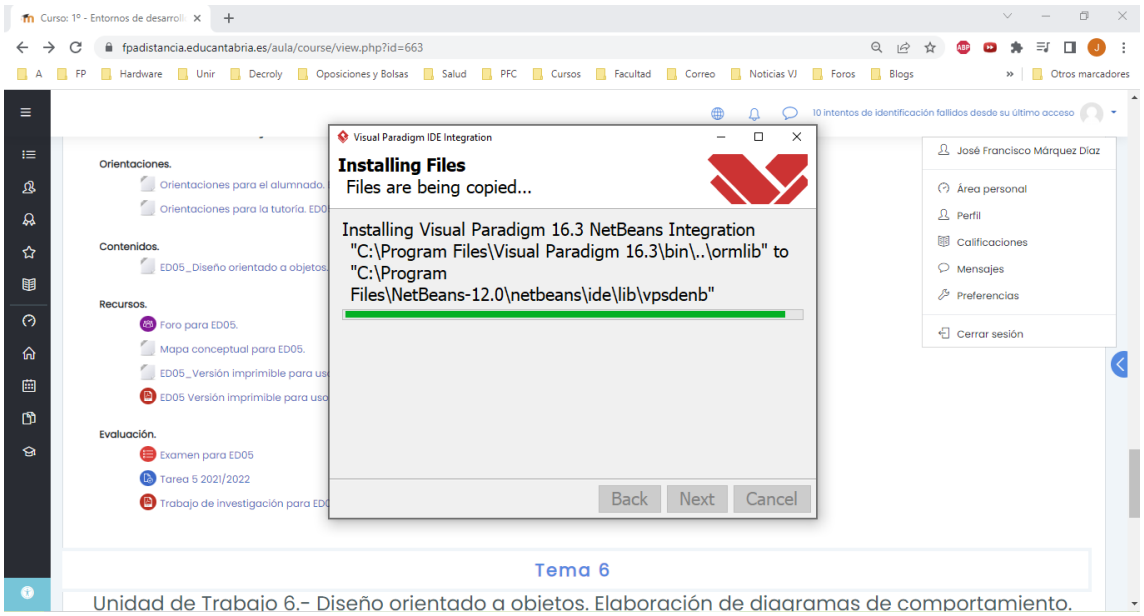
Para ello nos vamos a la pestaña de *Window* del Visual Paradigm y seleccionamos la opción *Integration -> IDE Integration* y marcamos la casilla de NetBeans.



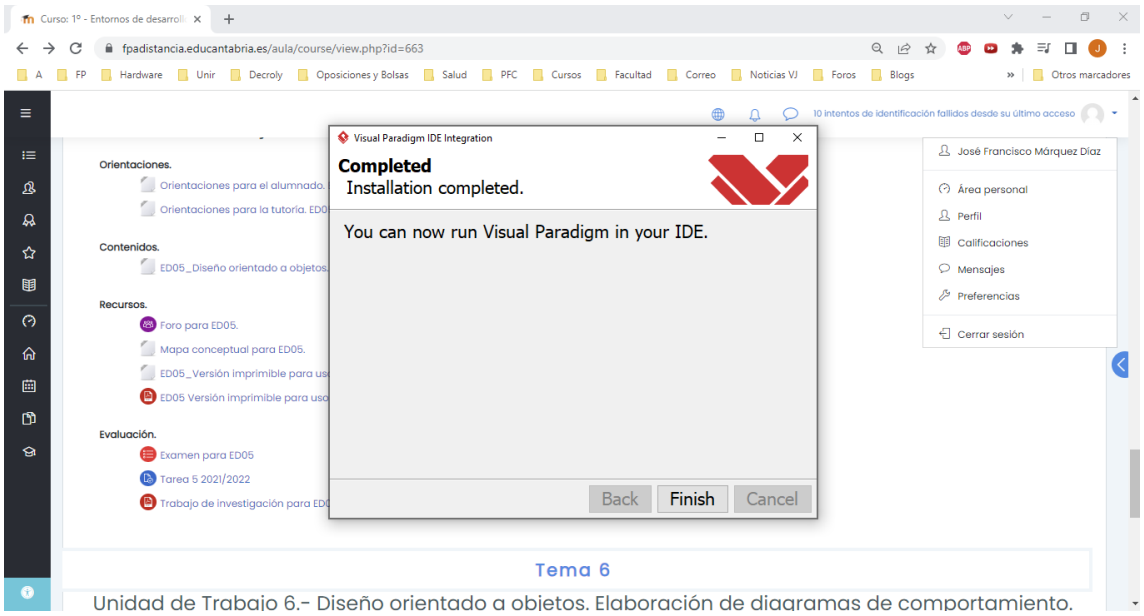
Seleccionamos la ruta de nuestro NetBeans y pulsamos en *Next*.



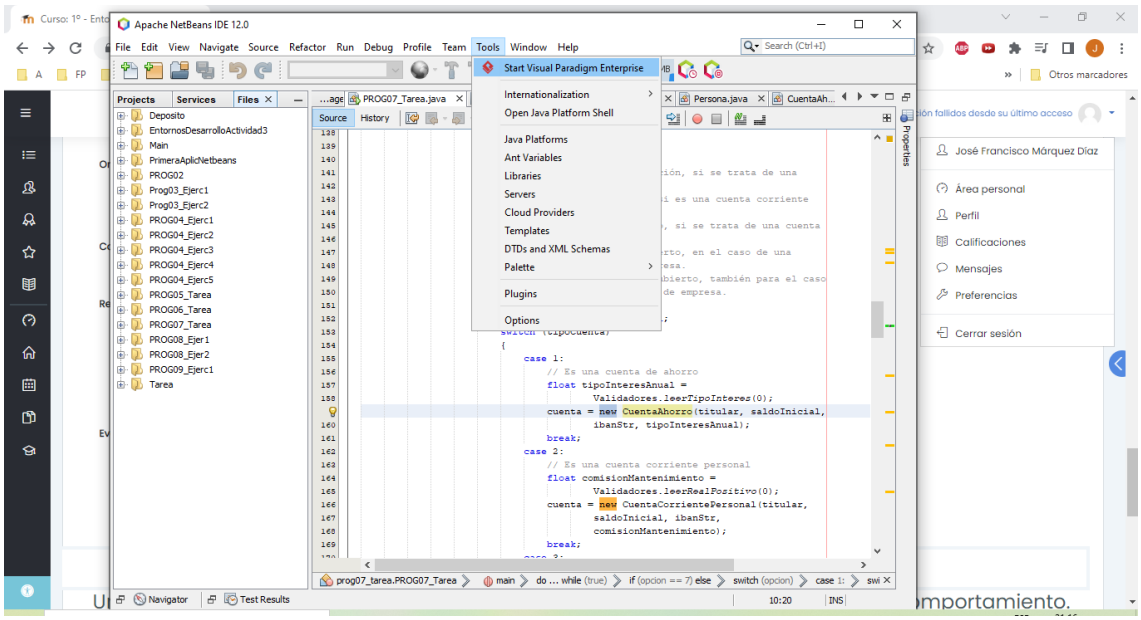
Comenzarán a copiarse los archivos:



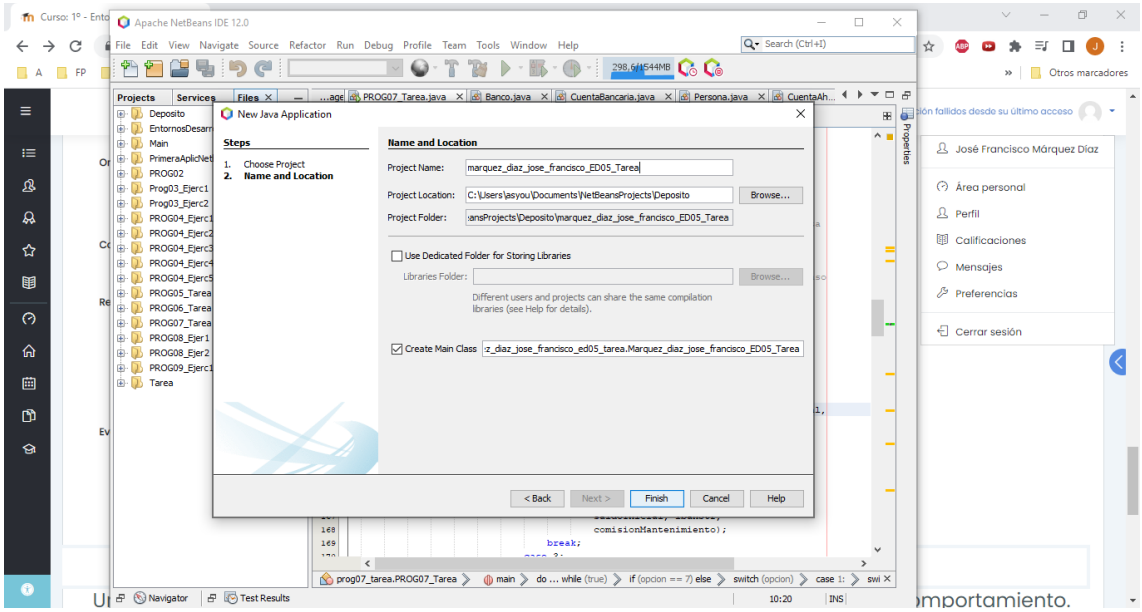
Y aparecerá un mensaje indicando que ya podemos usar Visual Paradigm en nuestro IDE.



Ahora abrimos NetBeans y vamos a la pestaña de *Tools* y seleccionamos *Start Visual Paradigm Enterprise.*



Creamos el proyecto marquez_diaz_jose_francisco_ED05_Tarea



Y con botón derecho sobre el proyecto marcamos seleccionamos *Import Visual Paradigm Project*