[Fecha]

Víctor Izquierdo

Tatiana Alonso

Kai Ye

Coral Manzanedo

TG3

Desarrollo de Tecnologías emergentes

Contenido

[1. Autores del trabajo, planificación y entrega 3](#_Toc513246137)

[1.1 Autores 3](#_Toc513246138)

[1.2 Planificación 3](#_Toc513246139)

[1.3 Entrega 3](#_Toc513246140)

[2. Requisitos del prototipo a implementar 4](#_Toc513246141)

[2.1 Requisitos funcionales 4](#_Toc513246142)

[2.2 Otros requisitos 4](#_Toc513246143)

[3. Criterios de comparación en la implementación 5](#_Toc513246144)

[3.1 Criterio 1: Tiempo del desarrollo del diagrama de clases 5](#_Toc513246145)

[3.2 Criterio 2: Tiempo del desarrollo de diagrama de casos de uso 5](#_Toc513246146)

[3.3 Criterio 3: Tiempo del desarrollo de diagrama de actividades 5](#_Toc513246147)

[3.4 Criterio 4: Velocidad del sistema 5](#_Toc513246148)

[3.5 Criterio 5: Control y Libertad para Explorar 5](#_Toc513246149)

[3.6 Criterio 6: Errores 6](#_Toc513246150)

[3.7 Criterio 7: Recursos Necesarios 6](#_Toc513246151)

[3.8 Criterio 8: Aprendizaje 6](#_Toc513246152)

[4. Proyecto de implementación de un prototipo del sistema utilizando YUML 7](#_Toc513246153)

[4.1 Documentación de diseño 7](#_Toc513246154)

[4.2 Documentación de construcción 7](#_Toc513246155)

[4.3 Documentación de pruebas 8](#_Toc513246156)

[4.4 Documentación de instalación 8](#_Toc513246157)

[4.5 Manual de usuario 8](#_Toc513246158)

[5. Proyecto de implementación de un prototipo del sistema utilizando Plantext 9](#_Toc513246159)

[5.1 Documentación de diseño 9](#_Toc513246160)

[5.2 Documentación de construcción 9](#_Toc513246161)

[5.3 Documentación de pruebas 9](#_Toc513246162)

[5.4 Documentación de instalación 9](#_Toc513246163)

[5.5 Manual de usuario 9](#_Toc513246164)

[6. Comparación de las dos implementaciones 10](#_Toc513246165)

[6.1 Evaluación de los criterios en la implementación usando la tecnología A 10](#_Toc513246166)

[6.2 Evaluación de los criterios en la implementación usando la tecnología B 10](#_Toc513246167)

[7. Comparación de la implementación de las tecnologías 11](#_Toc513246168)

[8. Conclusiones 13](#_Toc513246169)

# 1. Autores del trabajo, planificación y entrega

## 1.1 Autores

Victor Izquierdo Ochoa: Grado sistema de información, Politécnica superior, Universidad Alcalá de Henares, España.

Coral Manzanedo: Grado sistema de información, Politécnica superior, Universidad Alcalá de Henares, España.

Tatiana Alonso Vegas: Grado sistema de información, Politécnica superior, Universidad Alcalá de Henares, España.

Kai Ye: Grado sistema de información, Politécnica superior, Universidad Alcalá de Henares, China.

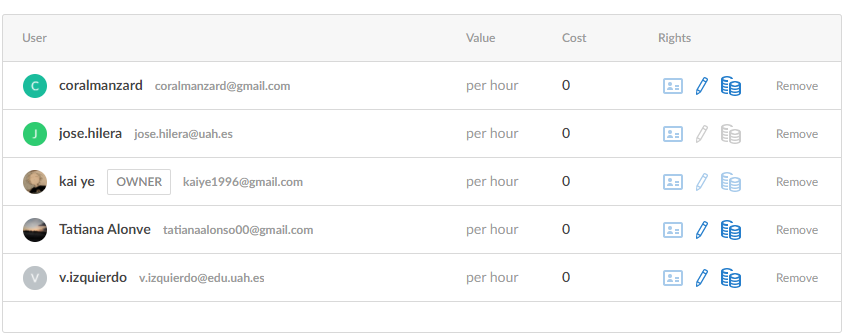
## 1.2 Planificación

Por periodo de prueba gratis con la herramienta de ganttpro, nos vemos obligado a crear una cuenta nueva para poder seguir utilizando la herramienta.

Debido a que hicimos uso de la plataforma de prueba de GanttPro, la planificación hecha no podemos compartir en enlace, a menos que pagemos una cuota premium. Por lo que optamos por incluir al profe como miembro para que pueda ver la planificación.



Compartiendo el proyecto a través de correo de UAH.



## 1.3 Entrega

<https://github.com/vizquierdouah/TG3.git>

# 2. Requisitos del prototipo a implementar

En este trabajo vamos a implementar un prototipo que será una empresa dedicada a la cadena de Bares con el propósito de ayudar a los desarrolladores, veremos de qué manera se tiene que implementar el software. Tenemos diferentes requisitos acordados dividiéndolos entre funcionales y no funcionales. Todo esto con el objetivo de comparar la implementación de un mismo prototipo de sistema utilizando dos tecnologías diferentes yUML y PlantUML (A y B).

## 2.1 Requisitos funcionales

En la siguiente tabla se indicará el catálogo de requisitos funcionales del sistema.

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| RF01 | Identificación de la herramienta |
| RF02 | Permitir elegir diagramas |
| RF03 | Enviar diagramas a otras aplicaciones |
| RF04 | Eliminar partes de un diagrama |
| RF05 | Permitir anotar escribir código |
| RF06 | Confirmación de guardado |
| RF07 | Permitir añadir anotaciones en el diagrama |
| RF08 | Compartir diagrama con otros usuarios |
| RF09 | Modificar diagramas |
| RF10 | Eliminar Diagramas |

## 2.2 Otros requisitos

En este apartado tenemos otros requisitos para el prototipo que no se consideran como funcionales. En la siguiente tabla se indicará el catálogo de requisitos no funcionales del sistema.

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| R01 | Seguridad al guardar diagramas |
| R02 | Seguridad al hacer diagramas |
| R03 | Identificación y Registro a la plataforma |

# 3. Criterios de comparación en la implementación

En este trabajo definiremos criterios para la comparación de la implementación de las tecnologías en la construcción del prototipo de sistema de ejemplo que ya hemos expuesto en el apartado 2 con los requisitos establecidos.

## 3.1 Criterio 1: Tiempo del desarrollo del diagrama de clases

*Nombre del criterio: Tiempo de creación del diagrama de clases del sistema.*

*Descripción: Horas invertidas en la creación del diagrama de clases*

*Tipo de valor: Numérico (minutos)*

## 3.2 Criterio 2: Tiempo del desarrollo de diagrama de casos de uso

*Nombre del criterio: Tiempo de creación del diagrama de casos de uso*

*Descripción: Horas invertidas en la creación del diagrama de casos de usos*

*Tipo de valor: Numérico (minutos)*

## 3.3 Criterio 3: Tiempo del desarrollo de diagrama de actividades

*Nombre del criterio: Tiempo de creación del diagrama de actividades*

*Descripción: Horas invertidas en la creación del diagrama de actividades*

*Tipo de valor: Numérico (minutos)*

## 3.4 Criterio 4: Velocidad del sistema

*Nombre del criterio: Tiempo de carga del sistema*

*Descripción: Velocidad de carga y actualización del sistema para mostrar los diagramas creados*

*Tipo de valor: Numérico (segundos)*

## 3.5 Criterio 5: Control y Libertad para Explorar

*Nombre del criterio: Control y Libertad para Explorar en la herramienta*

*Descripción: Control y Libertad para Explorar en la herramienta en la creación del diagramas*

*Tipo de valor: Texto libre*

## 3.6 Criterio 6: Errores

*Nombre del criterio: Errores que surgieron en la creación del diagrama de clases del sistema.*

*Descripción: Errores surgidos haciendo los diagramas utilizando el editor de la herramienta*

*Tipo de valor: Numérico (ocasiones)*

## 3.7 Criterio 7: Recursos Necesarios

*Nombre del criterio: Recursos necesarios en la propia aplicación.*

*Descripción: Horas invertidas en ver los recursos necesarios para crear los diagramas utilizando el editor de la herramienta.*

*Tipo de valor: Numérico(horas)*

## 3.8 Criterio 8: Aprendizaje

*Nombre del criterio: Tiempo de aprendizaje en el editor de la herramienta del sistema.*

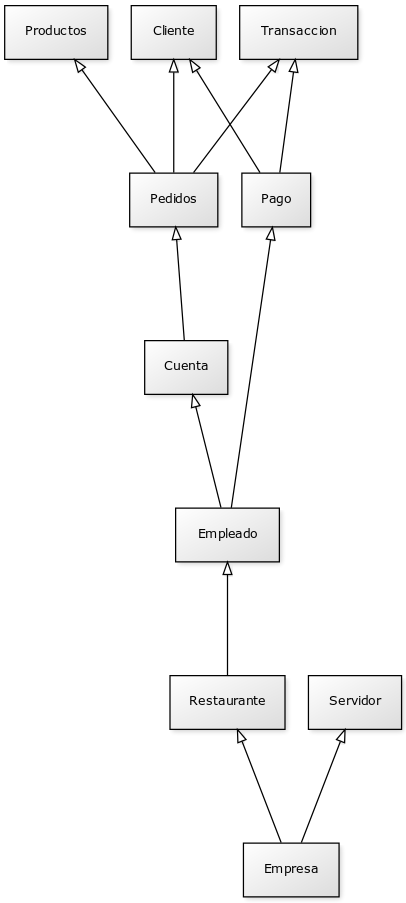
*Descripción: Horas invertidas en el aprendizaje utilizando el editor de la herramienta.*

*Tipo de valor: Numérico (horas).*

# 4. Proyecto de implementación de un prototipo del sistema utilizando YUML

## 4.1 Documentación de diseño

* Diagrama de Clases



* Diagrama de Casos de uso

|  |
| --- |
|  |

* Diagrama de actividades

|  |
| --- |
| Un pequeño ejemplo de interacción del camarero con la app. Para este caso hemos utilizado un diagrama de caso de usos creado de manera fácil y rápido. |

## 4.2 Documentación de construcción

Los diagramas se construyen escribiendo directamente en el recuadro reservado para ello en la página web. A la vez que vamos escribiendo se va dibujando e diagrama.

Para este caso vamos a coger el diagrama de casos de uso del apartado anterior. El código entero para la construcción del diagrama es:

|  |
| --- |
| [Camarero]-(Apunta pedido),(Apunta pedido)>(Modificar el pedido),[Camarero]-(Consultar el pedido),(Consultar el pedido)>(Modificar el pedido),(Consultar el pedido)>(Preparar el pedido),[Jefe de turno]-(Apunta pedido),[Jefe de turno]-(Consultar el pedido),[Jefe de turno]-(Validar el pago), [Gerente] - (Control de producción) |

## 4.3 Documentación de pruebas

Los casos de prueba se han basado en coger diagramas de otras asignaturas he intentar rehacerlos mediante texto utilizando la página web de la herramienta.

Han ido surgiendo varios problemas, por ejemplo:

* Cuando había un fallo, la aplicación no mostraba dibujo alguno, ni tampoco marcaba donde se encontraba el error. Esto nos obligaba a ir borrando lo último escrito, ya que si saltaba algún error iba a estar ahí, porque antes si mostraba las cosas correctamente.
* Otro problema se encuentra a la hora de elegir los tamaños y orientación de los esquemas, aunque a priori te deje marcar las opciones que tú quieras, luego ves que se ponen por defecto las cosas y no te deja cambiar nada.
* También un problema de la aplicación con el que hemos tenido que lidiar, ha sido que, en la página web, el dibujo se muestra debajo del texto. Por lo que a medida que se va escribiendo, el dibujo aparece más abajo o incluso oculto.

## 4.4 Documentación de instalación

La herramienta no necesita ser instalada para ser usada. Se puede hacer uso de su app web gratuitamente. Los diagramas creados en la web pueden ser implementados fácilmente en otras herramientas.

También existe la posibilidad de usar la herramienta como complemento o extensión, según el programa que sea, la instalación será diferente, aunque la propia web te explica como instalar el complemento en las herramientas más famosas.

## 4.5 Manual de usuario

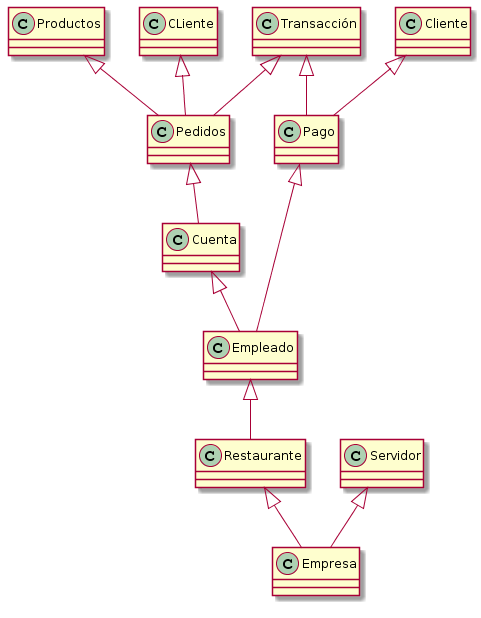
El código en la herramienta es bastante fácil, no necesita mucho tiempo de aprendizaje, es muy intuitivo. Además, no necesitas manuales para comenzar a crear diagramas, ya que te vienen unos ejemplos, desde unos facilillos hasta otros más complicados, los cuales tú puedes modificar para adaptarlos a tu gusto.

# 5. Proyecto de implementación de un prototipo del sistema utilizando PlantUML

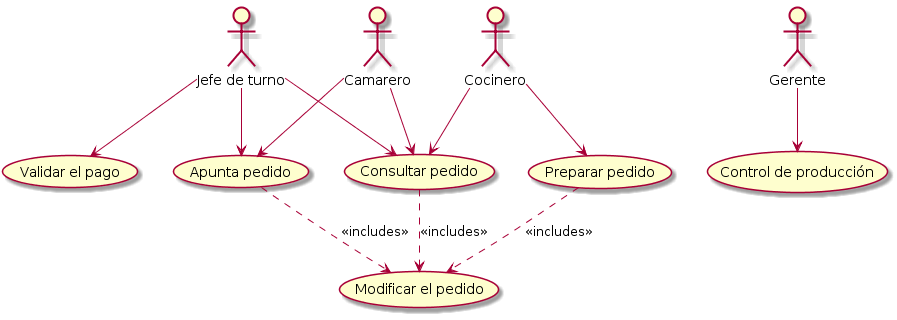
## 5.1 Documentación de diseño

En este caso utilizaremos el mismo prototipo que hemos utilizado en la herramienta anterior, y realizaremos los tres diagramas para poder comparar entre ambas herramientas:

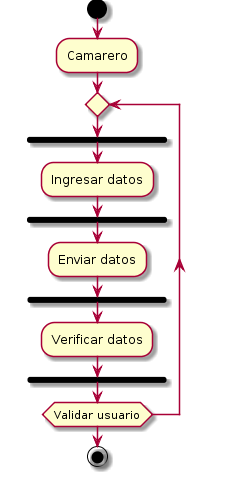
* Diagrama de clases:



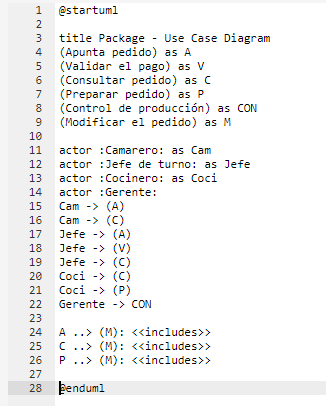
* Diagrama de caso de uso:



* Diagrama de actividades:



## 5.2 Documentación de construcción

El código de esta herramienta varía un poco de lo conocido habitualmente, pero acaba siendo intuitivo con la ayuda de todos los ejemplos de diagramas que incluye la herramienta. Para cada diagrama nos aporta distintos ejemplos de las cosas básicas que podríamos necesitar en cuanto a código. Por ejemplo, para realizar el diagrama de casos de uso nos ofrece ejemplos de casos de uso básicos, especificación de actores, conexiones, estereotipos o direcciones, lo que ayuda mucho a la hora de saber cómo hacer el diagrama sin tener muchos conocimientos de la herramienta. Por otra parte, en esta herramienta, para poder visualizar el diagrama primero hay que tener escrito el código, y una vez escrito procederemos a darle al botón “refresh”, el cual hace que aparezca el diagrama.

En la siguiente imagen observamos el código del diagrama de casos de uso en el que primero asignamos los nombres acortados para que sea más sencillo nombrarlos, después, declaramos los actores, los cuales podemos también asignarle otro nombre o acortarlo. Por último, trazamos las relaciones con flechas simples o con dos puntos si queremos hacer línea no continua.

## 5.3 Documentación de pruebas

Como casos de prueba hemos establecido un prototipo de software de otra asignatura y hemos intentado rehacer el caso y los diagramas mediante la herramienta online de PlantUML, que es PlantText: <https://www.planttext.com/>

En la realización de los distintos diagramas han ido surgiendo varios problemas, como:

* Al haber errores de código (sobre todo al inicio de la implementación debido al poco conocimiento sobre el lenguaje del código) simplemente muestra un mensaje indicando que hay un error, concretando la primera línea de código en la que muestra el error y lo que hay escrito en dicha línea, pero no indica cuál es exactamente el error, o no ayuda a saber como solucionarlo. Además, al haber un error no muestra ningún diagrama, y como el diagrama no se carga hasta que no pulsamos el botón “refresh”, no podemos saber si hay errores en el código hasta entonces.
* También hemos encontrado dificultades a la hora de orientar los diagramas ya que influye mucho el orden del código a la hora de la colocación de los actores. Al principio nos aparecía un actor en cada esquina del diagrama y líneas cruzadas por todo el diagrama con las relaciones. Esto supone un lío a la hora de interpretar el diagrama ya que no sabíamos si lo estábamos haciendo correctamente.
* Al realizar el diagrama de actividades, hemos encontrado más complicaciones. No nos aparecía el diagrama, pero tampoco nos indicaba que hubiera ningún error por lo que no sabíamos si lo estábamos haciendo bien, o era problema de la herramienta por que no estaba mostrando el diagrama.

## 5.4 Documentación de instalación

Esta herramienta nos permite dos opciones: acceder de manera online, o descargar la extensión de PlantUML para instalarla en otras herramientas más conocidas.

Nosotros hemos optado por la opción online, ya que este trabajo se basa en comparar dos herramientas de modelado UML online, por lo que los pasos para acceder a la herramienta de PlantUML online son sencillos. Como parte fundamental necesitamos tener acceso a internet para acceder a la página de <https://www.planttext.com/> en la que podremos comenzar a utilizar la herramienta onlie.

## 5.5 Manual de usuario

En primer lugar identificamos la herramienta accediendo a la página web de [PlantText](https://www.planttext.com/) (RF01), una vez aquí nos muestra una pantalla de código para poder escribir los diagramas, y encima de la pantalla un menú desplegable en el cual nos da todas las posibilidades de diagramas de PlantUML (RF02). En el menú desplegable también nos indica ayudas para editar los diagramas, o añadir anotaciones (RF05, RF07). Una vez que tenemos escrito el código para visualizar el diagrama debemos dar al botón de “refresh” para que se nos muestre el diagrama en la parte derecha de la pantalla. Para modificar o eliminar el diagrama simplemente deberemos cambiar el código escrito (RF04,RF09,RF10). Si queremos guardar el diagrama nos da varias opciones de formato, simplemente seleccionamos en el que queremos guardarlo (RF06). Otra opción que tenemos es registrarnos, se puede hacer fácilmente con la cuenta de Google, y así podremos guardar nuestros diagramas para modificarlos posteriormente.

# 6. Comparación de las dos implementaciones

## 6.1 Evaluación de los criterios en la implementación usando la tecnología A

Tecnología A yUML

| **CRITERIO** | **EVALUACIÓN** | **Comentario** |
| --- | --- | --- |
| 1. Tiempo del desarrollo del diagrama de clases | 2 minutos tarda en desarrollar el diagrama de clase. | Para poder desarrollar el diagrama de clase requiere un aprendizaje previo de 2 minutos para que el usuario comprenda las sintaxis. |
| 2. Tiempo del desarrollo del diagrama de casos de uso | 9.36 minutos tarda en desarrollar el diagrama de caso de usos. | Para poder desarrollar el diagrama de clase requiere un aprendizaje previo de 5 minutos para que el usuario comprenda las sintaxis. |
| 3. Tiempo del desarrollo del diagrama de actividades | 9 minutos tarda en desarrollar el diagrama de actividades. | Para poder desarrollar el diagrama de clase requiere un aprendizaje previo de 5 minutos para que el usuario comprenda las sintaxis. |
| 4. Velocidad de sistema | Tiempo de carga 0.2 segundos. | Dependiendo de la conexión de internet. |
| 5. Control y Libertad para Explorar | El usuario posee una gran libertad a la hora de escribir los códigos, pero exige que el usuario defina previamente la orientación y tamaño de los esquemas.  Por otra parte, el usuario debe poner todo el código dentro del cuadrado. Y el diagrama se genera debajo del texto, si es un código muy extenso, la imagen del diagrama se queda oculto. | Aspectos relacionados a la hora de escribir el código y generación del diagrama. |
| 6. Errores | En una ocasión, cuando el usuario escribe mal una línea de código, el programa deja de mostrar los esquemas hasta el usuario corrige ese error. | Errores que se generan durante la ejecución de la tecnología. |
| 7. Recursos necesarios | 2 horas en localizar y entender las documentaciones de la tecnología. | Tiempo necesario para localizar y entender las documentaciones. |
| 8. Aprendizaje | 12 minutos de aprendizaje para desarrollo del diagrama. | Tiempo necesario para aprender las sintaxis de yUML. |

## 6.2 Evaluación de los criterios en la implementación usando la tecnología B

Tecnología B PlantText

| **CRITERIO** | **EVALUACIÓN** | **Comentario** |
| --- | --- | --- |
| 1. Tiempo del desarrollo del diagrama de clases | 9.23 minutos tarda en desarrollar el diagrama de clase. | Para poder desarrollar el diagrama de clase requiere un aprendizaje previo de 2 minutos para que el usuario comprenda las sintaxis. |
| 2. Tiempo del desarrollo del diagrama de casos de uso | 14.58 minutos tarda en desarrollar el diagrama de caso de usos. | Para poder desarrollar el diagrama de clase requiere un aprendizaje previo de 2 minutos para que el usuario comprenda las sintaxis. |
| 3. Tiempo del desarrollo del diagrama de actividades | 6.13 minutos tarda en desarrollar el diagrama de actividades. | Para poder desarrollar el diagrama de clase requiere un aprendizaje previo de 4 minutos para que el usuario comprenda las sintaxis. |
| 4. Velocidad de sistema | Tiempo de carga 0.1 segundo. | Dependiendo de la conexión de internet. |
| 5. Control y Libertad para Explorar | La orientación de diagrama depende de la colocación de código, el usuario no tiene control directo sobre ello. | Aspectos relacionados a la hora de escribir el código y generación del diagrama. |
| 6. Errores | Un error encontrado durante el desarrollo del diagrama de actividad, el programa no muestra ningún esquema, tampoco indica donde está el error. | Errores que se generan durante la ejecución de la tecnología. |
| 7. Recursos necesarios | 3.5 horas en localizar y entender las documentaciones de la tecnología. | Tiempo necesario para localizar y entender las documentaciones. |
| 8. Aprendizaje | 13 minutos de aprendizaje para desarrollo del diagrama. | Tiempo necesario para aprender las sintaxis de PantText. |

# 7. Comparación de la implementación de las tecnologías

En la siguiente página se muestra una tabla resumen en horizontal con la comparación de los criterios y valoraciones de ambas tecnologías.

| **CRITERIOS** | **TECNOLOGÍA A** | **TECNOLOGÍA B** | **COMENTARIOS** |
| --- | --- | --- | --- |
| 1. Tiempo del desarrollo del diagrama de clases | 2 minutos | 9.23 minutos | En comparación apreciamos una diferencia de 7 minutos a la hora de desarrollar el diagrama de clases. Tecnología A tarda menos en desarrollar el diagrama. |
| 2. Tiempo del desarrollo del diagrama de casos de uso | 9.36 minutos | 14.58 minutos | En comparación apreciamos una diferencia de 5.22 minutos a la hora de desarrollar el diagrama de clases. Tecnología A tarda menos en desarrollar. |
| 3. Tiempo del desarrollo del diagrama de actividades | 9 minutos tarda en desarrollar el diagrama de actividades. | 6.13 minutos tarda en desarrollar el diagrama de actividades. | En comparación apreciamos una diferencia de 2.87 minutos a la hora de desarrollar el diagrama de clases. Tecnología B tarta menos en desarrollar el diagrama. |
| 4. Velocidad de sistema | Tiempo de carga 0.2 segundos. | Tiempo de carga 0.1 segundo. | Tecnología B tiene un menor tiempo de carga. |
| 5. Control y Libertad para Explorar | El usuario posee una gran libertad a la hora de escribir los códigos, pero exige que el usuario defina previamente la orientación y tamaño de los esquemas.  Por otra parte, el usuario debe poner todo el código dentro del cuadrado. Y el diagrama se genera debajo del texto, si es un código muy extenso, la imagen del diagrama se queda oculto. | La orientación de diagrama depende de la colocación de código, el usuario no tiene control directo sobre ello. | Tecnología A ofrece un mayor grado de libertad a la hora de diseñar el diagrama, aunque el usuario debe definir la orientación y el tamaño de esquema previamente. Mientras que tecnología B todo se controla por el código. |
| 6. Errores | En una ocasión es cuando el usuario escribe mal una línea de código, el programa deja de mostrar los esquemas hasta el usuario corrige ese error. | Un error encontrado durante el desarrollo del diagrama de actividad, el programa no muestra ningún esquema, tampoco indica donde está el error. | Tecnología B aporta un mayor control de errores, ya que indica la línea de código donde contiene el error, y el usuario ahorra el tiempo para localizar el error. |
| 7. Recursos necesarios | 2 horas en localizar y entender las documentaciones de la tecnología. | 3.5 horas en localizar y entender las documentaciones de la tecnología. | En comparación, el usuario tarda menos en localizar y entender las documentaciones de tecnología A. |
| 8. Aprendizaje | 12 minutos de aprendizaje para desarrollo del diagrama. | 13 minutos de aprendizaje para desarrollo del diagrama. | No hay mucha diferencia a la hora de realizar un aprendizaje para empezar el desarrollo del diagrama, pero sí que es cierto tecnología A tarta menos de aprender. |

# 8. Conclusiones

Bajo nuestro punto de vista, ambas herramientas nos han resultado muy útiles para realizar diagramas UML de manera online. Nos aportan rapidez ya que no es necesario que nos descarguemos e instalemos ningún programa, siempre que tengamos una conexión a internet.

En comparación, la herramienta de yUML resulta más llamativa a simple vista ya que su interfaz es bastante más amigable que la de PlantUML, y además es más sencilla de encontrar la URL de la herramienta online. Cabe destacar la simplicidad de código de yUML frente a PlantUML. El lenguaje que utilizamos en yUML resulta más cercano al lenguaje humano, y con la ayuda de los ejemplos que incluyen es bastante sencillo hacerse a la idea de cómo funciona el código.

Como desventaja de yUML hemos comprobado que, si tenemos algún error en el diagrama, simplemente no se muestra, ni india que exista error alguno, por lo que ralentiza bastante la tarea si no te das cuenta del error. Otra de las desventajas que encontramos en esta herramienta es que, a la hora de visualizar los diagramas se muestran en la parte inferior de la pantalla, por lo que si escribimos demasiado código llega un momento en el que dejamos de ver el diagrama y esto resulta bastante molesto e incómodo a la hora de trabajar con diagramas mas extensos ya que si escribes bastante código sin mirar el diagrama y después lo observas y ha habido un error, no se verá ningún diagrama ni errores, y habrá que borrar la mayor parte y observar dónde nos hemos equivocado.

Por otra parte, analizando la herramienta de PlantUML, el lenguaje que ésta utiliza es más lenguaje tipo código, no es tan cercano al lenguaje humano, ni tan complicado como lenguaje de programación, pero lo suficiente como para tardar un poco más examinando cómo realizar el diagrama. Otro de los motivos que ralentiza la ejecución del diagrama en esta herramienta es que tenemos que esperar a tener escrito el código para después visualizar el diagrama pulsando un botón. Esto resulta bastante incómodo ya que no vas viendo el proceso de tu diagrama y pierdes más tiempo realizando las modificaciones.

Las ventajas de PlantUML es que ofrece muchos ejemplos de diagramas y especifica más ejemplos dentro de cada uno de ellos. Otro de los pros de esta herramienta es que el diagrama se visualiza a la derecha de la pantalla, pudiendo escribir el código y posteriormente visualizarlo completo sin tener que desplazar la pantalla. Y, por último, otra de las ventajas que hemos apreciado en esta herramienta es que, a la hora de identificar errores resulta mucho más sencillo ya que indica en un panel en que línea está el error y cual es el código erróneo.

Como conclusión, si tenemos que usar alguna de estas dos herramientas para un determinado proyecto, en primer lugar, valoraríamos el tamaño del proyecto. Si se trata de un proyecto como el que hemos realizado en esta comparación (sencillo y corto) utilizaríamos sin duda la herramienta A, yUML, ya que nos proporciona más rapidez y facilidad con el diagrama. Sin embargo, si tenemos que realizar un proyecto más complejo o extenso, nos decantamos más por la herramienta B, PlantUML, ya que nos aporta más ayuda en información, ya sea de realización de diagramas o de información sobre los errores.