



Clase 1: Intro

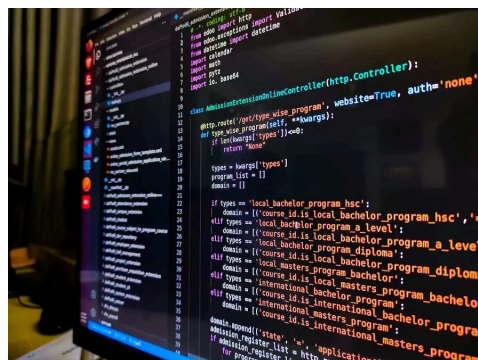
Bienvenidos a Entornos de Desarrollo



El profesor se llama **Cesar Tejedor** va a ser nuestro profesor de **Entornos de Desarrollo** de nuestro ciclo de **Desarrollo de Aplicaciones Web**.

De que va esta asignatura?

Entornos de Desarrollo lo que trata es que conozcamos de una manera (desde diferentes puntos de vista) lo que viene a ser un **desarrollo** → **Un Proyecto de desarrollo.**



Un Proyecto de Desarrollo que NO se enfoque solamente en el lenguaje de programación que vayamos a utilizar (como Java, C#, Php...) que no se enfoque

en una parte de ese proyecto. Sino en un enfoque global.



Tendremos un **enfoque global** desde el inicio hasta el fin de como se realiza un proyecto de desarrollo.

Pero... **¿Como se hace un proyecto de desarrollo a nivel de qué?**

- Conocer los tipos de lenguajes de programación y para que se utilizan.
- Que tipos de herramientas existen y para que se utilizan.
 - No es lo mismo un IDE como *VSCode*, *Eclipse* y como se utiliza eso.
- Que herramientas se utilizan para generar según que cosas del proyecto.
 - No es lo mismo hacer documentación que hacer un programa.
 - No es lo mismo generar un programa con un lenguaje (como Java, o JavaScript, o PHP) que desplegarlo en un entorno. Subirlo a un entorno con diferentes fases, a un repositorio, desplegarlo a un entorno de producción...

Digamos que todas las partes **necesarias** de gestión del *Software*, y gestión de *Herramientas* e procesos de un proyecto de desarrollo, desde su inicio hasta el fin — es lo que tenemos que conocer.



Es una visión 360: de todo lo que tiene que ver con un proyecto de desarrollo.

A nivel de:

- Cual es el ciclo de vida de desarrollo.
 - En un proyecto de desarrollo hay **fases** - una fase de análisis, una de requerimientos, una de preparación, despliegue...

Todo ese ***ciclo de vida de desarrollo*** es algo que **tenemos que conocer.**

Como digo, una **visión 360** de un producto.

Con esta asignatura obtendremos:

- Conocimiento a nivel de **lenguajes**
- Conocimiento de **herramientas**
- Conocimiento de **metodologías**
 - Agile, SCRUM, Waterfall...

Una metodología es **como hacemos las cosas** y como trabajamos en equipo.

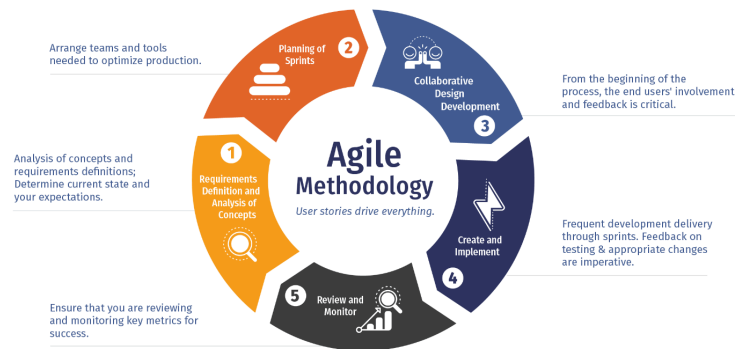
- Que herramientas necesitamos a nivel de software
 - Como se generan versiones y evoluciona un software...
 - **V1.0** , **V6.0** ...
- Como se hace la gestión del software y que herramientas tenemos.
 - Ya no solo es como generamos el programa, sino como lo probamos, como lo gestionamos, como lo subimos, como trabajamos en equipo..

Todo lo que este alrededor de nuestro programa (Software) en si, es lo que tenemos que conocer.

Esto es una asignatura un poco más **conceptual**, más teórica que otras, pero también tiene una parte **práctica** en la que nos pondremos manos a la obra.

Metodología AGILE

Vamos a trabajar sobretodo con la metodología más utilizada por equipos de desarrollo a día de hoy. La **metodología AGILE**.



En este caso un **SCRUM** — que ya veremos que lo vamos a entender rápido. Es una asignatura más sencilla que otras, más conceptual para otorgarnos una visión de todo lo que son las herramientas, del proyecto que nos da una perspectiva más amplia.

Ser desarrollador no solo implica saber *escribir código*, sino de entender como escalarlo, distribuirlo, gestionar ese software, tener en cuenta las mejores prácticas, los ciclos de vida, que herramientas se utilizan etc...



Mucha gente se piensa que ser desarrollador es solo escribir código en nuestro IDE.

En realidad tenemos una serie de estándares, una forma de hacer las cosas, teniendo en cuenta a compañeros que van a tocar nuestro código, etc...

Desde una etapa más inicial, hasta una etapa más final.

Que al final de lo que se trata esto es de que algo nazca como una idea, y se traduzca en un final en el que esta desplegado en un **entorno de producción** productivo e utilizable para el usuario final o el cliente.

Este es un poquito el resumen de la asignatura.

Conceptos sencillos, conceptos fáciles que uniremos a todo lo que estamos haciendo. Porque esta asignatura es una especie de **recubrimiento** de todo lo que hacemos.

Por eso iremos encajando las piezas muy rápido.


Presentación de las unidades

Vamos a ver un poco por encima las unidades que vamos a tocar, que es lo que nos va a pedir nuestro profesor Cesar, la planificación, etc...

El próximo día ya veremos contenido a nivel de lo que son las **planificaciones**, guías de aprendizaje, etc... Y veremos también la presentación de la unidad 1.

Unidad 1: Desarrollo de Software

Ya tenemos los libros sobre lo que son los temas en **PDF** — en la **Unidad 1** (Desarrollo de Software) vamos a hablar de la parte más teorica, de **que es un lenguaje de programación**, que tipos de lenguajes hay, si son lenguajes interpretados o compilados, que herramientas hay, que es un Framework, que es un IDE...

▼ Unidad 1: Desarrollo de software
Temario
 Desarrollo de software

Todo lo que vienen a ser **herramientas** — a nivel de lo que es el **desarrollo de software** desde un punto de vista más teorico.

Conocer más en profundidad los lenguajes de programación, que paradigmas existen, que es un paradigma (la manera o filosofia de hacer las cosas), que es una *programación orientada a eventos*, que es una *programación orientada a objetos* POO...

Conceptos desde un punto de vista más teorico, sobre el desarrollo de software. Para que entendamos que es lo que vamos a hacer.

Para que entendamos que (por ejemplo) cuando estemos con Java, se trata de un lenguaje de alto nivel, o de bajo nivel, o nivel medio... Un lenguaje compilado o desinterpretado...

Que sepamos de alguna manera cuando estemos en un lenguaje **de que va** y el porque de ello. Porque este lenguaje y porque sucede así, y porque tiene esta sintaxis...

Eso es lo que vamos a ver en la **unidad 1**.

Unidad 2: Instalación y uso de entornos de desarrollo

En la **unidad 2** lo vamos a llevar a nivel de **instalacion y uso de herramientas**. Saber si necesitamos Eclipse con un plugin porque queremos desarrollar X aplicación, vamos a conocer las herramientas de desarrollo y como funcionan a nivel de **que es lo que proveen a nivel de funcionalidad** y saber como elegir las e instalarlas.

▼ Unidad 2: Instalación y uso de entornos de desarrollo

Temario

Por ejemplo, Eclipse o IntelliJ son IDE's que aprenderemos a instalar porque son **entornos de desarrollo en si**, en esta unidad aprenderemos a instalarlas, conocerlas e utilizarlas.

Saber como depuramos, debuggeamos el código. Como instalar plugins para una funcionalidad, como es la configuración del desarrollo que vamos a hacer.

Es **conocer las herramientas de desarrollo** y lo que nos proveen.

Herramientas como por ejemplo:

- Eclipse
- IntelliJ
- VSCode...

Y todo lo que compone esas herramientas.

Unidad 3: Sistemas de control de versiones y herremientas de automatización

Como hemos mencionado brevemente antes, al crear un *Software* como desarrolladores, este va a ir evolucionando. Un programa al final se traduce como a un fichero (ejemplo: `gestionClientes.java`) ese fichero, que contiene el código de nuestro programa, tendra versiones.

▼ 3: Sistemas de control de versiones y herramientas de automatización

Temario

🔗 Sistemas de control de versiones y herramientas de automatización

Imaginate que el cliente pide una nueva funcionalidad, y se realizan dentro de 1 semana. Dentro de esa semana, haremos una **versión nueva**.

Si ahora tenemos la **1.0**, pues dentro de una semana dependiendo como se contabilice la **1.1** o la **1.2** → entonces todo ese versionado del proyecto, de los ficheros y todo ese código veremos **como versionarlo**.

Estaremos todos de acuerdo, que si tenemos hoy una versión de desarrollo, dentro de tres meses tendremos que tener esa funcionalidad más lo que hemos desarrollado en tres meses.



Imaginad que dentro de tres meses el cliente nos dice:

"Es que quiero ir atrás a una versión de hace 3 meses, para eliminar esa funcionalidad nueva"

Que crees que es mejor:

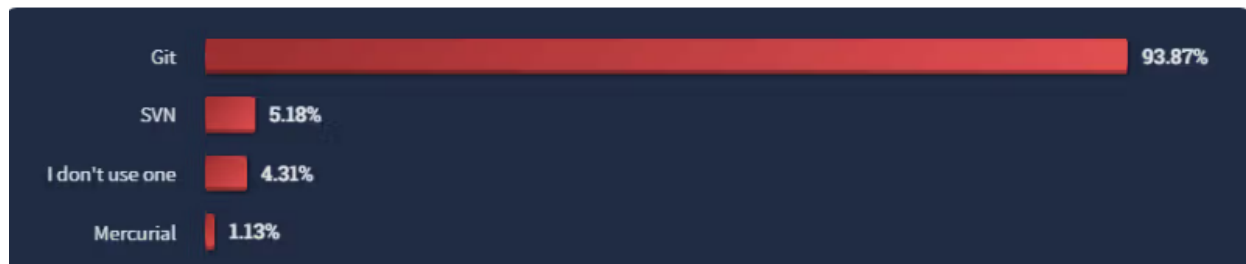
- **Eliminar todo el desarrollo manualmente.**
- **Ir directamente a la versión de hace tres meses.**

Es un poco la idea, de ir versionando el código y el proyecto en general para poder ir hacia atrás, hacia adelante, hacia los lados... No perder el código que ya habíamos hecho, partir del código de una fecha en concreto e ir evolucionandolo...

Todo ese sistema de control de versiones, o esa manera de trabajar con un control de versiones, en el 99% de los casos se hace con un sistema determinado y con herramientas determinadas, llamada **GIT** que es un *Software de Control de Versiones* y el más utilizado a día de hoy.



De hecho podemos ver en las encuestas de **Stack Overflow** que realmente lo utiliza la mayoría de equipos de desarrollo a día de hoy:



El trabajo de control de versiones para trabajar en equipo, se hace con herramientas como **git** que tendremos que aprender a utilizar (para no ir como locos por la selva tropical).

Se hace con **repositorios**, que son como almacenes que almacenan el código, donde se controlan las fuentes, que se evolucionan, etc...

Ahora mismo no tenemos que entender esto, porque ya nos pondremos al día cuando lleguemos a esta unidad.

Además de evolucionar nuestra aplicación a través de generar nuevas versiones, tenemos que conocer herramientas para **automatizar lo que son pruebas**.

Si queremos realizar pruebas de nuestra aplicación, y queremos un robot que vaya clickando en enlaces, y intente romper cosas, no tenemos que hacerlo manualmente ni robar un mono de la feria, tenemos herramientas que **automatizan ese testing**.


Incluso pruebas automatizadas para que (por ejemplo) el programa resista una carga de muchos usuarios, no tenemos una manera de replicarlo, porque igual sigue en nuestro equipo local y no está desplegado aún.

Pues tenemos herramientas automáticas para **estresar** nuestra aplicación con fines de prueba y testing.

Para garantizar que la aplicación funcione antes de enviarla a producción, y poder garantizarlo de alguna manera y no hacerlo solo, hay herramientas que generan esa "carga" o estrés.

Unidad 4: Diseño y realización de pruebas

Hay una parte en lo que es todo el **ciclo de vida de desarrollo** (que ya veremos que partes tiene) en una de las fases, se hace a partir de unos requisitos y unas especificaciones, se **trabaja en el diseño de la aplicación**.

4: Diseño y realización de pruebas
Temario
 Diseño y realización de pruebas


Esto es como si un arquitecto — una persona quiere construir una casa — y pide sus requerimientos:

- Quiero una piscina de estos metros y profundidad.
- Plantar estos arboles alrededor.
- Unas escaleras ahí...

Entonces el arquitecto lo apunta todo, y lo primero que hace en su estudio es **diseñarlo**.

Nosotros a nivel de aplicación de *Software*, también tenemos que **diseñar una aplicación**. El como y el que — dependiendo de lo que queramos hacer lo veremos en este tema.

También veremos lo que es la realización de **pruebas** (testing) todo tipo de pruebas de lo que es la aplicación.

 **Tener en cuenta:** No solo vamos a hacer una aplicación, no solo tenemos que hacer una aplicación.

Tenemos que hacer una aplicación que funcione bien, y asegurarnos de que funcione bien significa que tiene calidad.

Que sea legible (el código), que tenga un buen rendimiento, que cuando lo estreses se comporte bien, que la funcionalidad que me han mandado y especificado el cliente **concuera con lo que se ha desarrollado**.

Todo eso lo tenemos que probar, para asegurar una calidad **antes** de que este en productivo, que el cliente y sus usuarios lo vean.



Cuando nos referimos a producción: Nos referimos a cuando la aplicación esta en **REAL**.

Para asegurar su calidad, hay que hacer **pruebas**, en un entorno anterior.

Y eso es otra cosa que veremos en esta unidad.nid

Unidad 5: UML: El lenguaje de modelado unificado

En la **unidad 5**, para el diseño de las aplicaciones, o documentar las aplicaciones, vamos a trabajar en un lenguaje visual llamado **UML** — para **representar** como se comporta la aplicación.

▼ Unidad 5: UML: El lenguaje de modelado unificado
Temario
🔗 UML: El lenguaje de modelado unificado

Es decir, vamos a **representar desde diferentes puntos de vista**, desde diferentes maneras, de representar la aplicación.

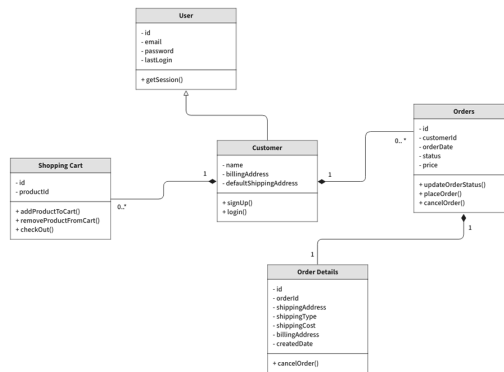
Con por ejemplo, un diagrama de componentes. Es decir, que componentes se utilizan en la aplicación.

Con un diagrama de secuencia, como la aplicación secuencialmente hace cosas.

Con un diagrama de actividad, que actividad hay en la aplicación.

Un diagrama de casos de uso, casos funcionales de como el cliente numero 1, pincha en un enlace y le aparece un listado de alumnos...

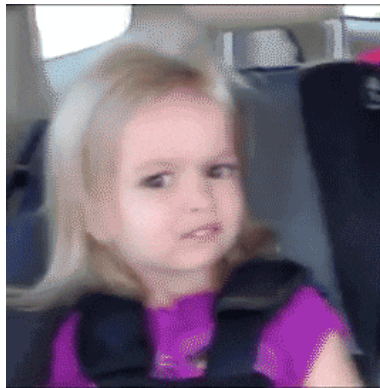
Todo eso, se determina en diagramas.



El lenguaje de modelado unificado (que utilizaremos una herramienta muy sencilla) lo representaremos con *diferentes diagramas* que aprenderemos a diseñar y entender.

En pocas palabras: es representar la aplicación desde diferentes puntos de vista.

Ahora te preguntarás... por que coño quiero diseñar diferentes diagramas de una aplicación y representarlo así en vez de picar código directamente?



Pues para que alguien y que la aplicación este bien documentada. Porque si voy a participar en un proyecto, o un desarrollo para una empresa, **tiene que haber un entendimiento de que es y como funciona la aplicación.**

Igual viene una persona nueva de fuera al equipo (de desarrollo o de la empresa) → que pueda entender.

Es dejar una especie de *documentación* que lo que haga es **facilitar el entendimiento de la aplicación desde diferentes puntos de vista.**

Unidad 6: UML: Optimización y documentación

Como podemos (y será la última unidad) → de alguna manera **mejorar una aplicación**. Mejorarla bien cuando estemos desarrollandola o bien cuando este terminada.

▼ 6. Optimización y documentación

Temario

🔗 Optimización y documentación

Hay maneras y maneras de hacer las aplicaciones.

Hay maneras de hacer una aplicación **bien hecha y con buenas practicas** de forma optimizada, y hay maneras mal hechas, mal optimizada y con mucho código spaghetti, desestructurada... Todo eso son problemas.

Entonces tenemos que aprender a **realizar buenas practicas** para hacer aplicaciones. Que ya veremos cuales son.

Esto a la hora de ir desarrollando, pero también veremos casos de proyectos que ya estan terminados, y se aprueba un presupuesto **para mejorarlo**.

Igual porque la aplicación tarda mucho, o tiene mal rendimiento, tiene mucho código y se podría reducir y mejorar la *mantenibilidad*, poder evolucionarla mejor...

Luego también veremos en esta unidad a nivel de **documentación** — ya no solo UML sino de documentos word, pdf, de guias, manuales asociados a un proyecto.

Como vamos a trabajar

Cesar (el profe) siempre trabaja de la misma manera. Al final lo que queremos es que después de cada unidad, que haremos algún trabajito con algún código o un ejemplo que nos quiera poner, cuando acabemos esa unidad.

Cesar subira uno o más ejercicios asociados a esa unidad, que en principio **NO** tendran fecha limite, sera para que hagamos **alguna practica**.

Eso en todas las unidades.

También, tendremos un **test de autoevaluación**. Para que veamos un poco si vamos bien, mal, repetirlo...

Cuando terminemos una unidad se subirán unos ejercicios y un test.

Esos ejercicios o prácticas que Cesar sube **NO son evaluables**, y sabemos que **SI son evaluables** las tres que haremos todo el año que son las **actividades trimestrales**.

Para todas las asignaturas son las mismas fechas, y que esas sí que serán evaluables.

Actividades evaluables

Estas fechas tenemos las actividades evaluables:

ENTREGAS 20%		Fecha máxima de entrega	Fecha máxima de subida a campus virtual	Fecha máxima para enviar a técnicas	Fecha máxima de corrección docente	Puntuación máxima ACTIVIDAD
Primer trimestre	Entrega T1 (todos los módulos)	30/11/2025	30/10/2025		20/12/2025	0,5
Segundo trimestre	Entrega T2 (todos los módulos)	22/02/2026	29/01/2026		15/03/2026	0,5
Tercer trimestre	Entrega T3 (todos los módulos)	26/04/2026	11/03/2026		16/05/2026	1
Simulacro		05/05/2026			Fecha máxima subida docente	

Estas son las entregas que **si que tenemos que hacer** → son evaluables. Si que cuentan para subir de nota. Si no las entregamos no pasa nada, no son obligatorias, pero tener en cuenta que nos subiran la nota de la asignatura.

Simulacro de examen

Para el final de la asignatura y antes del examen final, tendremos un **simulacro de examen** que lo haremos en Mayo (sobre el 5 de mayo).