Federico Goldaracena

Instituto CEI

Carrera: Analista Programador Informe Trabajo Practico 1

# **Resumen**

El programa es una simulación de un Parlamento que permite a los usuarios realizar diversas operaciones relacionadas con legisladores y sus actividades en una legislatura. Los legisladores representan a diferentes partidos políticos y departamentos, y tienen la capacidad de presentar propuestas legislativas, registrar votos, participar en debates y más. El Parlamento gestiona a estos legisladores y proporciona funcionalidades como contar votos y listar legisladores y propuestas legislativas.

Utiliza ampliamente conceptos de programación orientada a objetos, incluyendo herencia y polimorfismo, para modelar y gestionar los legisladores y sus actividades en el Parlamento de manera eficiente y estructurada.

Contenido

[**Resumen** 1](#_Toc146809230)

[**1. Introducción** 3](#_Toc146809231)

[1.1. Lenguaje C# 3](#_Toc146809232)

[1.2. Programación Orientado a Objetos (POO) 3](#_Toc146809233)

[1.3. Herencia 4](#_Toc146809234)

[1.4. Polimorfismo 4](#_Toc146809235)

[**2.** **Metodología de estudio** 5](#_Toc146809236)

[2.1 UML Trabajo Practico 1 5](#_Toc146809237)

[**3. Referencias** 7](#_Toc146809238)

# **1. Introducción**

## 1.1. Lenguaje C#

C# (pronunciado "C sharp") es un lenguaje de programación de propósito general desarrollado por Microsoft. Fue creado a principios de la década de 2000 y se ha convertido en uno de los lenguajes de programación más populares y ampliamente utilizados en el mundo del desarrollo de software. C# Es conocido por su sintaxis limpia y estructurada, lo que lo hace relativamente fácil de aprender y leer. Este lenguaje se utiliza principalmente para desarrollar aplicaciones de escritorio, aplicaciones web y aplicaciones móviles en el entorno de desarrollo de Microsoft, incluyendo el uso de la plataforma .NET. Una de las características distintivas de C# Es su capacidad de programación orientada a objetos, que permite a los desarrolladores crear software modular y escalable. También ofrece soporte para la programación asincrónica, lo que facilita la creación de aplicaciones que pueden manejar múltiples tareas de manera eficiente.

Microsoft pone al alcance, de toda la comunidad planetaria de programadores, sus plataformas de desarrollo, como Visual Studio Code, de licencia gratuita (Freeware: no pagas por utilizarla), multiplataforma (para Windows, Linux y Mac OS), bajo el entorno .NET Core; y también Visual Studio (de pago), en sus versiones de 2017/2019, para Windows y Mac OS, bajo el entorno de .NET Framework. La diferencia entre ambos entornos es el destinatario final de las aplicaciones, siendo el último solo para Windows, mientras que el primero para las 3 plataformas mencionadas.

Para trabajar con C# es recomendable utilizar Microsoft Visual Studio, ya que ha sido desarrollado especialmente para ello y soporta la carga y trabajo con su mismo lenguaje.

Por ello, nos descargamos Microsoft Visual Studio y ejecutamos el programa. Puede ocurrir que te pida la instalación de algunas librerías esenciales para un correcto funcionamiento del programa, que nos la podemos descargar gratuitamente desde Microsoft.

Una vez ejecutado el programa, puedes crear un documento de tipo C# y ya puedes escribir en este código.[1]

1. POO

## Programación Orientado a Objetos (POO)

La Programación Orientada a Objetos (POO) es un enfoque de programación que se fundamenta en la utilización de objetos para modelar entidades del mundo real y llevar a cabo tareas complejas. La POO se encuentra presente en numerosos lenguajes de programación populares, tales como Java, Python, C++, y muchos más. En este artículo, examinaremos los cuatro pilares fundamentales de la POO: el encapsulamiento, la herencia, el polimorfismo y la abstracción.[2]

1. POO

## 1.3. Herencia

La herencia es una de las tres características principales de la programación orientada a objetos, junto con la encapsulación y el polimorfismo. Permite crear clases que reutilizan y extienden el comportamiento de otras clases. En este proceso, una clase base proporciona la estructura inicial, mientras que una clase derivada hereda sus miembros. La clase derivada puede especializarse aún más, representando diferentes aspectos de la clase base. Por ejemplo, una clase base "Animal" puede tener clases derivadas como "Mammal" y "Reptile", que son especializaciones de "Animal". Esto facilita la organización y la reutilización del código en la programación orientada a objetos. [3]

1. Herencia

## 1.4. Polimorfismo

La palabra "polimorfismo" proviene del griego y significa "varias formas diferentes". En programación orientada a objetos, esta idea es fundamental. Al igual que la herencia se relaciona con las clases y su jerarquía, el polimorfismo se relaciona con los métodos. Existen tres tipos de polimorfismo: Polimorfismo de sobrecarga: Esto ocurre cuando funciones con el mismo nombre y funcionalidad similar existen en clases completamente independientes. Por ejemplo, clases como "complex", "image" y "link" pueden tener una función llamada "display". Esto significa que no es necesario preocuparse por el tipo de objeto cuando se quiere mostrar en pantalla. Polimorfismo paramétrico: Aquí, se pueden definir varias funciones con el mismo nombre, pero diferentes parámetros. El método correcto se selecciona automáticamente según el tipo de datos pasados como parámetros. Por ejemplo, se pueden tener métodos "addition" que sumen enteros, flotantes, caracteres, etc. Polimorfismo de subtipado: Este tipo de polimorfismo permite redefinir un método en clases derivadas de una clase base. Esto se llama especialización y permite llamar a un método de objeto sin necesidad de conocer su tipo intrínseco. Por ejemplo, en un juego de ajedrez, el método "movimiento" podría realizar el movimiento correspondiente según el tipo de pieza que se llama, sin preocuparse por los detalles de cada pieza en particular. El polimorfismo en programación orientada a objetos permite utilizar métodos de manera flexible, adaptándolos a diferentes situaciones y tipos de datos, lo que simplifica el diseño y la reutilización de código. [4]

1. Polimorfismo

1.5. UML

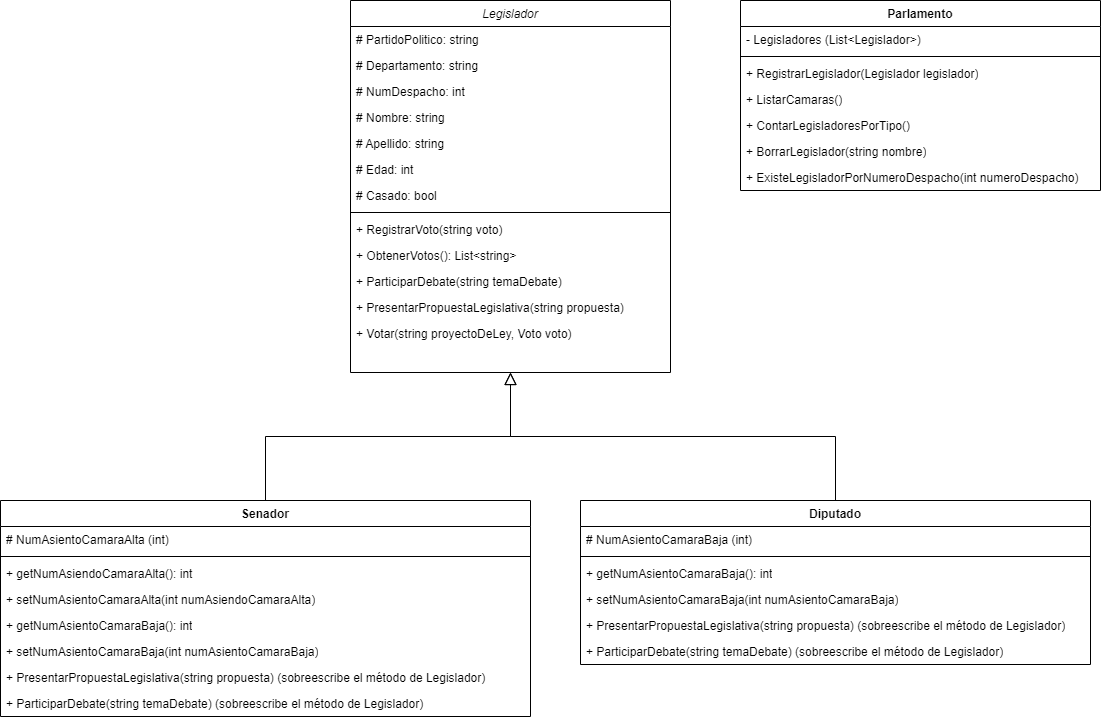
Un diagrama UML es una herramienta visual utilizada por ingenieros de software para comprender sistemas y software complejos. Ayuda a simplificar la comprensión de diseños, arquitectura de código y flujos de trabajo. Se basa en el Lenguaje Unificado de Modelado (UML) y se utiliza tanto en la programación de software como en la modelización de procesos empresariales. En lugar de enfrentarse a miles de líneas de código, un diagrama UML ofrece una representación visual más clara y fácil de comprender. Facilita el seguimiento de relaciones y jerarquías importantes entre las partes del código. Aunque se asemejan a los diagramas de flujo, los diagramas UML tienen características específicas. Estos diagramas simplifican la complejidad del código, lo que beneficia tanto a los ingenieros como a las partes interesadas no técnicas. Permiten mantenerse al tanto de los proyectos y evitan perderse en las complejidades inherentes a la programación. Además, ayudan a descomponer los componentes esenciales de un programa informático en partes más manejables. [5]

1. UML

# **Metodología de estudio**

## 2.1 UML Trabajo Practico 1

El diagrama UML representa un sistema de legislatura en C# con tres clases principales: Legislador, Senador y Diputado. Legislador es la clase base y tiene atributos como PartidoPolítico, Departamento, Nombre, etc. Puede registrar votos, presentar propuestas, votar en proyectos y participar en debates legislativos. Senador hereda de Legislador y agrega un atributo NumAsientoCamaraAlta. Sobrescribe los métodos de presentar propuestas y participar en debates. Diputado también hereda de Legislador y tiene un atributo adicional, NumAsientoCamaraBaja. Al igual que Senador, sobrescribe los métodos de presentar propuestas y participar en debates. Parlamento es una clase que maneja una colección de Legisladores. Permite registrar legisladores, contar senadores y diputados, listar las cámaras y verificar legisladores por número de despacho.Eeste diagrama UML modela la estructura y relaciones entre legisladores y su gestión en el contexto parlamentario.



# **3. Referencias**

[1] https://bsw.es/que-es-c/

[2] <https://webdesigncusco.com/conceptos-basicos-de-la-programacion-orientada-a-objetos-poo/#:~:text=La%20programaci%C3%B3n%20orientada%20a%20objetos%20(POO)%20es%20un%20paradigma%20de,%2C%20C%2B%2B%2C%20entre%20otros>.

[3] <https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/object-oriented/inheritance>

[4] https://algonzalezpoo.wordpress.com/polimorfismo/

[5] https://miro.com/es/diagrama/que-es-diagrama-uml/