# PEC 3

## TypeScript

Universitat Oberta de Catalunya

## Información relevante:

- Fecha límite de entrega: 10 de noviembre.
- Peso en la nota de FC: 15%.

# Contenido

# Información docente

## Presentación

Esta práctica se centra en desarrollar aplicaciones usando TypeScript sin la necesidad de utilizar un framework. Se crearán códigos en TypeScript que serán transpilados a JavaScript.

## Objetivos

Los objetivos que se desean lograr con el desarrollo de esta PEC son:

- **Desarrollar** código TypeScript.
- Comprender los **tipos de datos de TypeScript.**
- **Utilizar el patrón de arquitectura MVC** en el desarrollo de una aplicación Web.
- Comprender la complejidad del **transpilado** de una aplicación **TypeScript hacia JavaScript.**

# Enunciado

Esta PEC contiene 1 tarea y 4 ejercicios evaluables. Debes entregar vuestra solución de los 4 ejercicios evaluables (ver el último apartado).

Debido a que las actividades están encadenadas (i.e. para hacer una se debe haber comprendido la anterior), **es altamente recomendable hacer las tareas y ejercicios en el orden en que aparecen en este enunciado**.

Antes de continuar debes:

Haber leído los recursos teóricos disponibles en el apartado “Contenidos y recursos” del aula de esta PEC.

Crea una carpeta PEC3 e inicializa git en esa ruta. Al igual que en las anteriores PEC, debes crear un fichero README.md en la raíz de la carpeta que contenga que contendrá al menos esta información

• Login UOC

• Nombre y apellidos del alumno.

• Breve descripción de lo realizado en esta PEC, dificultades, mejoras realizadas, si hay que tener algo en cuenta a la hora de corregir/ejecutar la practica o cualquier aspecto que queráis destacar.

# Ejercicio 1 – Conociendo TypeScript (0,5 puntos)

TypeScript es un superconjunto sobre JavaScript que permite generar código de JavaScript que escale, según la propia definición de TypeScript. Es decir, está pensado para ser usado cuando JavaScript comienza a tener una complejidad de aplicación, dejando de lado su faceta de script.

En la PEC anterior se ha realizado una pantalla en la cual se hacía la gestión de una lista de TODOs y de gastos. Con ella has podido comprobar cómo la dificultad de gestionar JavaScript ha ido incrementando exponencialmente a medida que se introducen características/funcionalidades en la aplicación. No obstante, si analizas con cuidado la aplicación que se ha construido, no es más que una pantalla de las decenas que puede tener una aplicación web.

En esta PEC profundizaremos en TypeScript y las ventajas que nos aporta. En este ejercicio te proponemos que veas algunas de las ventajas que TypeScript tiene sobre JavaScript. Para ello te pedimos que compares el “mismo” código escrito en JavaScript y en TypeScript.

Antes de continuar debes:

- Descargar fichero PEC3_Ej1_Conociendo_TypeScript.zip y descomprimirlo dentro de la carpeta PEC3.

- Consultar el siguiente video sobre Typescript:

https://learning.oreilly.com/videos/understanding-typescript/9781789951905/9781789951905-video1_2

Ahora analiza el código en JavaScript (`Ejer1.js`) y su versión en TypeScript (`Ejer1.ts`)

Dentro de la carpeta PEC3_Ej1 crea un fichero Markdown que tenga como nombre PEC3_Ej1_respuestas_teoria.md en este fichero enumera y explica las principales ventajas que aporta el uso de TypeScript sobre JavaScript.

Durante el desarrollo del ejercicio ejecuta los comandos `git` necesarios para añadir esta nueva carpeta, ficheros y los cambios que realices en ellos al repositorio local `git` de la PEC.

# Tarea 1 – Configuración de TypeScript

Mira los 3 últimos vídeos del apartado “Setting up environment” del curso “Introduction to TypeScript” de Tamas Piros.

Te pasamos el enlace al primer vídeo que recomendamos ver:

https://learning.oreilly.com/videos/introduction-to-typescript/10000DIHV201907/10000DIHV201907-piros_u02m02

En esta tarea debes conseguir los ficheros resultado de las tareas realizadas en el video ( first.ts, first.js, y tsconfig.json ), es posible que alguna de las opciones del tsconfig.json que se indican en el video ya no funcionen en tu versión de TypeScript ( te lo indicara el Visual Studio Code o tu Editor) , averigua cual es el problema y corrígelo.

No es necesario entregar estos ficheros.

# Ejercicio 2 – Primeros códigos en TypeScript (3 puntos)

Antes de continuar debes:

Descargar fichero **`PEC3_Ej2_Primeros_codigos_TS.zip`** y descomprimirlo dentro de la carpeta PEC3.

Dentro de la carpeta nueva, crea un fichero `Markdown` que tenga como nombre PEC3_Ej2_respuestas_teoria.md

- Sitúate sobre las variables `a, b, c y d` y observa cómo TypeScript infiere el tipo de todas las variables por ti, de tal modo que `a` es un número, `b` es un número, `c` es un objeto con una específica forma, y `d` es también un número.
- Modifica el código para conseguir que aparezca una línea roja de error en el IDE avisándote de que se está disparando un `TypeError` . Toma una captura de pantalla de tu resultado y haz que se muestre dentro del fichero PEC3_Ej2_respuestas_teoria.md (0.5 puntos). Dentro de este mismo documento explica por qué se ha producido esto y qué ventajas tiene.

**En el mismo documento PEC3_Ej2_respuestas_teoria.md contesta las siguientes cuestiones:**

1. (1 punto) Para cada uno de los valores del fichero `code2.ts`, ¿Qué tipo de datos inferirá TypeScript? Explica por qué se ha inferido este tipo de datos.
2. (1 punto) ¿Por qué se dispara cada uno de los errores del fichero `code3.ts`?
3. (0.5 puntos) ¿Cuál es la diferencia entre una clase y una interface en TypeScript?

Durante el desarrollo del ejercicio ejecuta los comandos `git` necesarios para añadir esta nueva carpeta, ficheros y los cambios que realices en ellos al repositorio local `git` de la PEC.

# Ejercicio 3 – Práctica Typescript (2,5 puntos)

Antes de continuar debes:

Descargar el fichero **`PEC3_Ej3.zip`** adjunto en la PEC y descomprimirlo dentro de la carpeta PEC3.

a. (0,25 puntos) Analiza el fichero tsconfig indica con comentarios sobre este fichero que significan y para qué sirven cada una de las opciones del fichero tsconfig.json.

b. (0,75 puntos) Abre el fichero ejercicio1.txt ubicado en la carpeta src, cámbiale la extensión a .ts y sustituye los /***/ por las instrucciones adecuadas que cumplan las operaciones y salidas indicadas en los comentarios.

   Para comprobar resultado, desde la carpeta donde se encuentra el fichero tsconfig.json compila (tsc) y ejecuta (node ./dist/ejercicio1.js)

c. (0,75 puntos) Haz lo mismo con el fichero ejercicio2.txt

d. (0,75 puntos) Lo mismo para el fichero ejercicio3.txt

Durante el desarrollo del ejercicio ejecuta los comandos `git` necesarios para añadir esta nueva carpeta, ficheros y los cambios que realices en ellos al repositorio local `git` de la PEC.

# Ejercicio 4 – Arquitectura MVC usando TypeScript (4 puntos)

En este ejercicio vamos a transformar la aplicación TODO que te facilitamos en JavaScript en la PEC2 a TypeScript. Por tanto, todos los ficheros que componen nuestra aplicación son:

- `todo.model.js` – Los atributos (el modelo) de una tarea.
- `todo.controller.js` – El encargado de unir al servicio y la vista.
- `todo.service.js` – Gestiona todas las operaciones sobre los TODOs.
- `todo.views.js` – Encargado de refrescar y cambiar la pantalla de visualización.

Así pues, los ficheros anteriores deben transformarse a sus versiones TypeScript .

Para realizar esta transformación debes:

- Leer el recurso **`P02.PEC3_Arquitectura_MVC_TS.pdf`** que te guiara en cómo debes realizar este proceso.
- Descargar fichero **`PEC3_Ej4_Aplicacion_TODO.zip`** adjunto en la PEC

Una vez leído el documento y descargado el zip, realiza las siguientes acciones:

a) (0.50 puntos) Construye las clases relativas a modelos, controladores, servicios, vistas y lanzador de la aplicación desde donde irás desarrollando la aplicación. En este punto sólo debes crear la estructura de ficheros que modelan nuestro problema. Es decir, organizar las clases relativas a modelos (`todo.model.ts`), controladores (`todo.controller.ts`), servicios (`todo.service.ts`) y lanzadora (`app.ts`). (comprueba que se transpila correctamente todo y que todas las partes están conectadas).

b) (0.75 puntos) Codifica completamente la clase modelo (anémico) que sea necesaria para esta aplicación utilizando TypeScript, genera todos los interfaces y clases que requieras.

c) (0.75 puntos) Codifica completamente la clase servicio que es la encargada de realizar todas las operaciones sobre una estructura de datos.

d) (0.75 puntos) Codifica completamente la clase vista que controlará todas las operaciones relativas a la vista.

e) (0.75 puntos) Codifica completamente el controlador que es el encargado de poner en comunicación la vista con el servicio, en este proyecto.

g) (0.50 puntos) Crea un fichero README_PEC3_Ej4.md en el que debes indicar los comandos que se han de ejecutar para transpilar y ejecutar la aplicación. No es obligatorio, pero se valorará positivamente el uso de webpack para que transpile la aplicación completa a un único fichero JavaScript (`bundle.js`). Para ello se ha facilitado dos ficheros webpack de ejemplos (se tienen que modificar y adaptar a nuestro proyecto).

# Formato y fecha de entrega

Tienes que entregar un fichero *.zip, cuyo nombre tiene que seguir este patrón: `loginUOC_PEC3.zip`. Por ejemplo: `dgarciaso_PEC3.zip`. Este fichero comprimido tiene que incluir los siguientes elementos:

La **carpeta PEC3**, y en su interior:

- Un fichero README.md en la raíz de la carpeta con la información indicada.
- La **carpeta oculta `.git`** con el contenido del repositorio local `git`.
- Una carpeta `PEC3_Ej1` con el fichero PEC3_Ej1_respuestas_teoria.md
- Una carpeta `PEC3_Ej2` con:
  - El fichero `code1.ts` modificado siguiendo las peticiones y especificaciones del Ejercicio 2.
  - Un fichero `PEC3_Ej2_respuestas_teoria.md` que contenga enlace a la captura de pantalla y respuestas a todos los enunciados planteados en el Ejercicio2.
- Una carpeta PEC3_Ej3 con los ficheros resultado de haber realizado las tareas del Ejercicio
- Una carpeta PEC3_Ej4 con un fichero README_PEC3_Ej4.md y el resto de los ficheros resultado de haber realizado las tareas del Ejercicio 4.

**Penalizaciones**

- Entrega en otro formato que no sea el especificado (ej. en `zip`): **-0.75 puntos**
- Comprimir archivos dentro del `zip`: -**1 puntos**.
- Para cada ejercicio/apartado donde no se respete la nomenclatura exacta de las carpetas o ficheros indicados (símbolos, minúsculas, mayúsculas, etc.): **-0.75 puntos.**
- La no entrega del repositório local `git` **-3 puntos**

El último día para entregar esta PEC es el **10 de noviembre de 2021** hasta las **23:59**. Cualquier PEC entregada más tarde será penalizada.