educación IT
CODE YOUR FUTURE

Programación Orientada a Objetos

Evolución de los paradigmas de programación

A lo largo de la historia de la informática, los paradigmas de programación se han **adaptado a las necesidades cambiantes de la industria.** A excepción de la programación *Spaguetti*, los demás se siguen utilizando y coexisten en la actualidad.

Programación Spaguetti

Incluye a los lenguajes pioneros: Assembler (década de 1950), FORTRAN (1957), COBOL (1959), BASIC (1964).

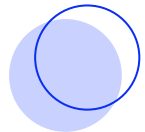
Caracterizada por el **uso excesivo de la instrucción `GOTO` y el código *spaghetti*** (con una lógica difícil de seguir).

Este enfoque de los primeros días de la programación fue desalentado debido a su **falta de legibilidad.**

Programación Estructurada/Imperativa

Lenguajes: Pascal (1970), C (1972), Ada (1980).

Se desarrolló como una evolución para **mejorar la legibilidad** y mantenibilidad del código. Se reemplaza `GOTO` por el **uso de métodos y estructuras de control.**



Programación Funcional

Lenguajes: Haskell (1990), Lisp (1958), ML (1973).

Se centra en uso de **funciones matemáticas.** Se utiliza en procesamiento de datos y desarrollo de sistemas concurrentes.

Programación Lógica

Lenguajes: Prolog (1972), Datalog (1977), Mercury (1995).

Basada en **reglas lógicas y deducción. Emplea inferencias** para resolver problemas. Esencial en inteligencia artificial, procesamiento de lenguaje natural.

Programación Declarativa

Lenguajes: SQL (1974), HTML (1993), CSS (1996), XAML (2006).

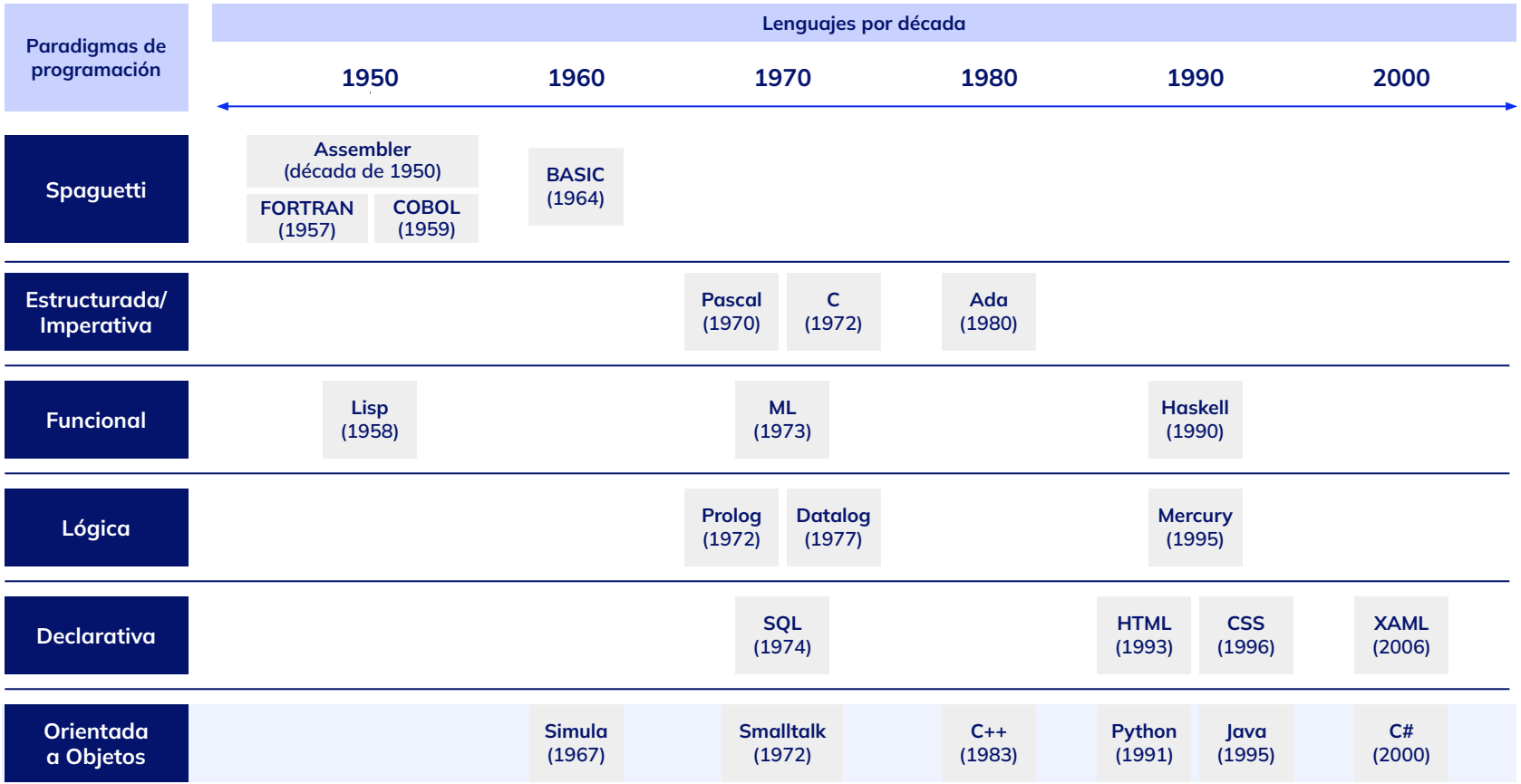
Estos lenguajes fundamentales en el desarrollo de sistemas modernos, permiten a los desarrolladores expresar **qué debe hacer el programa, en lugar de cómo** hacerlo.

Programación Orientada a Objetos

El paradigma que **revolucionó todo.**

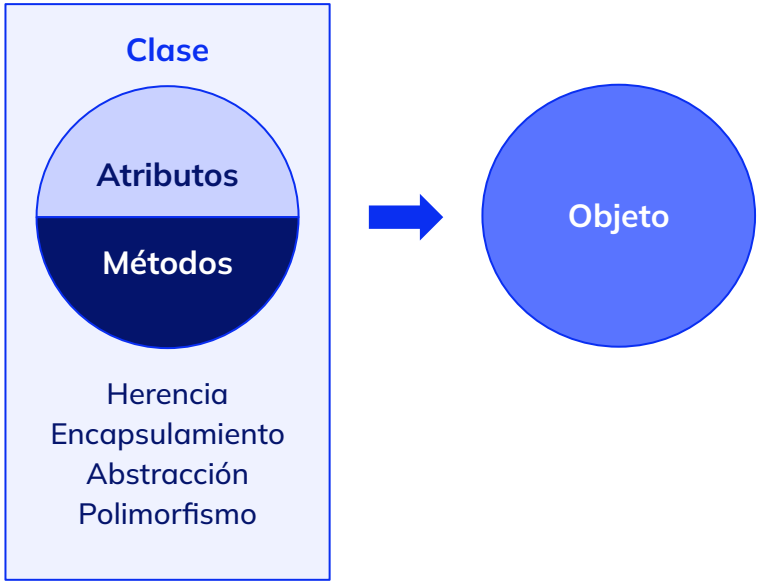
Surge en la década del 60, con Simula, que fue diseñado para la **simulación de sistemas.**

En las próximas diapositivas, lo desarrollaremos en detalle.


Paradigmas de programación
Lenguajes por década
1950
1960
1970
1980
1990
2000
Spaguetti
Assembler (década de 1950)
FORTRAN (1957)
COBOL (1959)
BASIC (1964)
Estructurada/ Imperativa
Pascal (1970)
C (1972)
Ada (1980)
Funcional
Lisp (1958)
ML (1973)
Haskell (1990)
Lógica
Prolog (1972)
Datalog (1977)
Mercury (1995)
Declarativa
SQL (1974)
HTML (1993)
CSS (1996)
XAML (2006)
Orientada a Objetos
Simula (1967)
Smalltalk (1972)
C++ (1983)
Python (1991)
Java (1995)
C# (2000)

Programación Orientada a Objetos

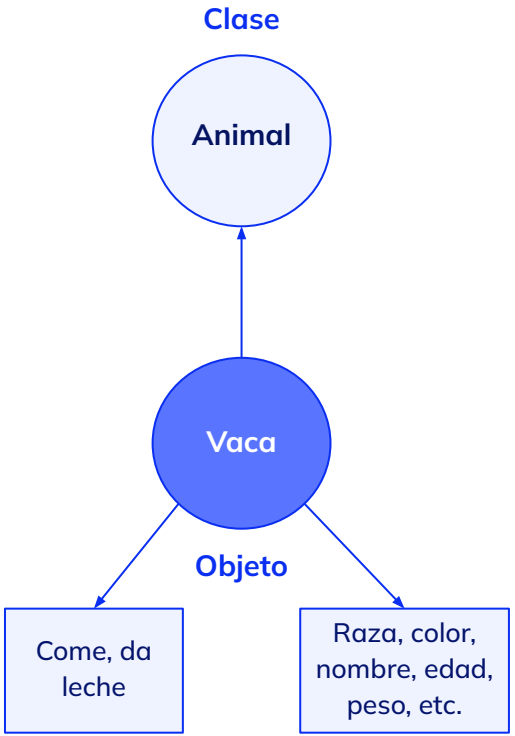
La POO ha revolucionado el desarrollo de *software* al proporcionar una **representación más natural de la realidad** para abordar los problemas y resulta un paradigma fundamental en el desarrollo de aplicaciones modernas.



Modelar la realidad

La POO busca emular el mundo real mediante la representación, en *software*, de **entidades llamadas objetos y sus interacciones.**

Observa las **características, el comportamiento y las interacciones** de los objetos para resolver problemas de manera intuitiva.



Modularidad

Anteriormente, el concepto de **módulo o librería** implicaba agrupar solamente subprogramas. Este enfoque resultaba en una **falta de claridad sobre la conexión** entre los datos.

La Programación Orientada a Objetos (POO) introduce conceptos adicionales de agrupación:

- Objetos.
- Clases.
- Paquetes / *namespaces.*
- *Frameworks.*

Esto permite una **organización más clara y cohesiva** de los datos y comportamientos relacionados.

Beneficios de la Programación Orientada a Objetos

- **Facilita el diseño de sistemas complejos:** Permite representar de manera más natural problemas del mundo real.
- **Promueve la reutilización del código:** Reduce el esfuerzo de desarrollo y acelera la entrega de *software.*
- **Mejora la mantenibilidad del *software:*** Facilita la modificación y actualización de una parte del código sin afectar otras partes del sistema.
- **Incrementa la productividad:** Al proporcionar una estructura clara y definida para el código, facilita la colaboración en equipo y la comprensión del *software.*
- **Flexibilidad en la extensión del *software:*** Permite añadir nuevas funcionalidades sin cambiar el código existente. Esto hace que el *software* sea más adaptable a nuevas necesidades.

Aplicaciones prácticas de la POO

- Desarrollo de juegos.
- Desarrollo de aplicaciones empresariales.
- Aplicaciones de interfaz gráfica.
- *Frameworks*.

Desarrollo de juegos

La POO permite el **modelado de personajes y escenarios** de videojuegos de forma intuitiva.

Este paradigma permite representar elementos como pociones y espadas **directamente como objetos** de un programa orientado a objetos.

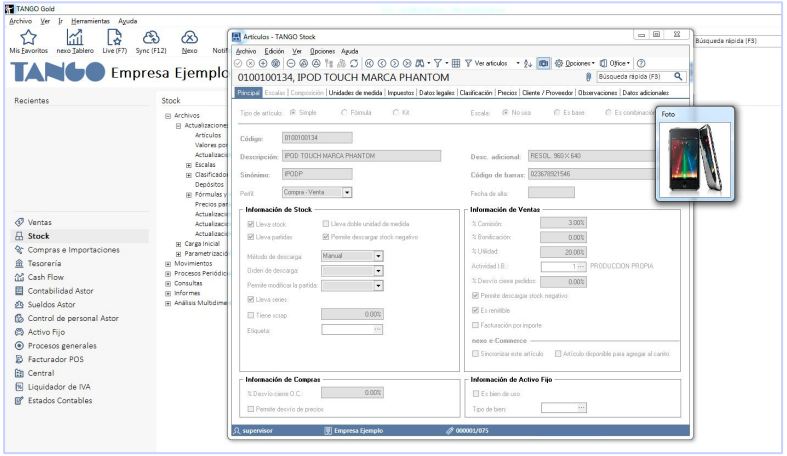
Seguramente tu juego favorito está desarrollado con programación orientada a objetos.

Desarrollo de aplicaciones empresariales

Estas aplicaciones necesitan adaptarse constantemente a una realidad cambiante y han ido evolucionando hasta encontrar, en la **programación orientada a objetos, la mejor manera de gestionar los datos** y funcionalidades empresariales.

Los conceptos de negocio como los clientes, las facturas o los pedidos, **se representan en memoria como objetos dentro de estos sistemas.** Estos objetos se suelen **persistir en una base de datos relacional SQL.** Esta técnica se conoce como **mapeo objeto relacional *(Object-Relational Mapping).***

Las aplicaciones empresariales que no migraron hacia este paradigma de programación se tornaron imposibles de mantener y terminaron siendo discontinuadas.

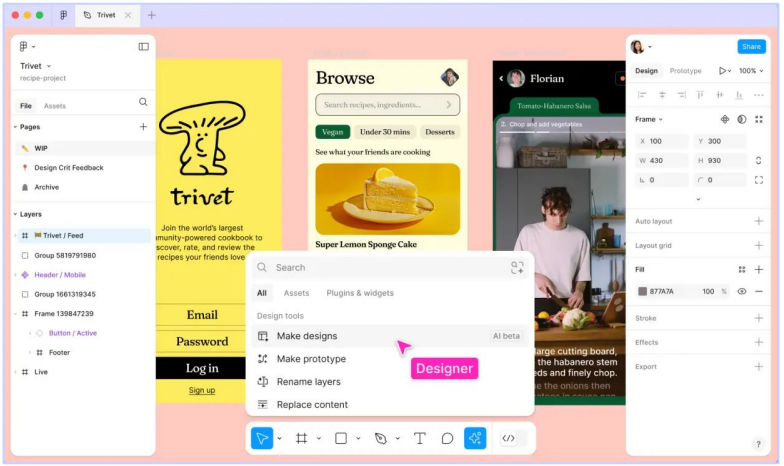


Aplicaciones de interfaz gráfica

La programación orientada a objetos permite una **organización clara** de los elementos de una interfaz.

Pensar en botones, cuadros de texto, combos, paneles y otros componentes **como objetos** resulta naturalmente **intuitivo y facilita su desarrollo**, incluso si el resto del sistema esta desarrollado con otro paradigma.

Los objetos que tienen una **representación visual** en una aplicación se suelen conocer como “Componentes”. En este contexto, se habla de la **programación orientada a componentes.**



Ejemplo: Figma.

Frameworks

Un *framework* es una **estructura** conceptual y tecnológica que proporciona una **base para el desarrollo de *software.***

Incluye un conjunto de **herramientas, bibliotecas, y pautas** que facilitan la creación de aplicaciones al proporcionar **componentes reutilizables** y soluciones estandarizadas para problemas comunes.

Los *frameworks* establecen una estructura específica y, frecuentemente, un **flujo de trabajo** que los desarrolladores deben seguir. Esta característica permite **acelerar el desarrollo** y promover buenas prácticas.

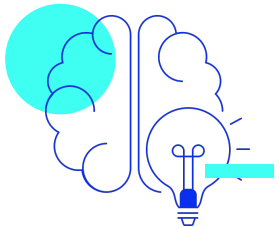
Los *frameworks*, frecuentemente, están **diseñados en torno a los principios de POO** y se benefician de ellos.

Importancia de la POO

Este paradigma, no solo enseña a codificar, sino a **crear, diseñar y pensar en grande**. Es la clave para desarrollar *software* que no solo funcione hoy, sino que también sea **escalable, mantenible y robusto** para el futuro.

La POO enseña a **dividir problemas complejos** en partes manejables. Cada **objeto representa una entidad** con responsabilidades claras y bien definidas. Esto hace que su código sea más fácil de entender, modificar y mejorar.

POO no es solo una habilidad técnica, es una **forma de pensar**. Ayuda a abordar problemas de manera más estructurada y creativa, aplicando conceptos que permiten diseñar **soluciones elegantes y eficientes.**



La POO en la era de la Inteligencia Artificial

En un mundo donde la IA está revolucionando industrias y mejorando vidas, **comprender los principios de la POO** se ha vuelto crítico.

Hoy no se trata solo de escribir código sino de entender y dominar **conceptos que permitirán diseñar sistemas inteligentes** escalables, flexibles y eficientes.

La IA podrá programar por nosotros, pero deberemos dominar la POO para **guiar a la IA en el desarrollo de sistemas** adaptativos que puedan evolucionar con las necesidades del futuro.