

ENTORNOS DE DESARROLLO

CASO PRACTICO IUD5



ALUMNO CESUR

24/25

Alejandro Muñoz de la Sierra

PROFESOR

Diego Tinedo Rodríguez

INTRODUCCION

El desarrollo de software no se limita a escribir código; planificar y diseñar desde el inicio marca la diferencia. Tener una visión clara de cómo se organizará el sistema es fundamental, y ahí es donde la Programación Orientada a Objetos (POO) se vuelve esencial, ya que nos permite dividir el sistema en bloques reutilizables y resistentes. Con este enfoque, cada componente se vincula de forma natural y se repite la idea de que organizar las partes facilita su mantenimiento.

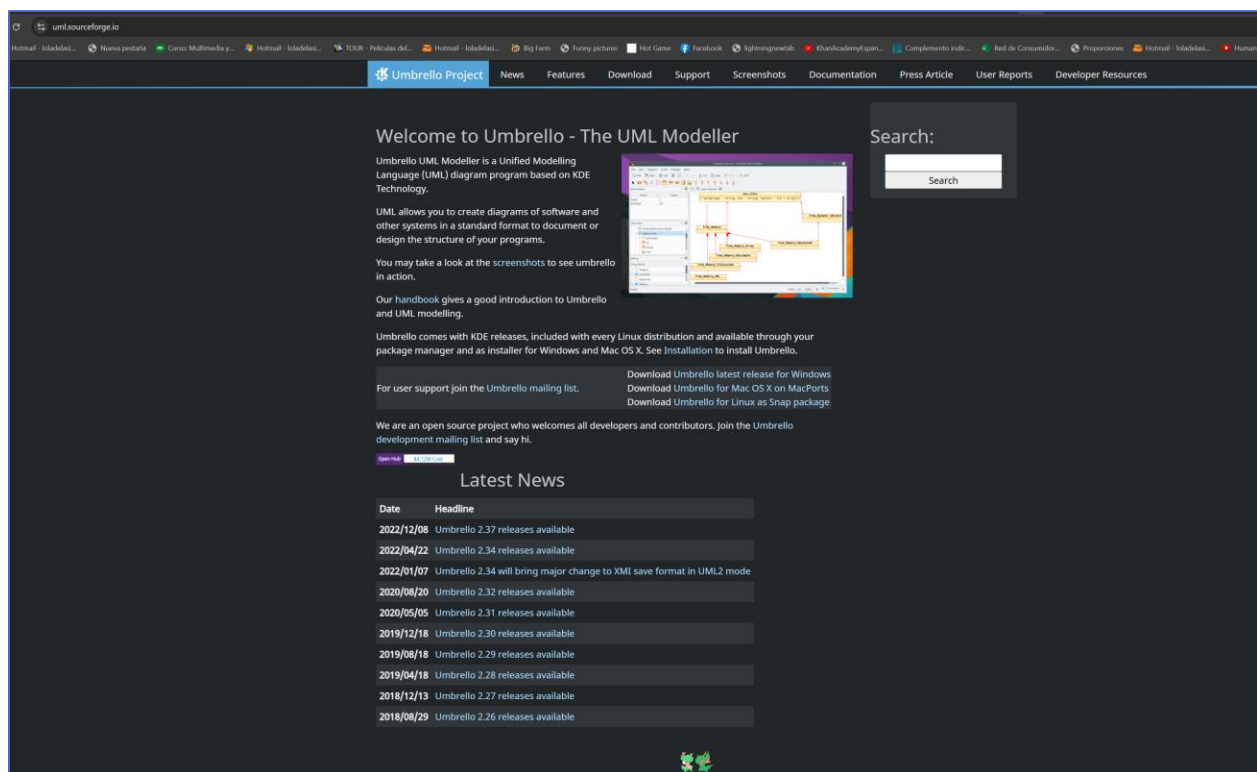
Una de las herramientas más utilizadas en este proceso es el diagrama de clases, que nos ayuda a “ver” de forma directa las entidades del sistema, sus atributos, sus métodos y, de modo un tanto inesperado, el modo en que se conectan entre sí. En este ejercicio, usaremos Umbrello, un software de modelado UML, que actuará como guía para desarrollar un diagrama basado en ciertas indicaciones. Data en mano, se busca reflejar la interrelación de cada parte sin caer en formalismos demasiado rígidos.

La tarea va más allá de simplemente recordar los fundamentos de la POO; se trata de mejorar la habilidad para diseñar e interpretar diagramas UML. En la mayoría de los casos se genera automáticamente código a partir del modelo, procurando que siga prácticas de programación sólidas. Además, se documenta todo el proceso –desde la instalación y configuración de Umbrello (con esa pequeña inconsistencia que a veces se nos escapa) hasta la creación del diagrama y una explicación de su estructura– complementándolo con capturas de pantalla y un informe completo.

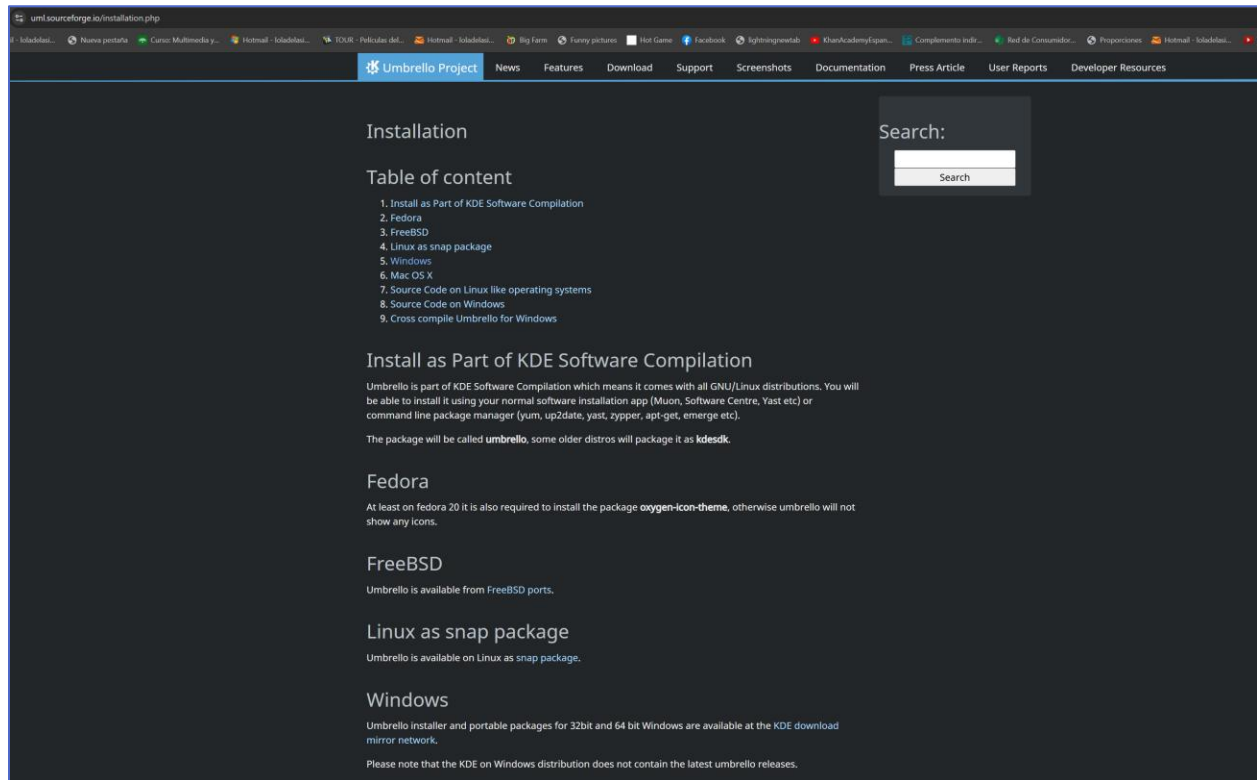
01

INSTALACIÓN Y CONFIGURACIÓN DE SOFTWARE

Accedemos a la pagina de Umbrello



Vamos a Downloads



The screenshot shows the Umbrello Project website. The navigation bar includes links for News, Features, Download, Support, Screenshots, Documentation, Press Article, User Reports, and Developer Resources. The main content area is titled "Installation" and contains a "Table of content" with links to various installation methods. A search bar is located in the top right corner.

Installation

Table of content

1. Install as Part of KDE Software Compilation
2. Fedora
3. FreeBSD
4. Linux as snap package
5. Windows
6. Mac OS X
7. Source Code on Linux like operating systems
8. Source Code on Windows
9. Cross compile Umbrello for Windows

Install as Part of KDE Software Compilation

Umbrello is part of KDE Software Compilation which means it comes with all GNU/Linux distributions. You will be able to install it using your normal software installation app (Muon, Software Centre, Yast etc) or command line package manager (yum, up2date, yast, zypper, apt-get, emerge etc).

The package will be called **umbrello**, some older distros will package it as **kdesdk**.

Fedora

At least on fedora 20 it is also required to install the package **oxygen-icon-theme**, otherwise umbrello will not show any icons.

FreeBSD

Umbrello is available from FreeBSD ports.

Linux as snap package

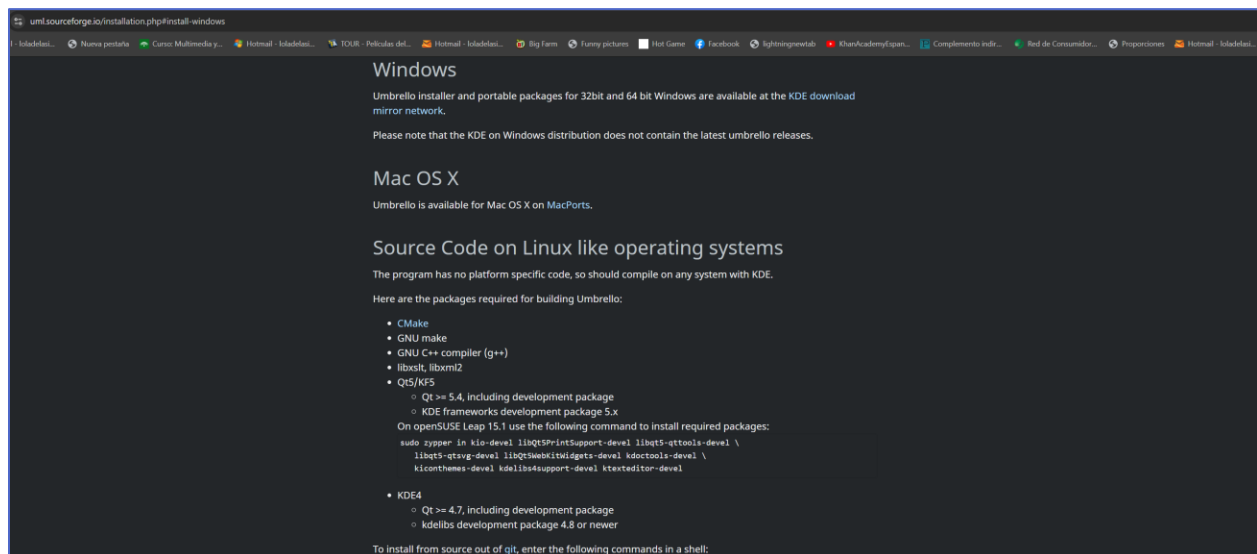
Umbrello is available on Linux as snap package.

Windows

Umbrello installer and portable packages for 32bit and 64 bit Windows are available at the KDE download mirror network.

Please note that the KDE on Windows distribution does not contain the latest umbrello releases.

Opcion 5 Windows



The screenshot shows the "Windows" section of the Umbrello Project website. It provides information about the availability of installers and portable packages, and includes a list of required packages for building Umbrello from source code.

Windows

Umbrello installer and portable packages for 32bit and 64 bit Windows are available at the KDE download mirror network.

Please note that the KDE on Windows distribution does not contain the latest umbrello releases.

Mac OS X

Umbrello is available for Mac OS X on MacPorts.

Source Code on Linux like operating systems

The program has no platform specific code, so should compile on any system with KDE.

Here are the packages required for building Umbrello:

- CMake
- GNU make
- GNU C++ compiler (g++)
- libx11, libxft2
- Qt5/KF5
 - Qt >= 5.4, including development package
 - KDE frameworks development package 5.x

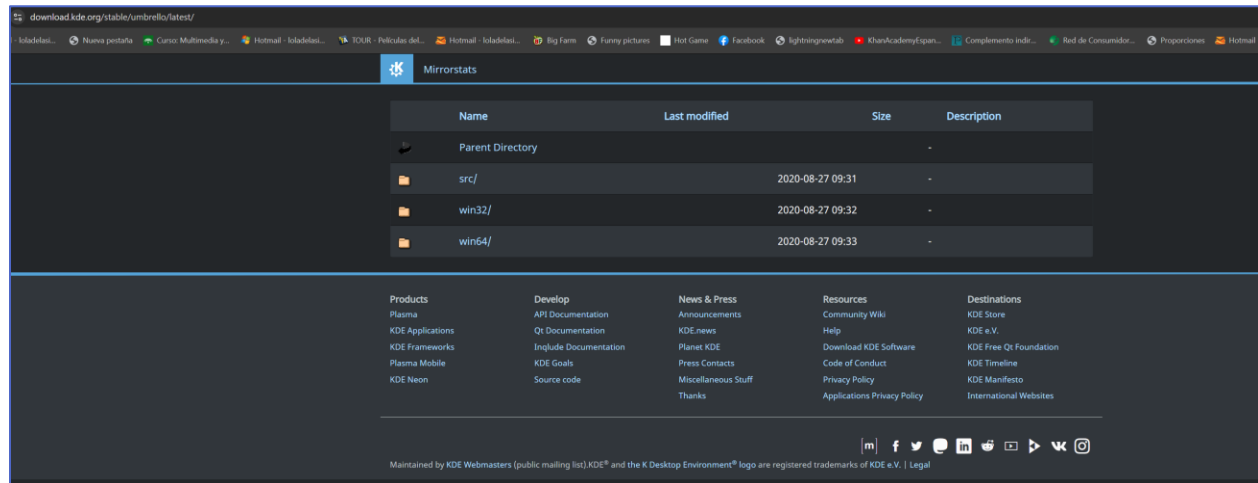
On openSUSE Leap 15.1 use the following command to install required packages:

```
sudo zypper in kio-devel libqt5PrintSupport-devel libqt5-qttools-devel \
libqt5-qtsvg-devel libqt5Webkitwidgets-devel kdoctools-devel \
kiconthemes-devel kdelibs4support-devel ktexteditor-devel
```

- KDE4
 - Qt >= 4.7, including development package
 - kdelibs development package 4.8 or newer

To install from source out of git, enter the following commands in a shell:

KDE Download mirror network

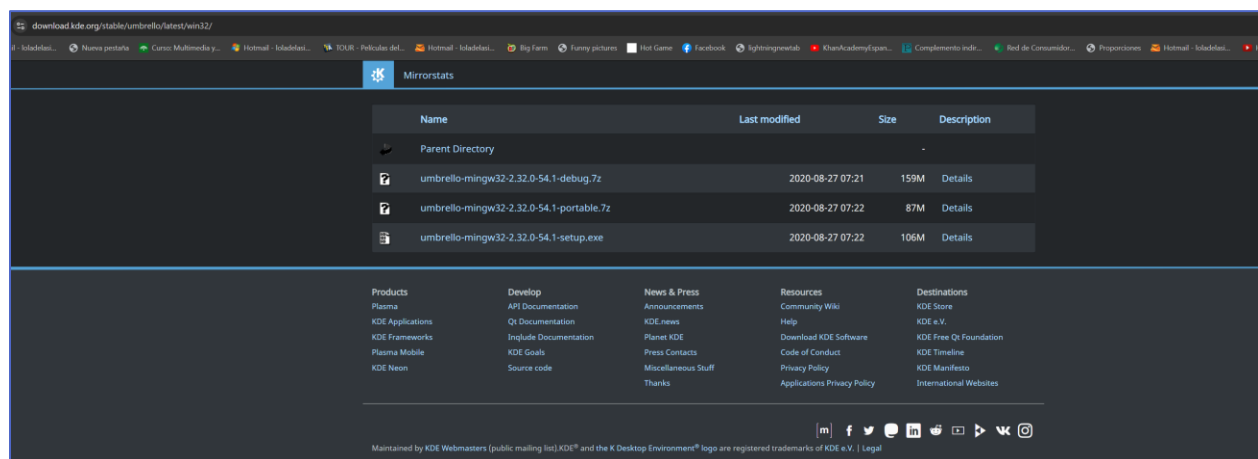


The screenshot shows the KDE Download mirror network website. The browser address bar displays 'download.kde.org/stable/umbrello/latest/'. The page title is 'Mirrorstats'. Below the title, there is a table with the following columns: Name, Last modified, Size, and Description. The table lists the following items:

Name	Last modified	Size	Description
Parent Directory			-
src/	2020-08-27 09:31	-	-
win32/	2020-08-27 09:32	-	-
win64/	2020-08-27 09:33	-	-

Below the table, there are five columns of links: Products, Develop, News & Press, Resources, and Destinations. The footer of the page states: 'Maintained by KDE Webmasters (public mailing list). KDE® and the K Desktop Environment® logo are registered trademarks of KDE e.V. | Legal'.

Carpeta Win32 (los compañeros han reportado que la versión de 64 bits da problemas)

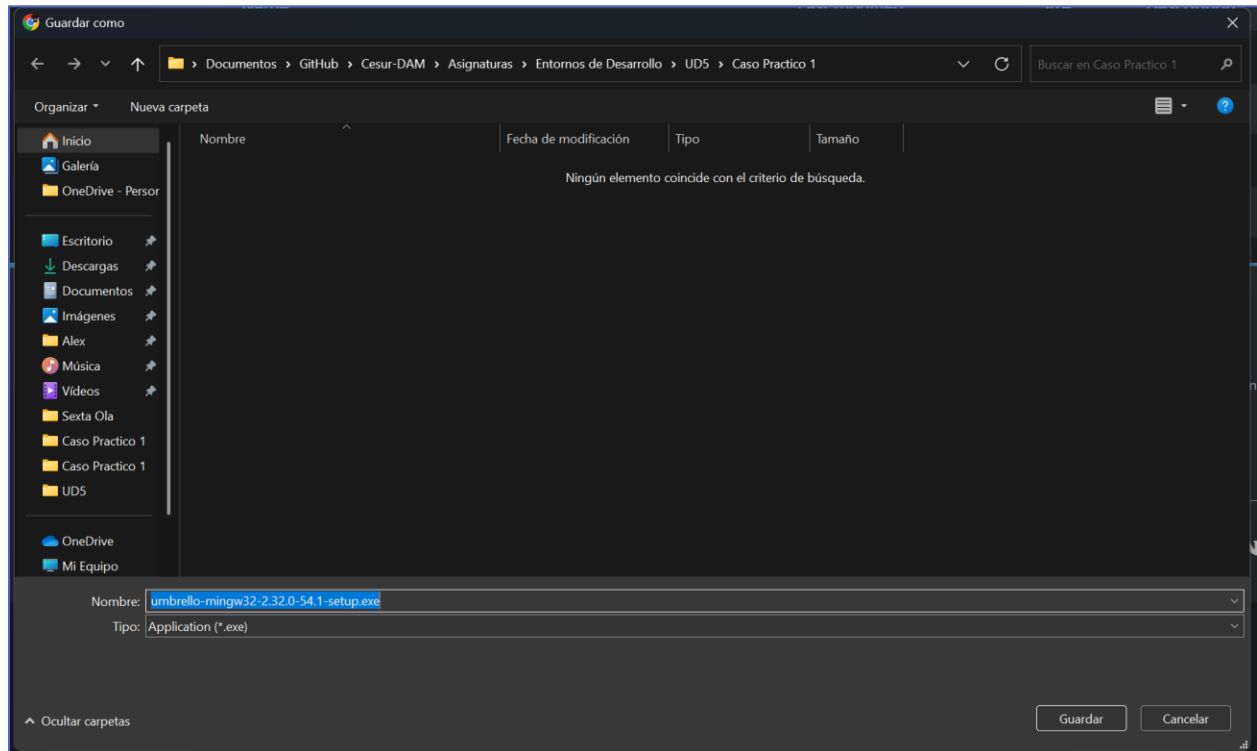


The screenshot shows the KDE Download mirror network website. The browser address bar displays 'download.kde.org/stable/umbrello/latest/win32/'. The page title is 'Mirrorstats'. Below the title, there is a table with the following columns: Name, Last modified, Size, and Description. The table lists the following items:

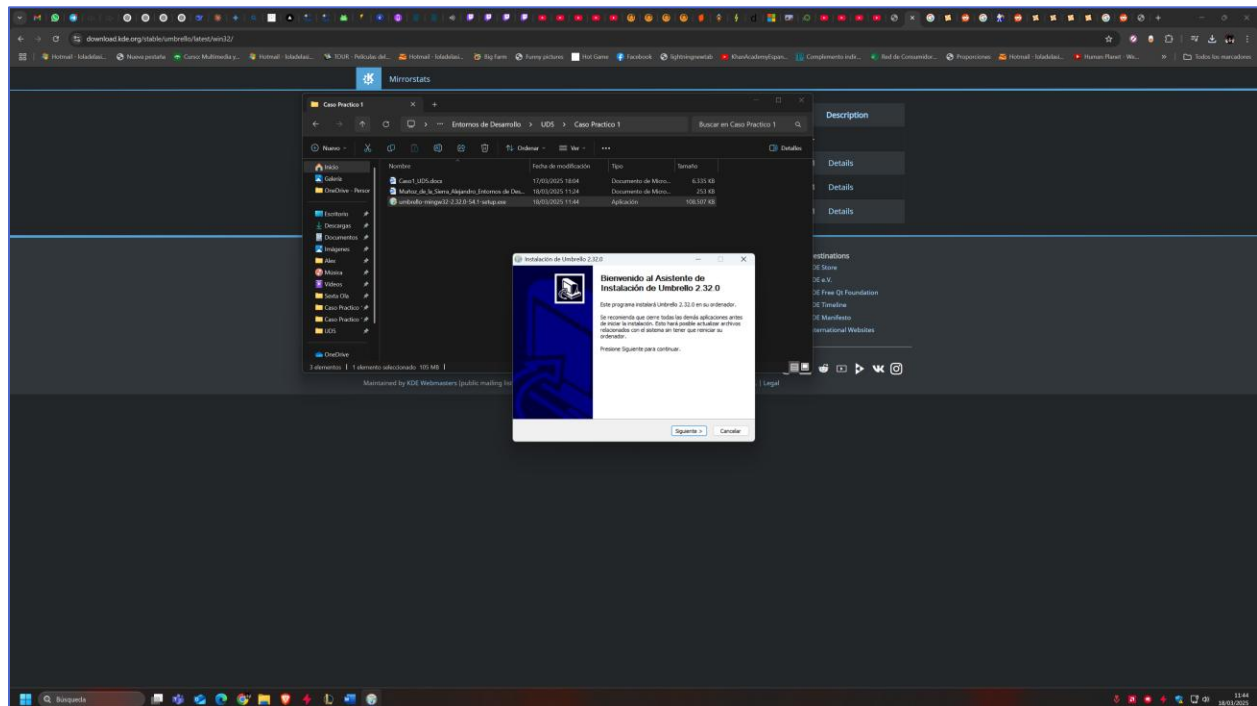
Name	Last modified	Size	Description
Parent Directory			-
umbrello-mingw32-2.32.0-54.1-debug.7z	2020-08-27 07:21	159M	Details
umbrello-mingw32-2.32.0-54.1-portable.7z	2020-08-27 07:22	87M	Details
umbrello-mingw32-2.32.0-54.1-setup.exe	2020-08-27 07:22	106M	Details

Below the table, there are five columns of links: Products, Develop, News & Press, Resources, and Destinations. The footer of the page states: 'Maintained by KDE Webmasters (public mailing list). KDE® and the K Desktop Environment® logo are registered trademarks of KDE e.V. | Legal'.

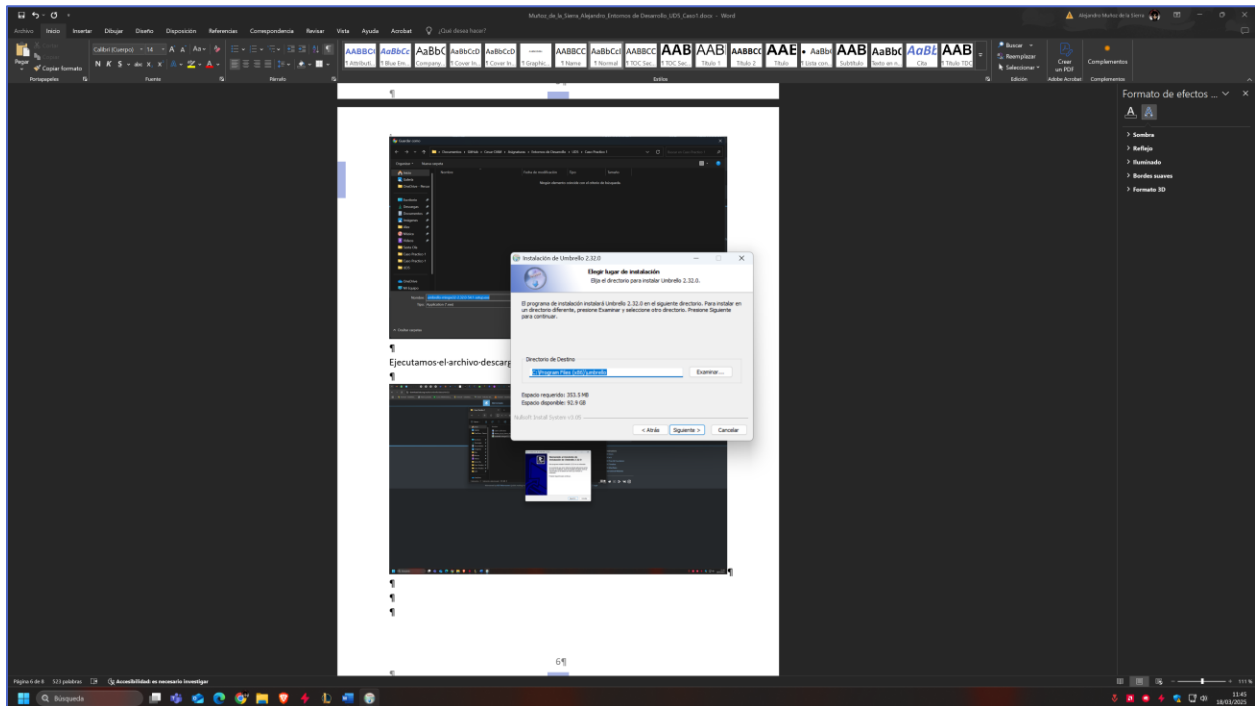
Seleccionamos `umbrello-mingw32-2.32.0-54.1-setup.exe` , lo descargamos



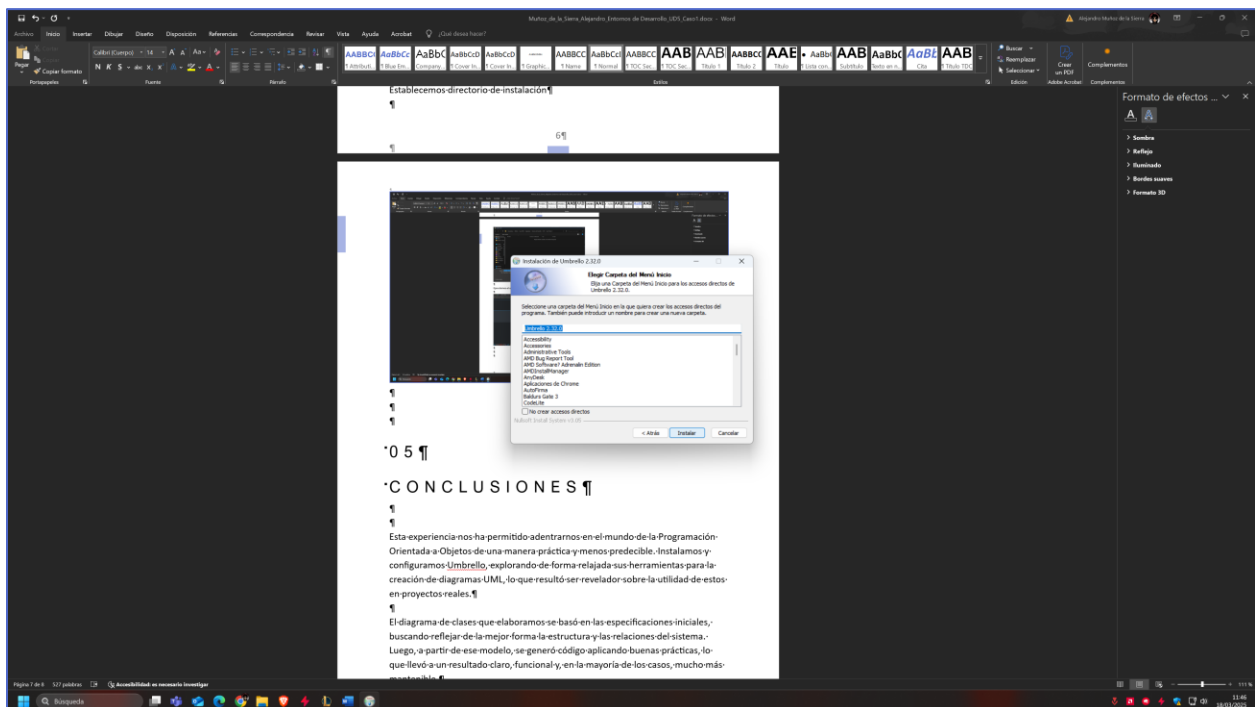
Ejecutamos el archivo descargado



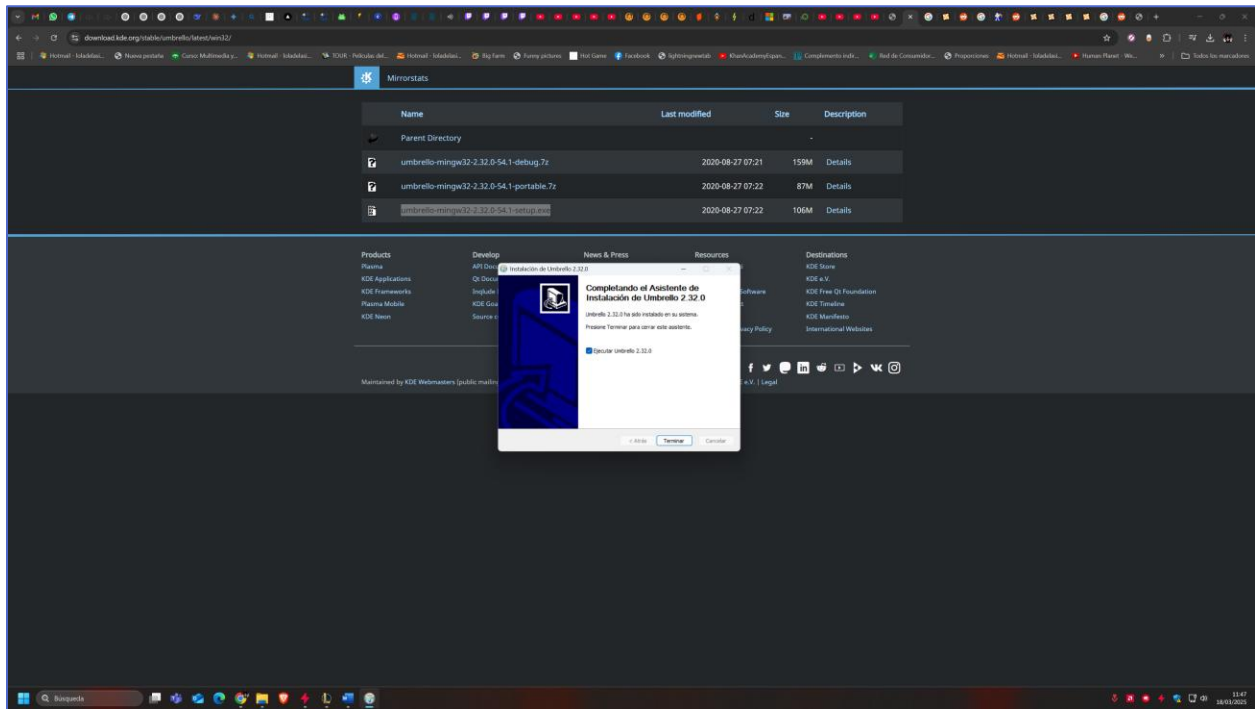
Establecemos directorio de instalación



Carpeta de inicio



Finalizamos instalación

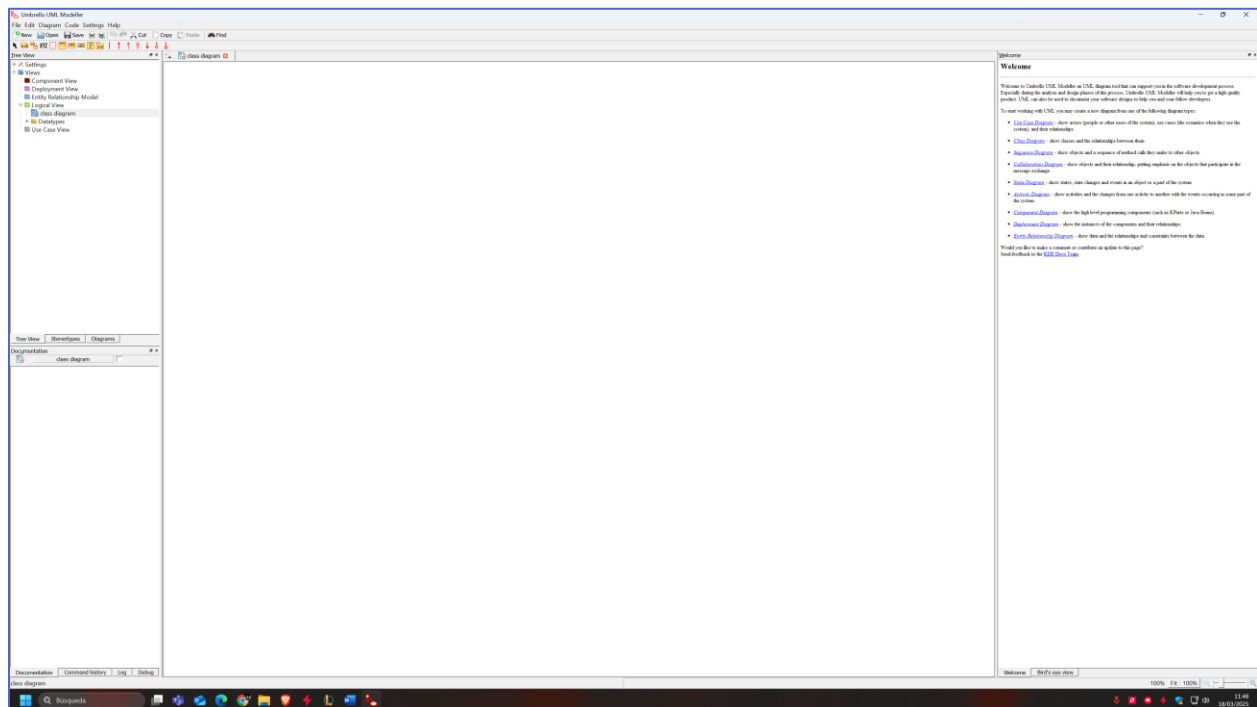


02

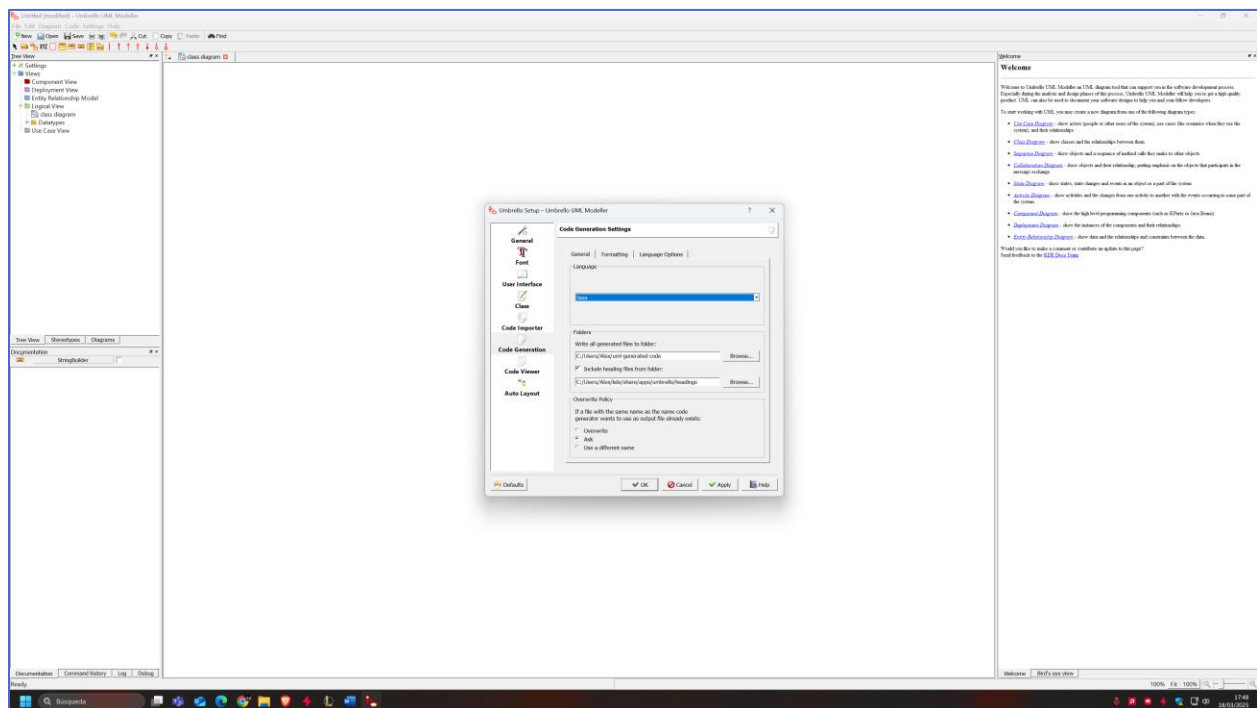
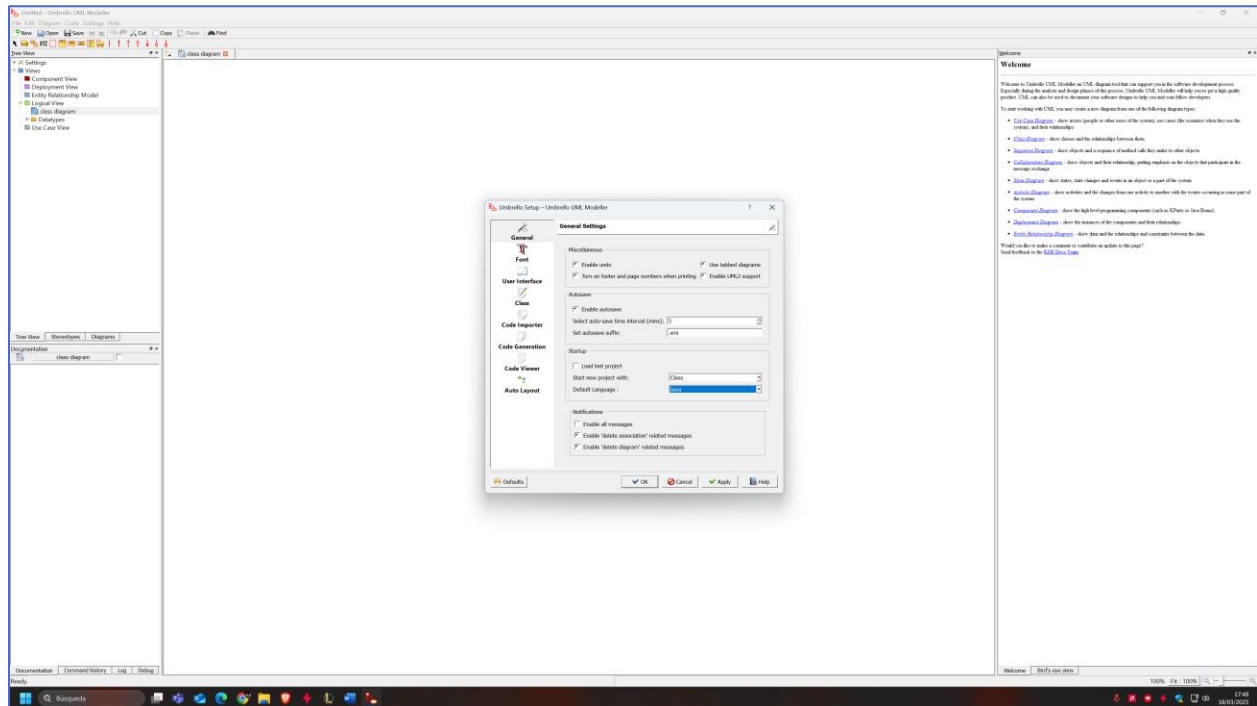
GENERACIÓN E INTERPRETACIÓN DE DIAGRAMAS DE CLASES CON UMBRELLA

1. Creamos un nuevo proyecto

Antes de meter clases y relaciones, nos tocó iniciar un proyecto nuevo. Abrimos Umbrello, esperamos a que se cargue todo (un par de segundos, en la mayoría de los casos), y luego fuimos a Archivo → Nuevo, o simplemente pulsamos Ctrl+N. Así, de repente, nos encontramos con un lienzo en blanco, perfecto para comenzar.



Configuramos el programa para que el lenguaje por defecto sea Java, tanto en la creación de clases como en la generación de código.



2. Añadimos clases al diagrama

Ya con el lienzo listo, decidimos poner las entidades del sistema:

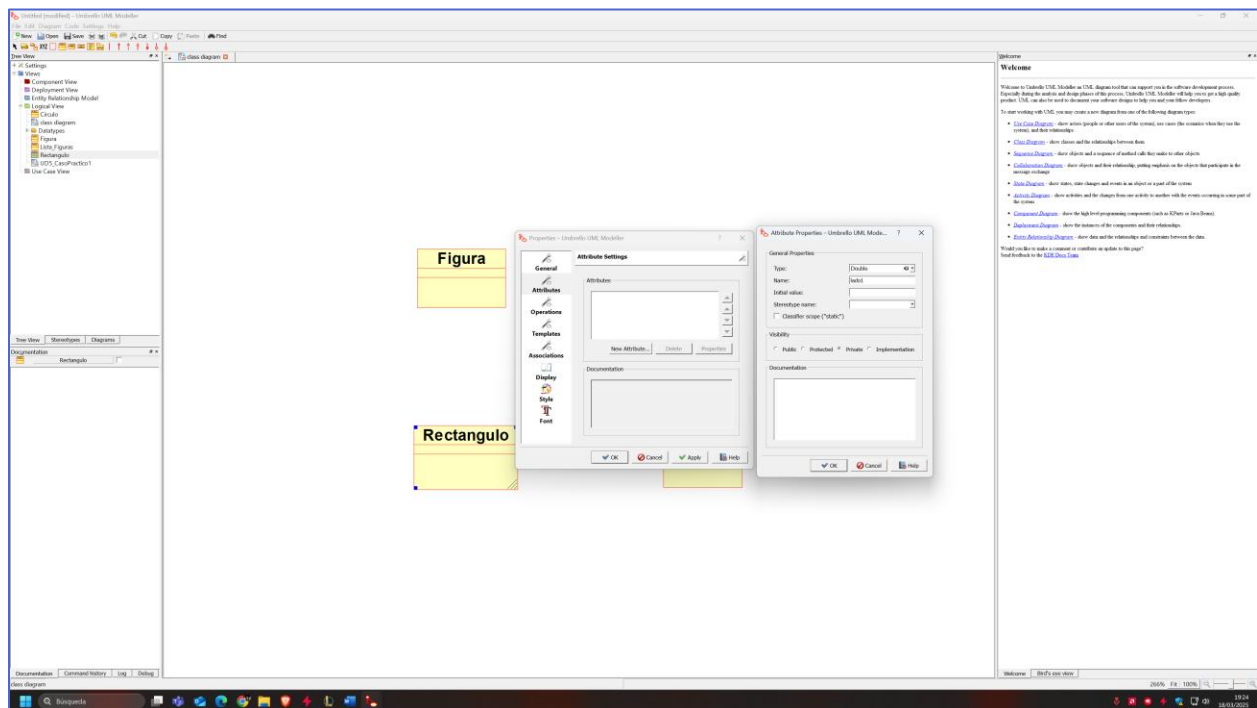
- En la barra lateral izquierda, nos aseguramos de estar en la pestaña que muestra la parte estructural – esa parte es importante.
- Luego botón derecho dentro de Logical View elegimos new y Clase, o con un clic en el icono arriba en la barra de trabajo, añadimos las clases de la guía.
- Para renombrarla, bastó con hacer doble clic sobre ella y escribir el nuevo nombre; así de sencillo.



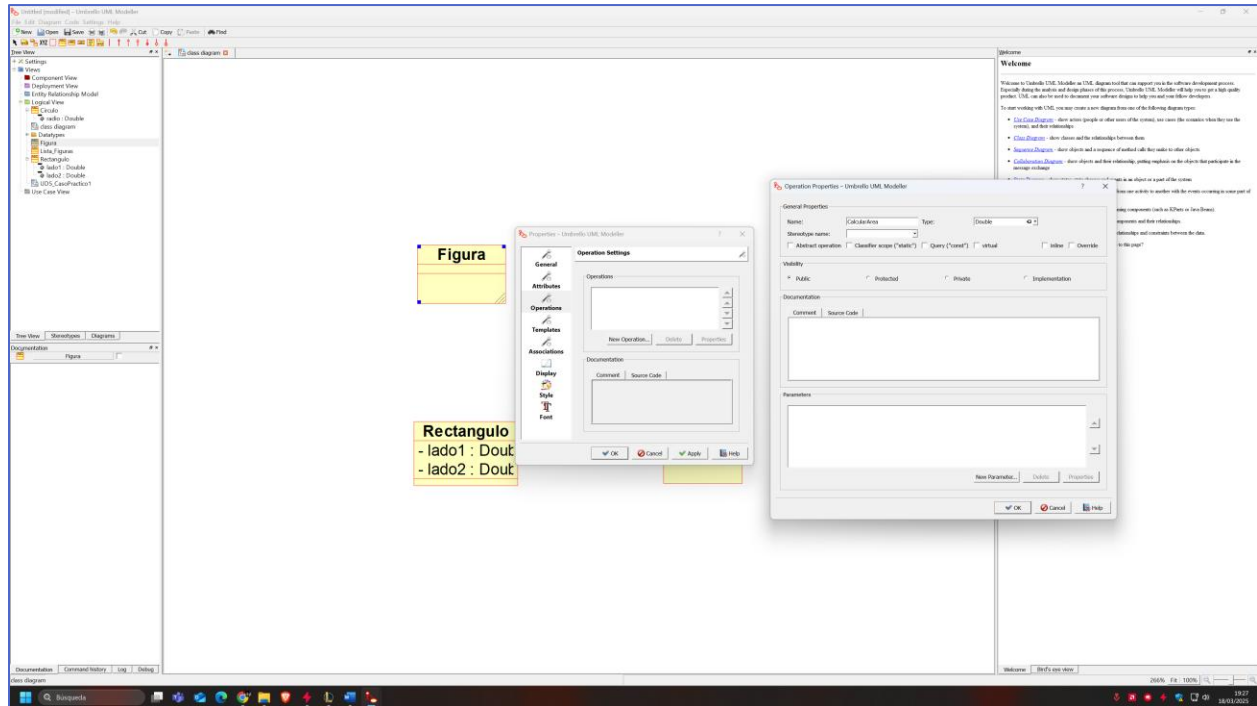
3. Agregamos atributos y métodos

Cada clase debía llevar sus detalles: variables y funciones. Aquí hicimos lo siguiente:

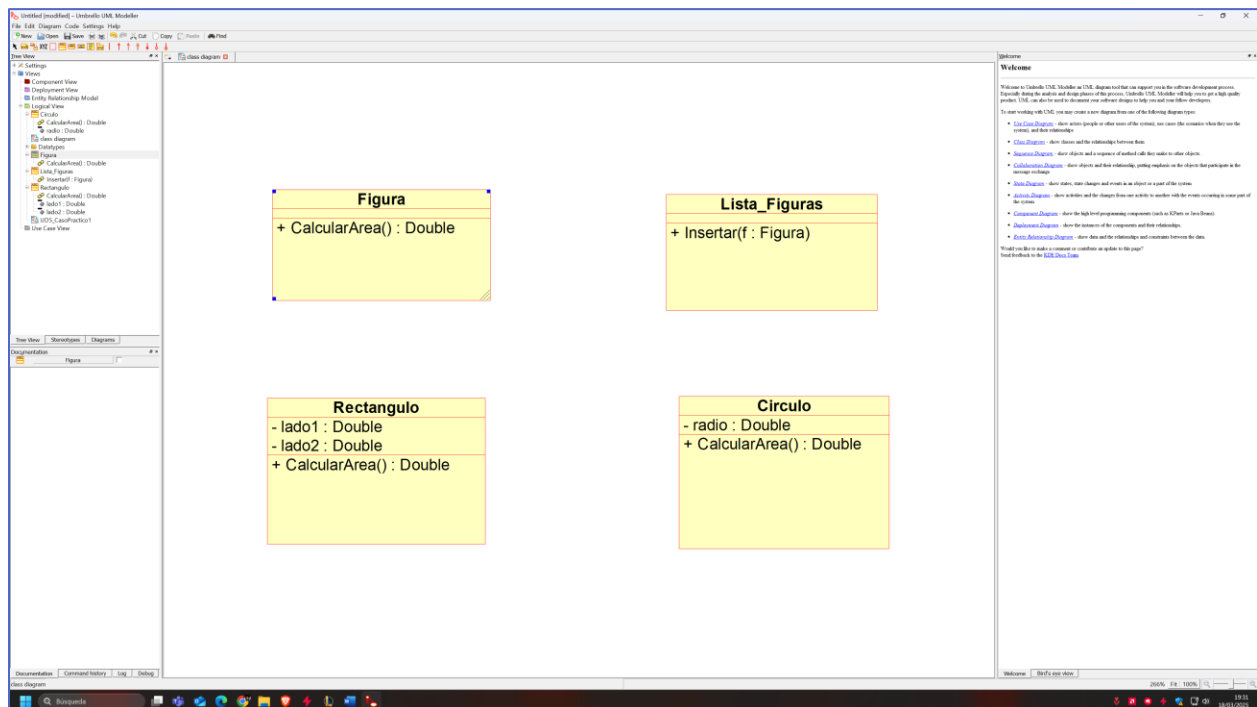
- Accedimos a las propiedades de la clase haciendo doble clic o con el menú del clic derecho → Propiedades.
- Para los atributos, fuimos a la pestaña correspondiente y pulsamos Añadir; escribimos el nombre, elegimos el tipo de dato y configuramos la visibilidad (ya sea public, private, o protected).



- Después, para los métodos, entramos en la pestaña de Operaciones y de nuevo hicimos clic en Añadir, definiendo el nombre, el tipo de retorno e incluso los parámetros si hacía falta.



Resultado del diagrama completo a falta de las relaciones



4. Creamos relaciones entre clases

El siguiente paso fue vincular nuestras clases; sin eso, el diagrama no tendría sentido.

- En la barra superior, seleccionamos el tipo de conexión que necesitábamos: puede ser una generalización (lo que a veces se llama herencia), asociación, composición o agregación.
- Hicimos clic en la clase de origen, arrastramos el cursor hasta la clase de destino y soltamos – y al instante, se establecía la relación.

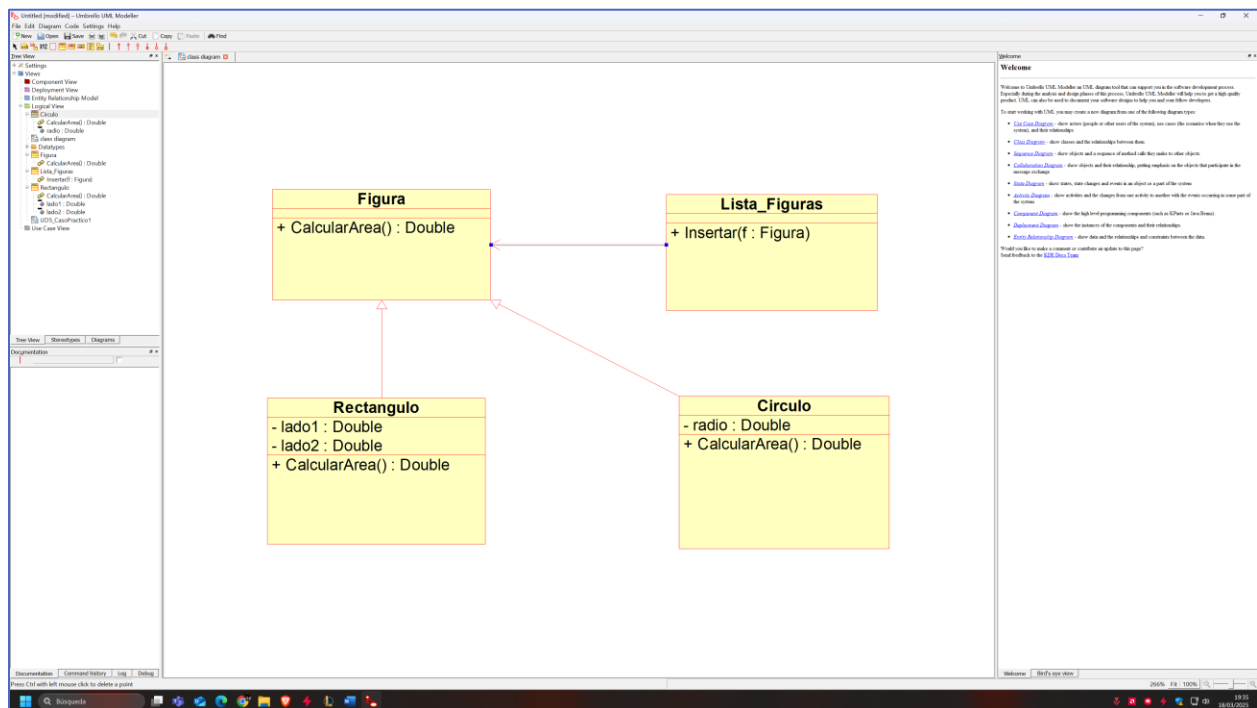


Figura es la Clase Padre, y de ahí tienen relación de herencia las clases Círculo y Rectángulo, ya que “son” figuras y por lógica tendrán todas las propiedades y métodos de la clase padre. Estas relaciones se reflejan con una línea que va de las clases hijas a la clase padre con una flecha cerrada en blanco.

La otra relación es de asociación entre la clase Lista de Figuras y Figura, aunque realmente debería ser una agregación porque Figura puede existir sin Lista de Figuras pero no al contrario, pero seguimos el criterio de la guía. Se representa por una línea de la clase hija a la clase padre con una flecha abierta al final.

La cardinalidad lógica es como queda reflejada en el diagrama. Sólo hay una lista, pero esta puede tener muchas figuras.

- Posteriormente, ajustamos detalles como el nombre de la relación y su multiplicidad, personalizándola según nuestros requerimientos.

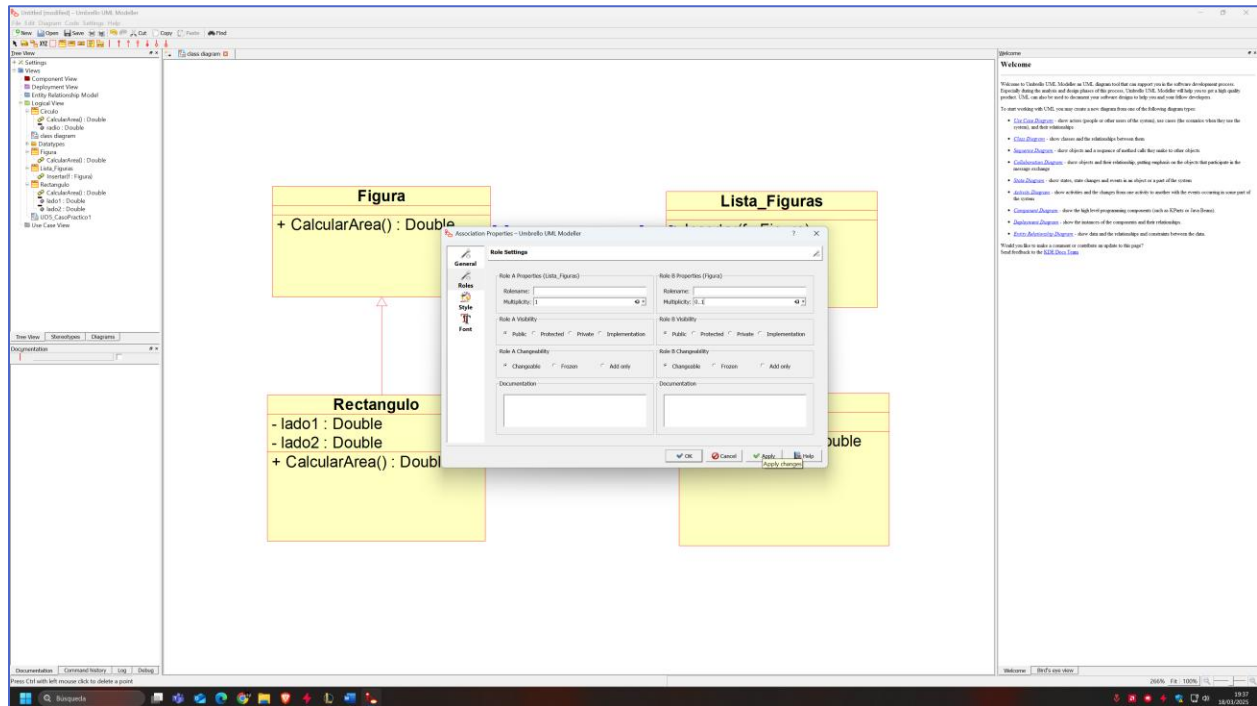
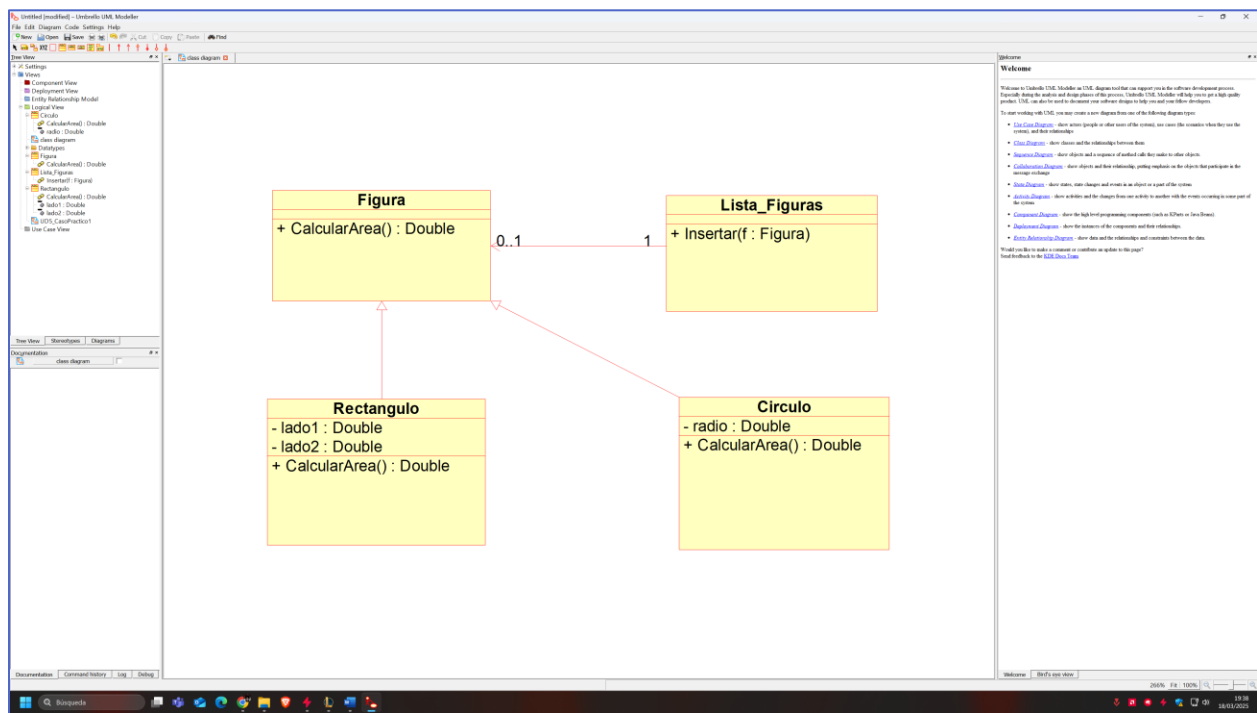


Diagrama finalizado con las relaciones

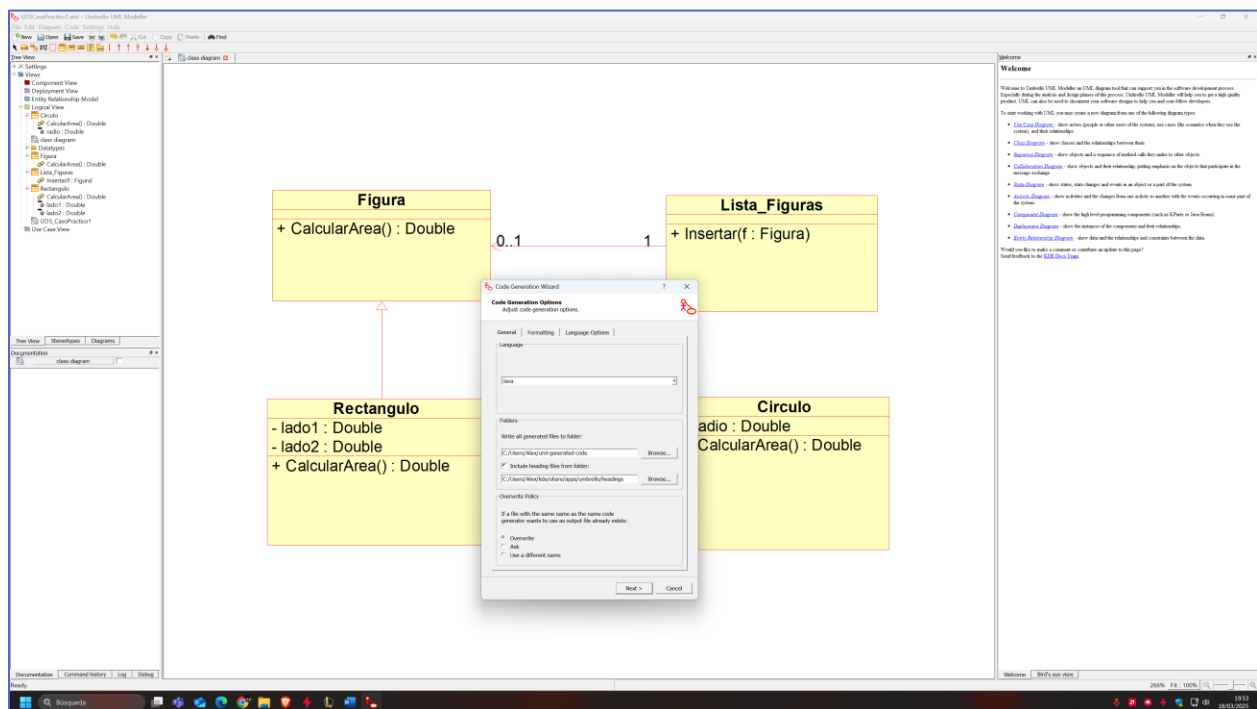


5. Para almacenar el proyecto, fuimos a Archivo → Guardar como... y elegimos el formato .xmi, que es adecuado para este tipo de diagramas.

03

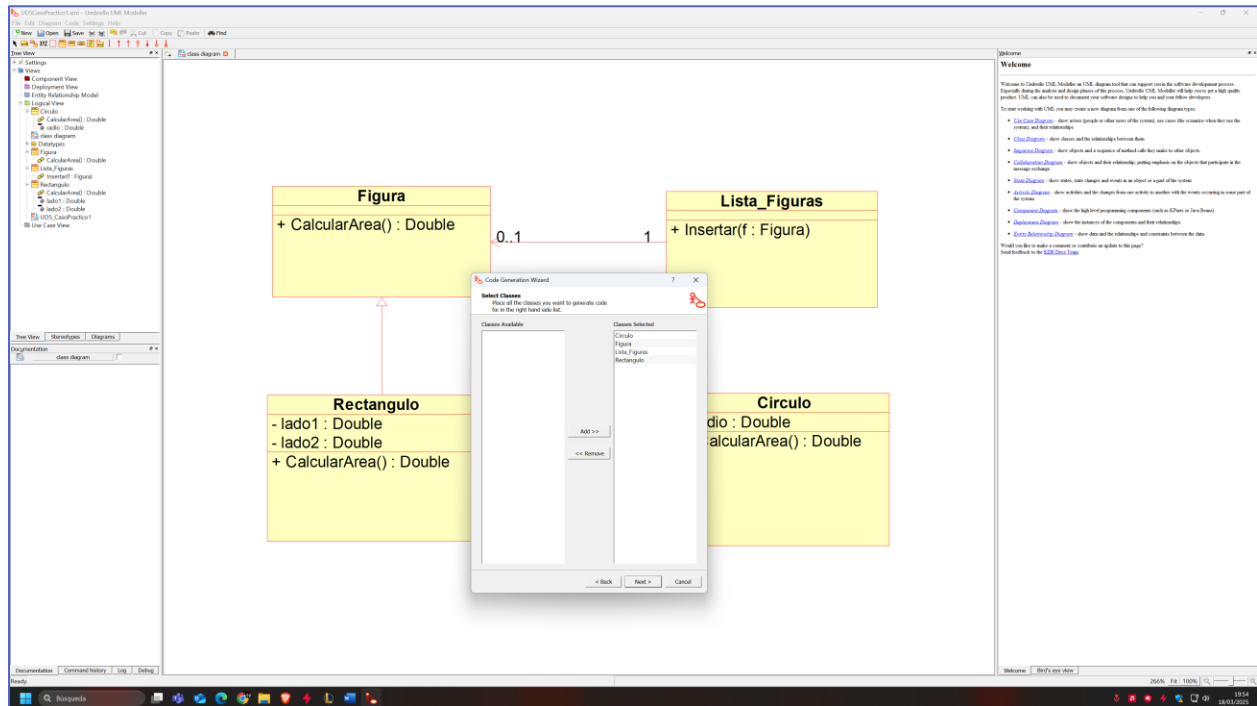
GENERACIÓN DE CÓDIGO DESDE EL DIAGRAMA DE CLASES

Para ello vamos a la barra superior, a la pestaña code y dentro de ahí la opción code generation wizard.

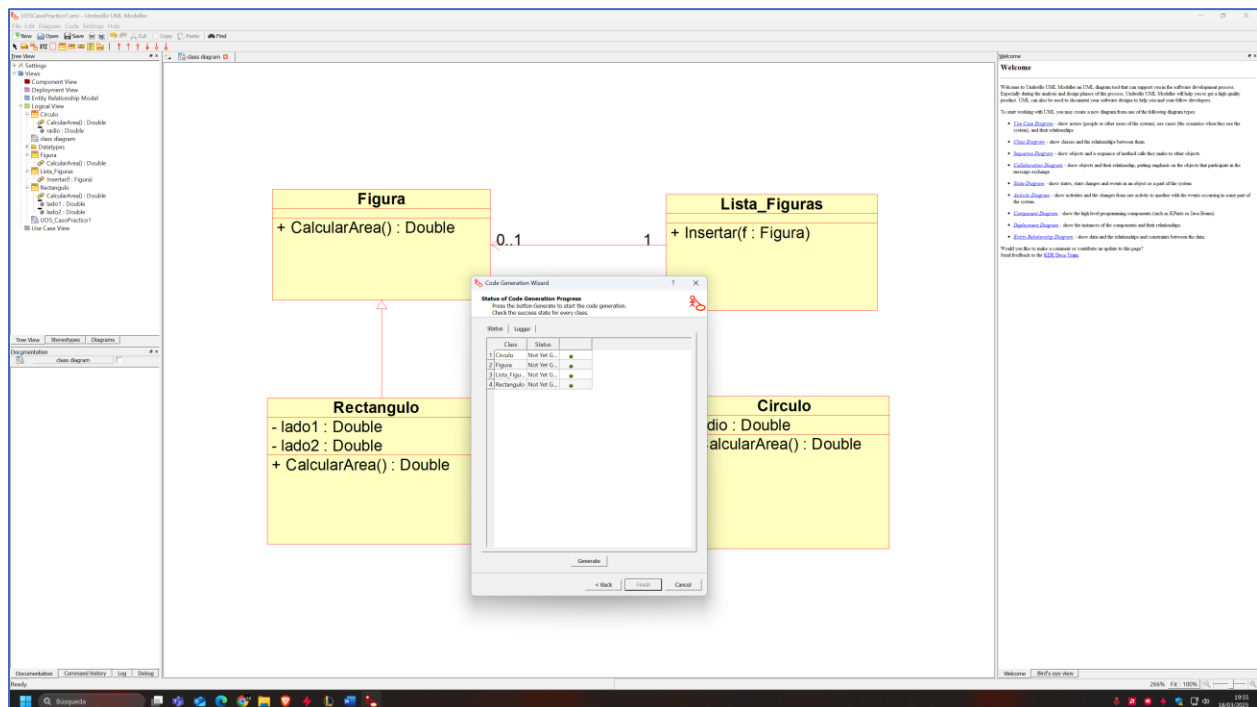


Comprobamos que el lenguaje elegido es el correcto y el directorio destino de los archivos.

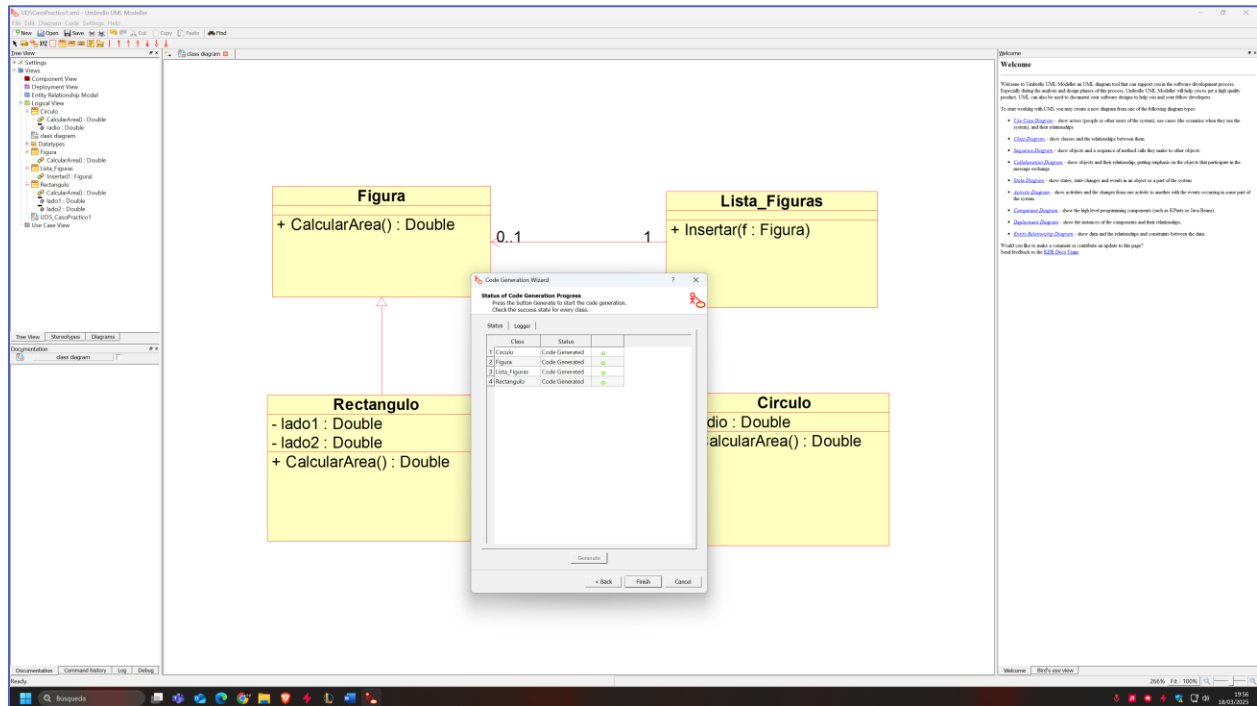
Seleccionamos las clases de las que queremos generar su código.



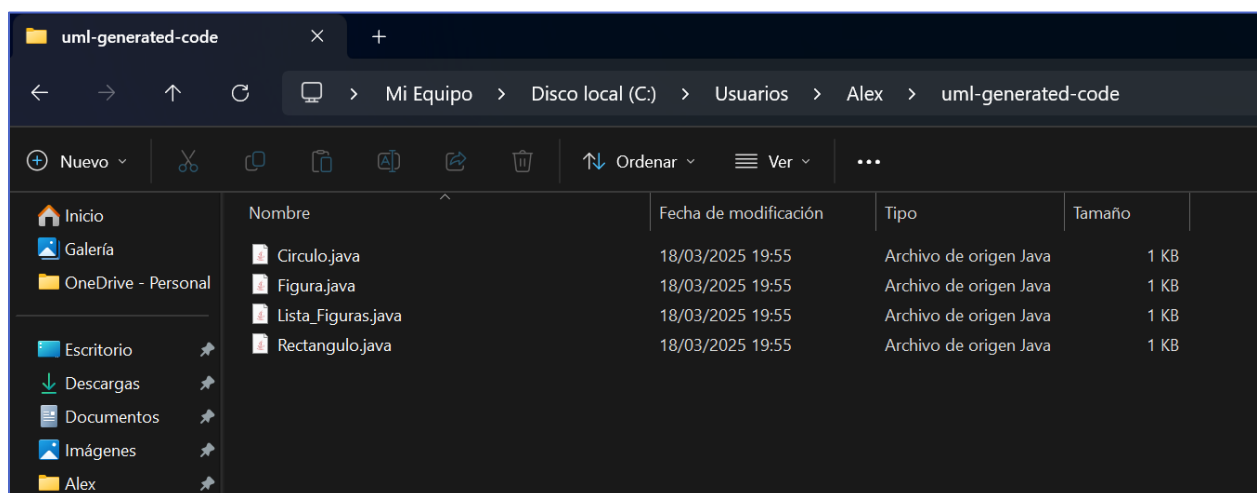
Vemos que están las clases correctas y pulsamos Generate



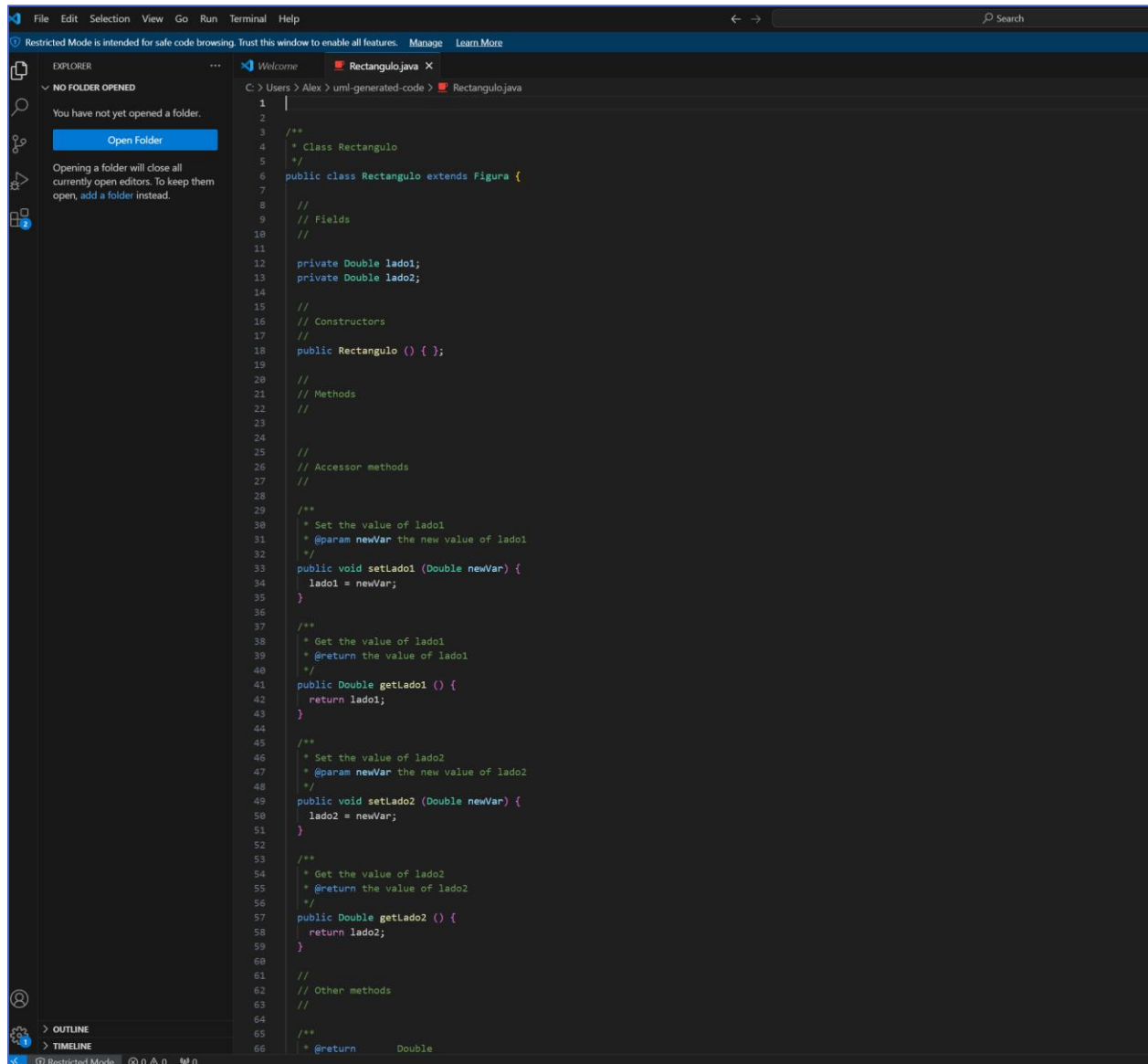
Vemos que no hay error generando el código.



Pulsamos Finish y vamos al directorio destino a comprobar que existen los archivos.



Comprobamos si los archivos tienen el código correcto.



The screenshot shows an IDE window with the file `Rectangulo.java` open. The Explorer panel on the left indicates 'NO FOLDER OPENED'. The main editor displays the following Java code:

```
1
2
3  /**
4   * Class Rectangulo
5   */
6  public class Rectangulo extends Figura {
7
8      //
9      // Fields
10     //
11
12     private Double lado1;
13     private Double lado2;
14
15     //
16     // Constructors
17     //
18     public Rectangulo () { };
19
20     //
21     // Methods
22     //
23
24     //
25     // Accessor methods
26     //
27     //
28
29     /**
30      * Set the value of lado1
31      * @param newVar the new value of lado1
32      */
33     public void setLado1 (Double newVar) {
34         lado1 = newVar;
35     }
36
37     /**
38      * Get the value of lado1
39      * @return the value of lado1
40      */
41     public Double getLado1 () {
42         return lado1;
43     }
44
45     /**
46      * Set the value of lado2
47      * @param newVar the new value of lado2
48      */
49     public void setLado2 (Double newVar) {
50         lado2 = newVar;
51     }
52
53     /**
54      * Get the value of lado2
55      * @return the value of lado2
56      */
57     public Double getLado2 () {
58         return lado2;
59     }
60
61     //
62     // Other methods
63     //
64
65     /**
66      * @return Double
```

Vemos que el código se corresponde con el diagrama creado.

CONCLUSIONES

Esta experiencia nos ha permitido adentrarnos en el mundo de la Programación Orientada a Objetos de una manera práctica y menos predecible. Instalamos y configuramos Umbrello, explorando de forma relajada sus herramientas para la creación de diagramas UML, lo que resultó ser revelador sobre la utilidad de estos en proyectos reales.

El diagrama de clases que elaboramos se basó en las especificaciones iniciales, buscando reflejar de la mejor forma la estructura y las relaciones del sistema. Luego, a partir de ese modelo, se generó código aplicando buenas prácticas, lo que llevó a un resultado claro, funcional y, en la mayoría de los casos, mucho más mantenible.

En definitiva, la experiencia demostró que los diagramas de clases no solo facilitan la comprensión del sistema, sino que también pueden agilizar el desarrollo, disminuir la cantidad de errores y organizar el código desde el comienzo. Gracias a este método, los desarrolladores pueden visualizar el conjunto del sistema antes de poner en marcha la codificación, lo que ayuda a prever problemas y a tomar decisiones más acertadas. Así, dominar herramientas como Umbrello y entender el valor del buen diseño previo se consolidan como habilidades esenciales para crear aplicaciones eficientes, bien estructuradas y capaces de escalar en el futuro.

REFERENCIAS

<https://www.youtube.com/watch?v=EPxX94RcjeA>

<https://docs.kde.org/trunk5/es/umbrello/umbrello/uml-elements.html>

<https://docs.kde.org/trunk5/es/umbrello/umbrello/>

<https://es.wikipedia.org/wiki/Umbrello>

<https://docs.kde.org/trunk5/es/umbrello/umbrello/code-import-generation.html>