UML simplificado con PlantUML y VS Code

Introducción

UML son las siglas de Unified Modeling Language. Es un lenguaje de modelado de propósito general para estandarizar una forma de visualizar la arquitectura de los sistemas de software. Fue desarrollado por Grady Booch , Ivar Jacobson y James Rumbaugh en Rational Software en 1994–1995. Posteriormente, en 1997, se adoptó como estándar de la industria.

UML es una forma de expresar el diseño de componentes de software en términos de notaciones gráficas ampliamente aceptadas. A menudo es bueno tener un modelo gráfico antes de comenzar a codificar el modelo usando cualquier lenguaje de programación textual. Más adelante, el modelo también se puede utilizar con fines de documentación. La utilidad de UML se puede describir con el siguiente escenario de la vida real: cuando incorporamos nuevos desarrolladores, no queremos que lean cada línea de código y adivinen de qué se trata; queremos darles una descripción general de todo el sistema. Del mismo modo, cuando queremos que trabajen en una función, no queremos que solo se les informe verbalmente; queremos que tengan un plano de la característica requerida, de ahí el uso de diagramas UML.

Seguir el enfoque mencionado anteriormente puede aliviar muchos dolores de cabeza y malentendidos sobre los sistemas. A menudo, los defectos de diseño también quedan atrapados en este proceso.

UML comienza como un boceto en una pizarra con una cantidad mínima de detalles para poder tener una discusión de diseño con los miembros del equipo. Cuando se finaliza una decisión, se coloca un boceto relevante en una herramienta más sofisticada donde el resultado final funciona como un modelo para que un programador trabaje.

Tipos de UML básicos

Hay un montón de diagramas UML para elegir dependiendo de su caso de uso. Sin embargo, se clasifican en conjuntos de diagramas UML **estructurales** y de **comportamiento** .

Los diagramas estructurales reflejan la estructura de un sistema, mientras que los diagramas de comportamiento describen cómo reacciona el sistema ante determinadas acciones. En total, hay 14 diagramas UML; algunos son importantes mientras que otros son menos importantes. Son los siguientes ...

Diagrama estructural de UML

- Diagrama de clase
- Diagrama de objetos
- Diagrama de componentes
- Diagrama de implementación

- Diagrama del paquete
- Diagrama de estructura compuesta
- Diagrama de implementación

Diagrama de comportamiento UML

- Use el diagrama del caso
- Diagrama de actividad
- Diagrama de secuencia
- Diagrama de estado
- Diagrama de comunicación
- Diagrama de tiempo
- Diagrama de descripción general de la interacción

Solo codificaremos algunos diferentes. Siga adelante y descubrirá lo que quise decir con la codificación de diagramas.

Herramientas disponibles para modelar

Para modelar, un simple lápiz y papel pueden hacer el truco. Sin embargo, para documentación duradera y propósitos de edición frecuente, es posible que desee buscar un poco más de una herramienta de grado profesional. La elección de una herramienta puede venir en forma de escritorio, en línea o simplemente como un complemento IDE (Entorno de desarrollo integrado). Algunos pueden solicitarle suscripciones mensuales / anuales, mientras que otros son gratuitos. Algunas herramientas conocidas son las siguientes:

Producto	Plataforma	Modelo de precios
Microsoft Visio	Escritorio / en línea	Shareware
draw.io	En línea	Freeware
LicidChart	En línea	Shareware / Freeware
PlantUML , Eclipse UML2	Complementos / extensiones IDE	Freeware

PlantUML al rescate

Si bien Visio y draw.io parecen buenas opciones, uno necesita unos pocos dólares después de su período de prueba gratuito y otro necesita una conexión constante a Internet. Es por eso que PlantUML suena genial si recién está comenzando y luchando por tomar una decisión.

PlantUML asume la tarea de hacer diagramas de manera un poco diferente. A diferencia de arrastrar y soltar diferentes formas de la caja de herramientas y conectarlas, usted expresa su diagrama en términos de un pseudo lenguaje de programación. El idioma es muy fácil de entender y no lleva mucho tiempo llevarse bien.

Integración de código de Visual Studio

PlantUML admite una amplia gama de integraciones IDE. Todos los IDE compatibles se enumeran en el sitio web oficial de PlantUML, http://plantuml.com/running .

Optamos por VS Code, ya que últimamente hace calor y no hay señales de detenerlo. Antes de instalar la extensión PlantUML en VS Code, asegúrese de tener los siguientes requisitos previos:

- Java
- GraphViz

Para los usuarios de Windows, si tiene Chocolatey (administrador de paquetes para Windows), puede facilitar el proceso de instalación con el siguiente comando:

Ocultar código de copia

```
choco install plantuml
```

Puede buscar e instalar la extensión PlantUML desde la pestaña **Extensiones** . Está disponible en Visual Studio Marketplace. También puede usar la paleta de comandos de VS Code presionando **Ctrl + Shift + P** en Windows o **Comando + Shift + P** en Mac y escriba:

Ocultar código de copia

```
ext install plantuml
```

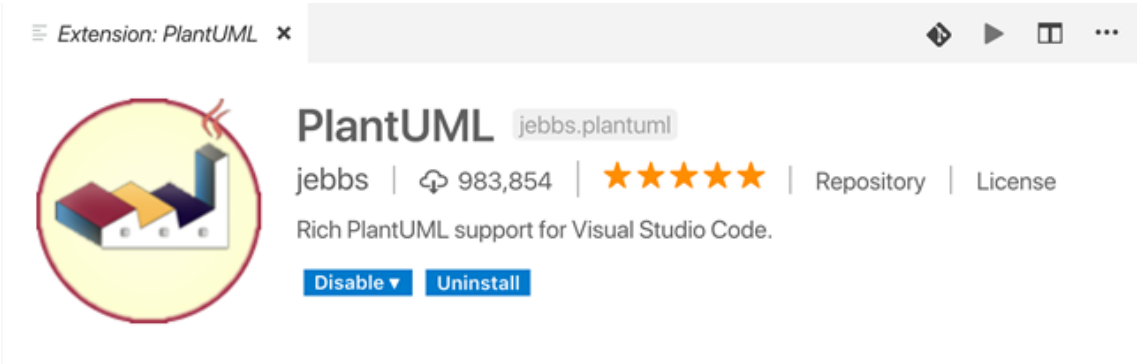

Figura 1: Instalación de la extensión PlantUML para código VS

Codificación de algunos modelos

Como se mencionó anteriormente, en PlantUML usamos un pseudo lenguaje de programación para generar diagramas. Este archivo de código puede tener una de las siguientes extensiones de archivo:

*** .wsd , * .pu , * .puml , * .plantuml , * .iuml**

Para realizar una prueba, cree un archivo con cualquiera de las extensiones mencionadas y pegue el siguiente código:

Ocultar código de copia

```
@startuml
```

```
scale 3
Alice -> Bob : test
@enduml
```

Presione **Alt + D** u **Opción + D** para obtener una vista previa del diagrama generado.

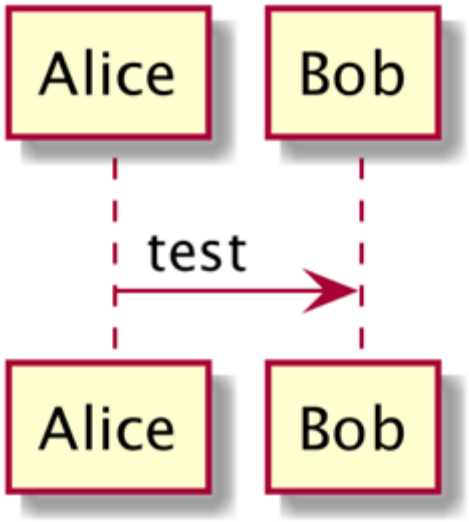

Figura 2: Prueba de ejecución con PlantUML en código VS

Diagrama de clase

Un diagrama de clases representa objetos en un sistema y varias relaciones entre ellos. Es el diagrama UML más común que encontrará al diseñar un sistema. Una clase contiene atributos que son simplemente campos, propiedades y operaciones que son simplemente métodos. Tanto los atributos como las operaciones tienen sus propios modificadores accesibles (visibilidad) expresados en términos de operadores como (+, -, ~, #) p. Ej. + Campo1 significa que hay un campo en la clase que es `public`.

Un diagrama de clases para administrar un bootcamp puede tener la siguiente arquitectura:

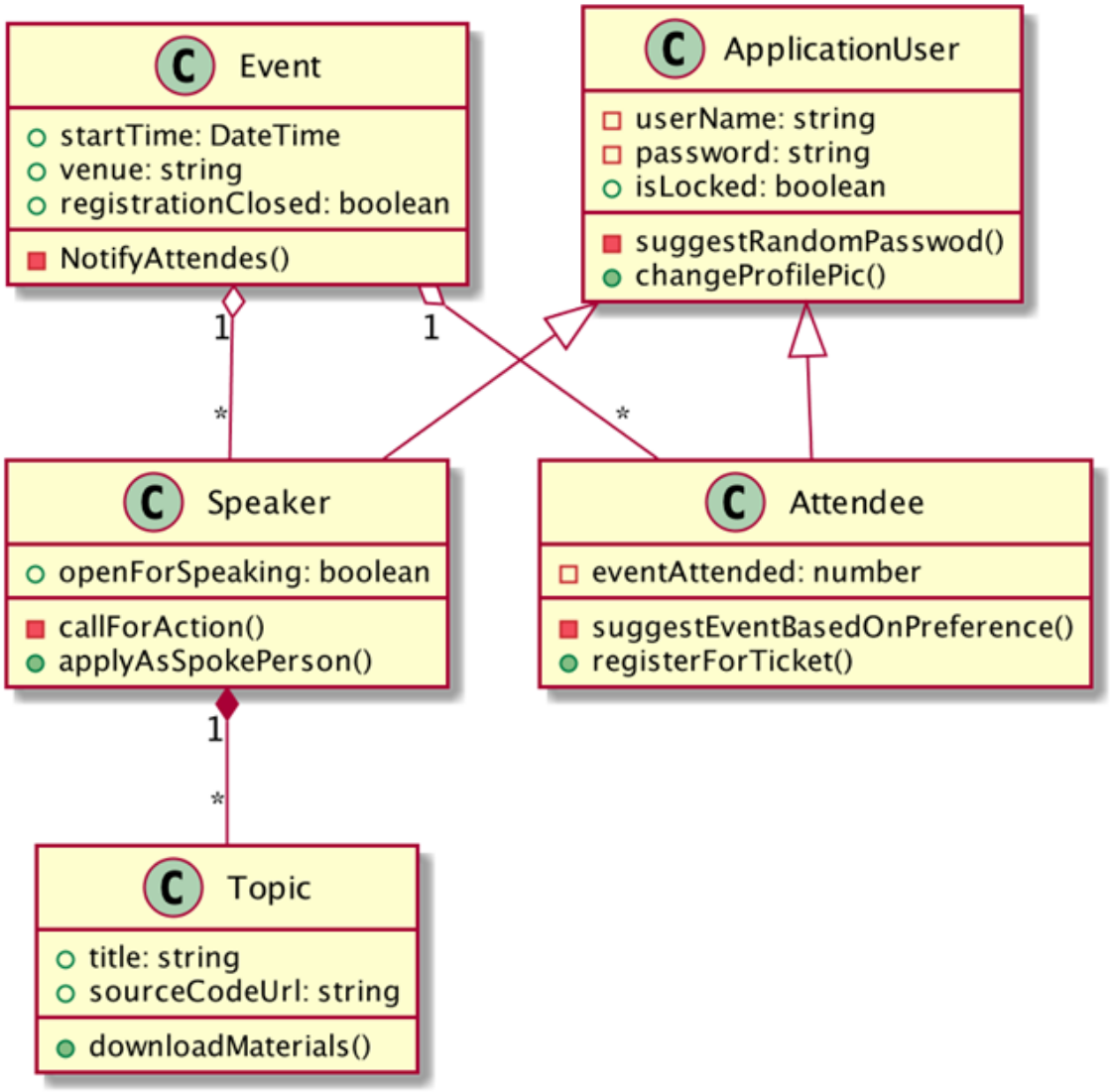

Figura 3: Diagrama de clases para la administración del bootcamp

El pseudocódigo de PlantUML correspondiente es el siguiente:


```
@startuml
scale 2
class Event {
    +startTime: DateTime
    +venue: string
    +registrationClosed: boolean
    -notifyAttendes()
}

class ApplicationUser {
    -userName: string
    -password: string
    +isLocked: boolean
    -suggestRandomPasswod()
    +changeProfilePic()
}
```

```
class Speaker {
    +openForSpeaking: boolean
    -callForAction()
    +applyAsSpokePerson()
}

class Topic {
    +title: string
    +sourceCodeUrl: string
    +downloadMaterials()
}

class Attendee {
    -eventAttended: number
    -suggestEventBasedOnPreference()
    +registerForTicket()
}

ApplicationUser <|-- Speaker
ApplicationUser <|-- Attendee
Speaker "1" *-- "*" Topic
Event "1" o-- "*" Speaker
Event "1" o-- "*" Attendee
@enduml
```

Vale la pena mencionar algunos de los símbolos del diagrama. Se describen de una manera más tabular.

Símbolos gráficos	Notación PlantUML	Sentido
Flechas vacías	<| -	Generalización / Herencia
Diamante lleno	* -	Composición
Diamante vacío	o—	Agregación

Diagrama de actividad

Los diagramas de actividad se utilizan ampliamente para describir el proceso empresarial y el flujo de trabajo. Se parece a un diagrama de flujo. Sin embargo, admiten comportamientos tanto secuenciales como paralelos.

Los diagramas de actividades pueden o no estar organizados en carriles de natación verticales. Los carriles de natación indican la participación de uno o más actores en el flujo de trabajo. En un diagrama de actividad, un círculo negro completo denota el inicio del proceso, mientras que un círculo hueco con un punto negro adentro denota un final. Los polígonos suelen ser fases de toma de decisiones. Las barras horizontales indican que dos o más acciones están sucediendo en paralelo.

Un diagrama de actividad para un sistema de compra de juegos en línea puede tener el siguiente flujo de proceso:

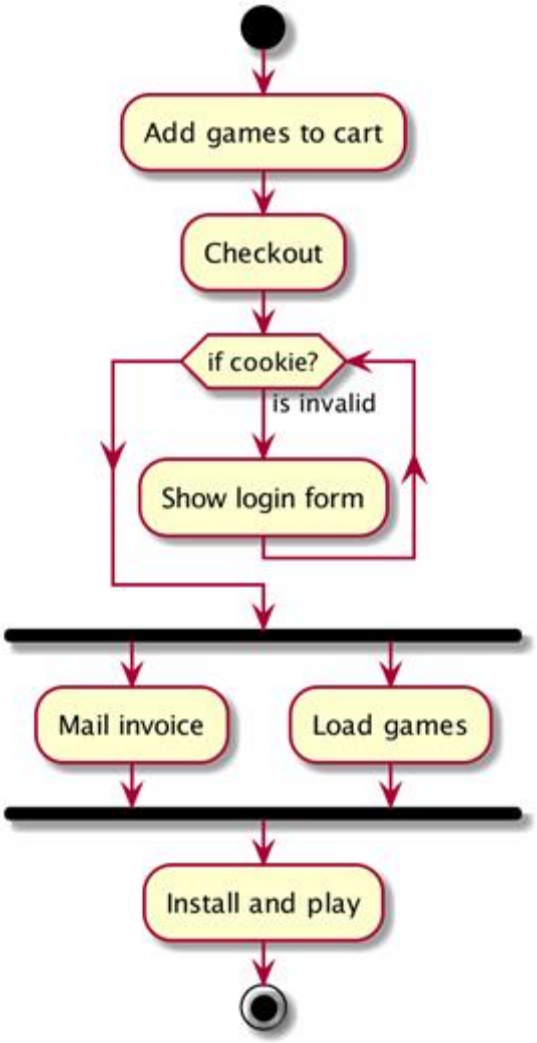

Figura 4: Diagrama de actividad para un proceso de compra de juegos en línea

El pseudocódigo de PlantUML correspondiente es el siguiente:

Ocultar código de copia

```
@startuml
scale 2
start
:Add games to cart;
:Checkout;
:Check cookie;
while (if cookie?) is (is invalid)
:Show login form;
endwhile
fork
:Mail invoice;
fork again
:Load games;
end fork
:Install and play;
stop
@enduml
```

Exportación para documentación

PlantUML admite la exportación de diagramas en diferentes formatos de archivo. Los siguientes son algunos de los tipos admitidos:

*** .png, * .svg, * eps, * .pdf, * vdx, * .xmi, * scmi, * html, * .txt, * .utxt, * .latex**

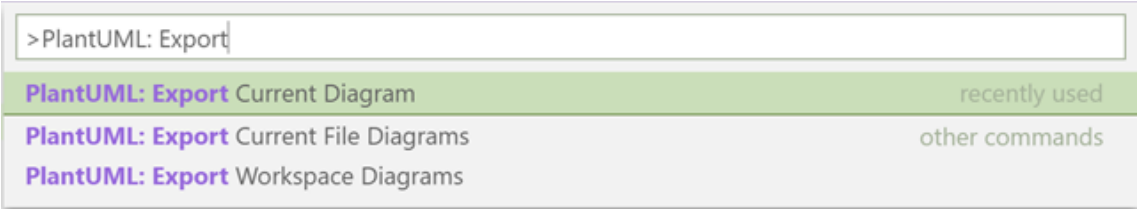

Figura 5: Exportación de diagramas desde código VS usando PlantUML

La ubicación de exportación predeterminada se establece en Escritorio. Sin embargo, seguir la configuración de usuario de VS Code asegurará que tanto el pseudocódigo como los diagramas estén junto a su directorio de código fuente en una carpeta llamada " *docs* ".

Ocultar código de copia

```
"plantuml.diagramsRoot": "docs/diagrams/src",
"plantuml.exportOutDir": "docs/diagrams/out"
```

Obtendrá resultados de exportación como:

Ocultar código de copia

```
Project Folder/
    docs/
       diagrams/
           src/
               architecture_overview.wsd
           out/
               architecture_overview.png
```