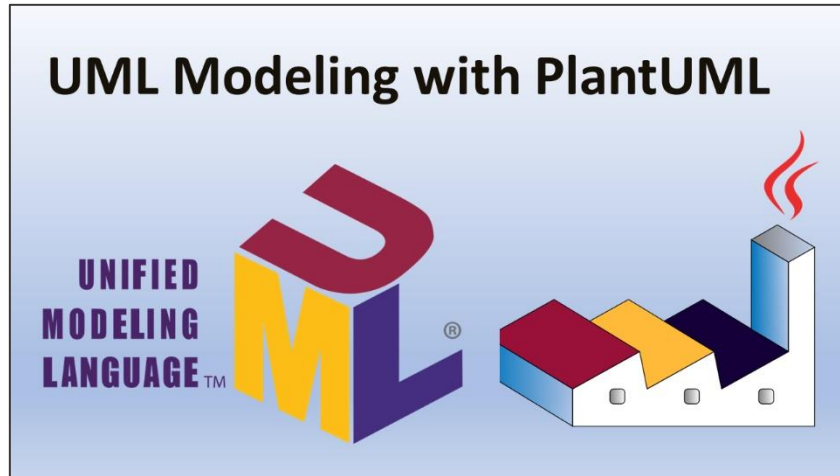


PRÁCTICA 2. MODELADO DE CLASES CON PLANTUML



<https://plantuml.com/es/>

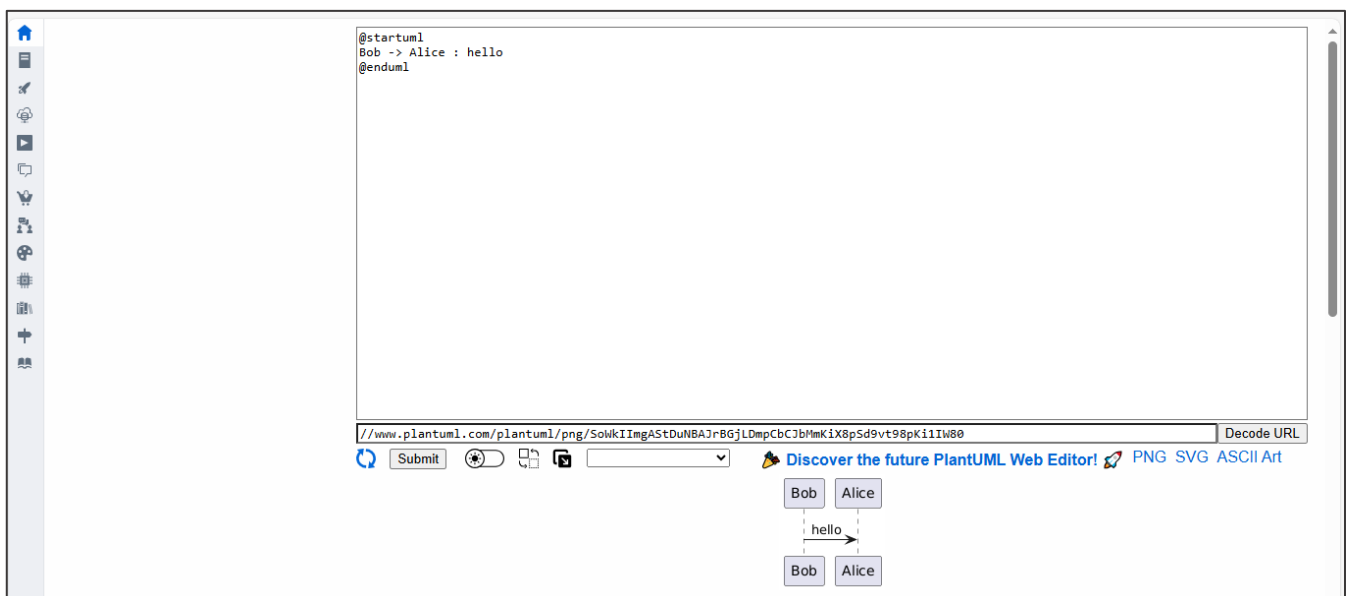
→ INTRODUCCIÓN: PlantUML de un vistazo

PlantUML es una herramienta gratuita muy versátil que facilita la creación rápida de una amplia gama de tipos de diagramas. Se encuentra en código abierto en el siguiente repositorio de *GitHub*: [plantuml/plantuml: Generate diagrams from textual description](https://github.com/plantuml/plantuml)

Esta herramienta utiliza un lenguaje sencillo que los usuarios pueden redactar sin demasiado esfuerzo. Para ver info detallada sobre las capacidades de este lenguaje y su sintaxis, se puede consultar la [Guía de Referencia del Lenguaje PlantUML](#), **pero tranquil@, que no lo escribiremos nosotros**.

→ INSTALACIÓN

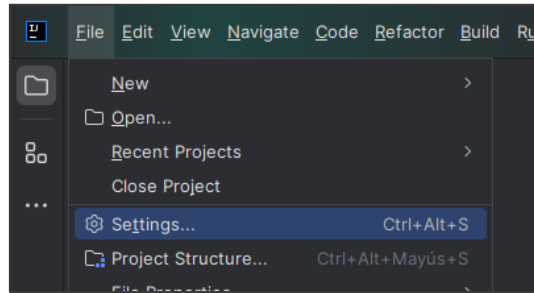
Hay varias formas de consumir esta plataforma, ya que permite la instalación local, el uso de su [servidor online](#) y la instalación de plugins en los IDE más importantes.



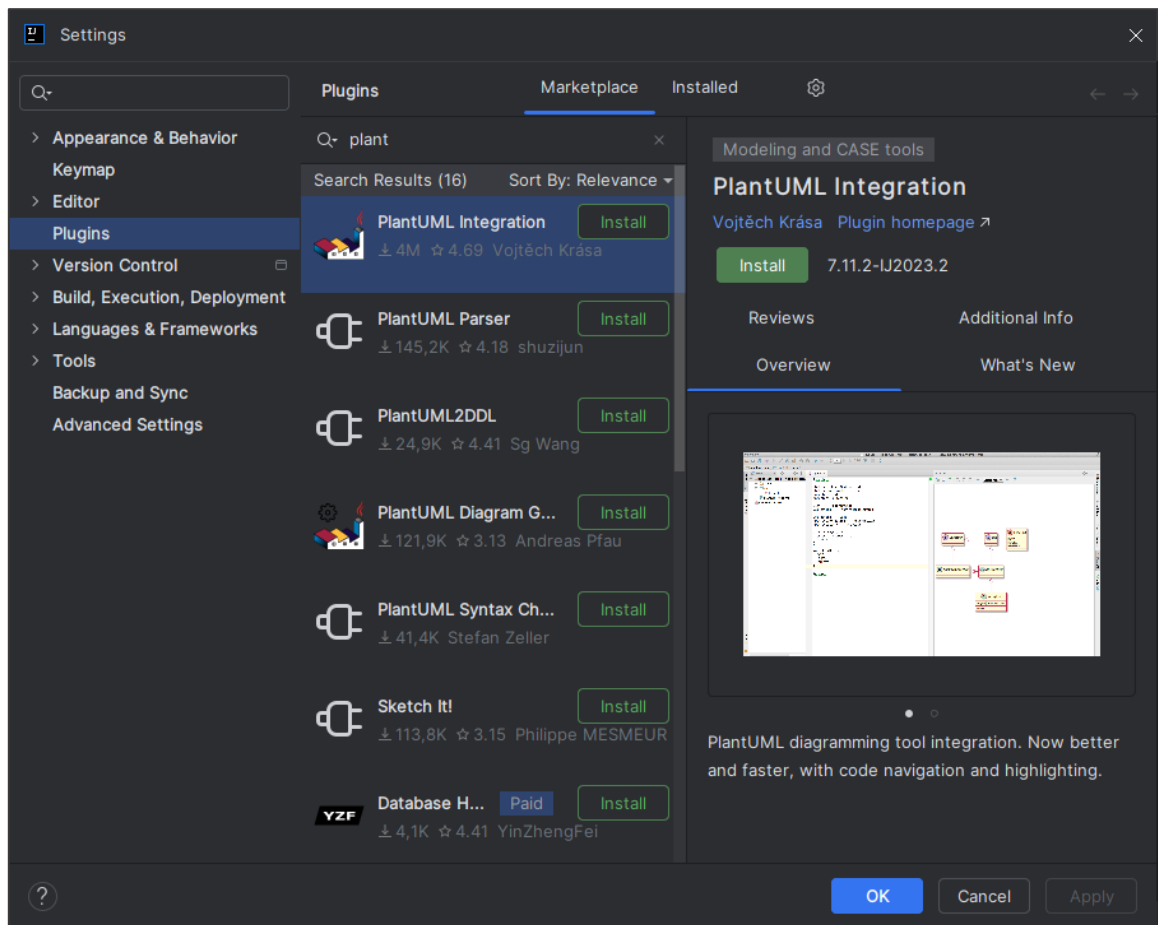
En nuestro caso, dado que las clases a modelar estarán en un paquete de proyecto *Maven* de *IntelliJ IDEA*, sería conveniente instalar el plugin correspondiente.

1. Instalación del plugin *PlantUML Integration*

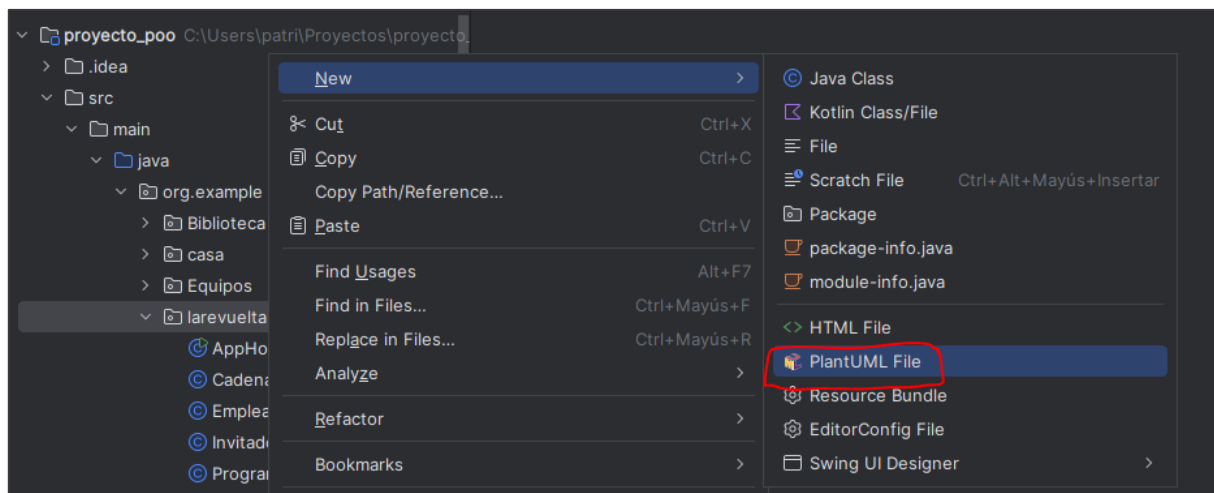
- Ve a tu *IntelliJ IDEA* y accede a **File** → **Settings** del menú superior:



- Busca la pestaña **Plugins** y escribe en el buscador el que necesitamos:



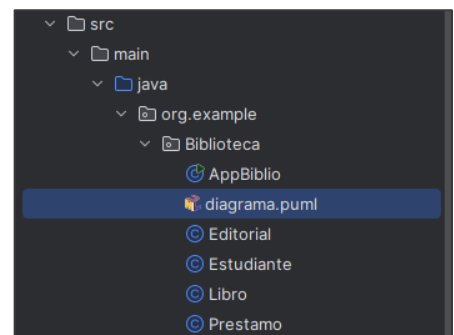
- Pulsa el botón ***Install*** y **reinicia el IDE**.
- Una vez hecho, si nos posicionamos sobre cualquier elemento del proyecto que tengamos abierto y hacemos *click* derecho para crear un nuevo componente, nos debe aparecer ***PlantUML File*** como opción:



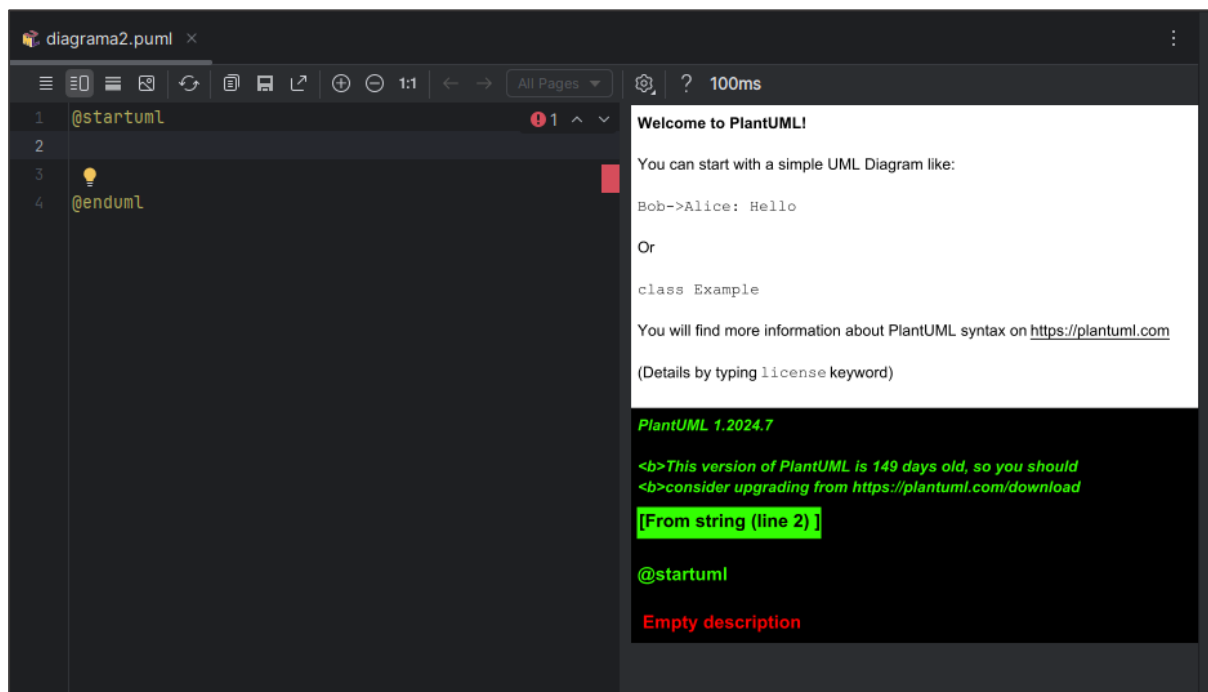
2. Creación de archivos *.puml* y generación de diagramas de clases

Siguiendo por donde nos hemos quedado en el apartado anterior, dirígete a alguno de los paquetes donde tengas los ejercicios sobre POO realizados durante las últimas semanas. Para este ejemplo, cogeremos el de la Biblioteca, que posee relaciones de varios tipos (asociación/agregación y composición).

- Creamos un archivo *.puml* en el paquete que contiene las clases relacionadas con nuestro proyecto de la Biblioteca:



- Nos aparecerá en la pantalla una vista previa del plugin. A la izquierda, el editor para escribir el código *PlantUML* entre las etiquetas *@startuml* y *@enduml* y a la derecha, una vista previa de lo que vamos añadiendo:

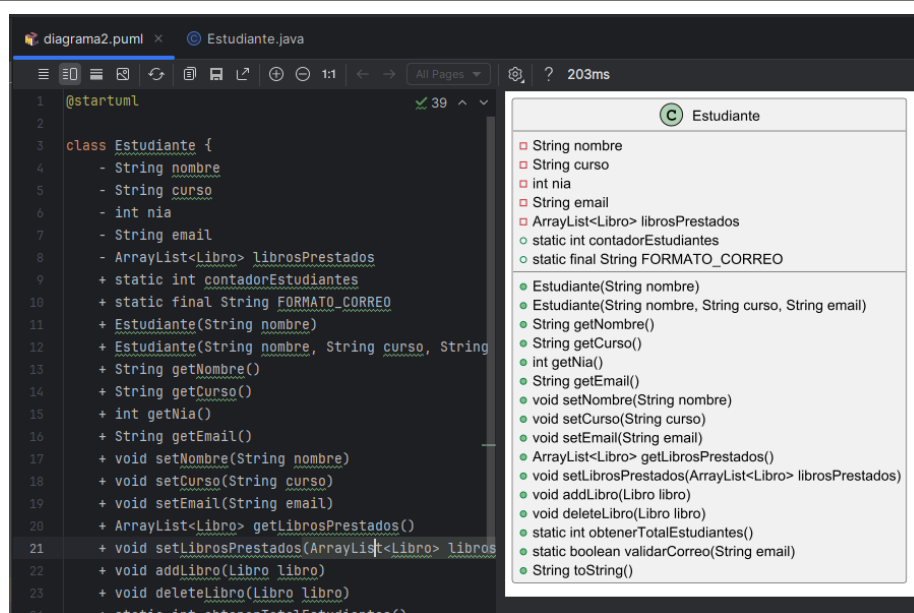
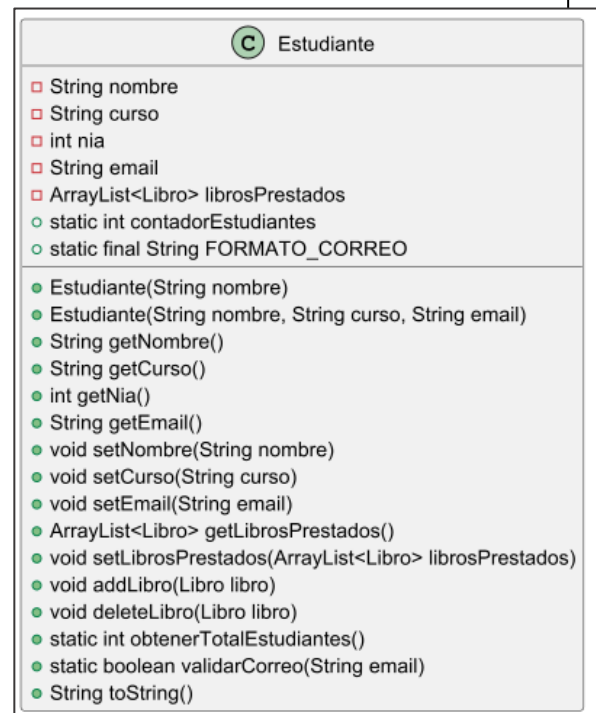


- Vamos a empezar a crear código *PlantUML*. Por ejemplo, para nuestra clase *Estudiante*, se generaría:

@startuml

```
class Estudiante {
- String nombre
- String curso
- int nia
- String email
- ArrayList<Libro> librosPrestados
+ static int contadorEstudiantes
+ static final String FORMATO_CORREO
+ Estudiante(String nombre)
+ Estudiante(String nombre, String curso, String email)
+ String getNombre()
+ String getCurso()
+ int getNia()
+ String getEmail()
+ void setNombre(String nombre)
+ void setCurso(String curso)
+ void setEmail(String email)
+ ArrayList<Libro> getLibrosPrestados()
+ void setLibrosPrestados(ArrayList<Libro> librosPrestados)
+ void addLibro(Libro libro)
+ void deleteLibro(Libro libro)
+ static int obtenerTotalEstudiantes()
+ static boolean validarCorreo(String email)
+ String toString()
}
```

@enduml



¿Te has asustado? Normal... Porque esto nos lo va a hacer *ChatGPT* 🙌

3. Uso de IA para generar código *PlantUML*

Como ya hemos visto, la facilidad de usar esta herramienta no sería tal si tuviéramos que escribir el código de nuestras clases todo el rato... Pero vamos a hacerlo más sencillo.

- Abre *ChatGPT* (o *Copilot*) y escribe un *prompt* para que nos genere automáticamente el código *PlantUML* de nuestras clases, así como sus relaciones. Le pediremos algo así:

hola copi, conoces plantuml?

¡Hola! Sí, conozco PlantUML. Es una herramienta de código abierto que permite crear diagramas UML (Lenguaje Unificado de Modelado) a partir de una descripción textual. Esto significa que puedes escribir un simple archivo de texto con la especificación de tus diagramas, y PlantUML se encargará de generar el diagrama gráfico correspondiente.

PlantUML es bastante versátil y admite varios tipos de diagramas, como diagramas de clases, secuencia, de actividades, de casos de uso, de componentes y más. Es muy útil para desarrolladores de software y arquitectos que necesitan visualizar sus diseños de una manera clara y precisa.

¿Estás interesado en aprender a usar PlantUML o ya lo has estado utilizando?

puedes generar el código de un diagrama de relación entre varias clases que te voy a dar?

¡Por supuesto! Dame los detalles de las clases y sus relaciones, y te generaré el código para el diagrama UML en PlantUML. Puedes incluir los nombres de las clases, sus atributos, métodos y cómo se relacionan entre sí (por ejemplo, herencia, asociación, composición, etc.). ¡Estoy listo para ayudarte!

```
package org.example.Biblioteca;

import java.util.ArrayList;

/**
 * Representa a un estudiante que tiene libros prestados.
 * Este estudiante puede pedir prestados libros de la biblioteca.
 */
public class Estudiante {

    public static int contadorEstudiantes = 0;
    public static final String FORMATO_CORREO = "[A-Za-z0-9+_.-]+@alu.edu.gva.es$";

    private String nombre;
```

...

```

        getPais() + " librosPublicados=" + getLibrosPublicados() + "];

    }

}

```

Aquí tienes el código de PlantUML para generar el diagrama de relaciones entre las clases **Estudiante**, **Libro**, **Prestamo** y **Editorial**:

```

Plantuml
@startuml
package org.example.Biblioteca {
    class Estudiante {
        - String nombre
        - String curso
        - int nia

```

En general, el resultado del modelado de las clases será correcto. Pero fíjate...

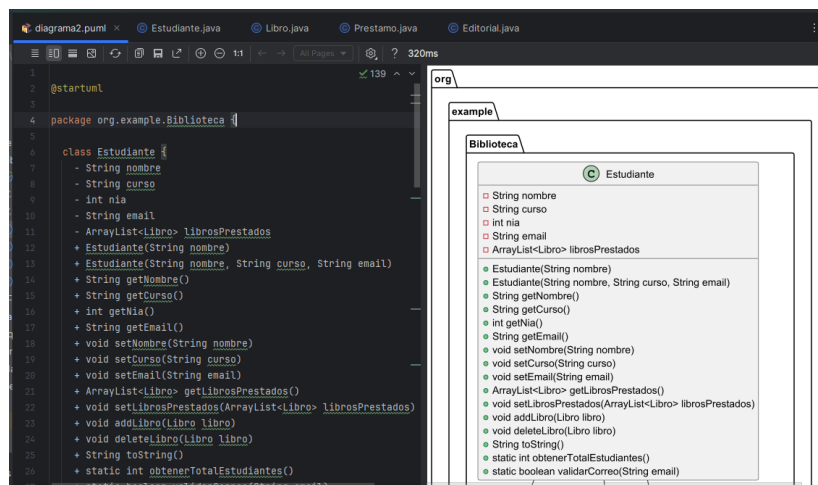
Para crear las relaciones, al final del código, escribe unas líneas tal que así:

```

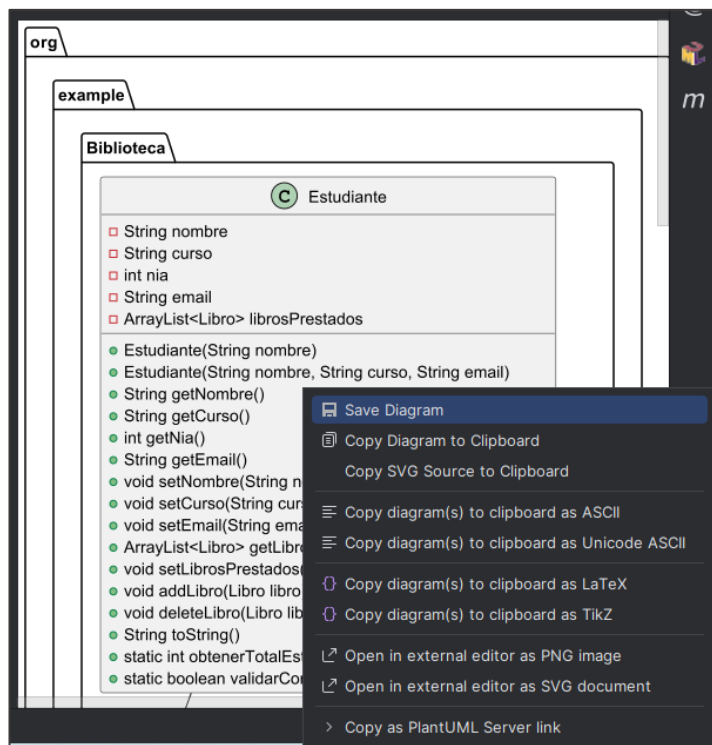
Estudiante "1" o-- "0..*" Libro : presta >
Estudiante "1" -- "0..*" Prestamo : realiza >
Libro "1" -- "1" Editorial : pertenece a >
Prestamo "1" -- "1" Libro : incluye >
Prestamo "1" -- "1" Estudiante : realizado por >

```

Esta parte siempre deberemos revisarla, ya que tiene por costumbre confundir cuando una relación es de asignación/agregación (utiliza el símbolo **o**) y de composición (utiliza el símbolo *****). **Si detectamos algún error, deberemos guiarlo hasta que devuelva el resultado que queremos.**



- Para guardar o exportar el diagrama tenemos varias opciones. Haz *click* derecho sobre el diagrama para verlas:



De todas las que hay, normalmente usaremos **Save Diagram** para guardarla en PNG o simplemente la copiaremos para pegarla en algún informe de prácticas.

Otro uso curioso es que *PlantUML* nos permite subirlo a su servidor para poder tenerlo accesible fuera de *IntelliJ IDEA* mediante una *URL*. Para ello, usaremos la opción **“Copy as PlantUML Server link”** y veremos que nos abre la plataforma online vista anteriormente, con nuestro diagrama precargado.

[PlantUML Web Server](#)

→ REALIZACIÓN DE LA PRÁCTICA

Sigue los siguientes pasos para realizar la práctica. A partir de ahora, haz capturas de todo y ve pegándolas en un documento de texto. **¡Guarda el documento de vez en cuando para evitar que se borre el avance de tu trabajo si se cierra el editor de textos u ocurre cualquier problema en tu equipo!**

1. Realiza el mismo proceso visto anteriormente para generar el diagrama del ejercicio *Biblioteca (Estudiante – Libro – Prestamo – Editorial)*. Guárdalo en formato *PNG* en la misma localización que tengas las clases.
2. Genera el diagrama del ejercicio *Casa – Habitación – Electrodoméstico*. Guárdalo en formato *PNG* en la misma localización que tengas las clases.
3. Actualiza el repositorio remoto para que se reflejen las modificaciones realizadas y guardemos nuestros archivos *.puml* y diagramas también en *GitHub*. **Pega aquí la URL:**



ENTREGA

****CONVIERTE EL DOCUMENTO DE TEXTO GENERADO SIGUIENDO TODOS LOS PASOS A PDF Y SÚBELO A TRAVÉS DE LA ENTREGA DISPONIBLE EN AULES.**