Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos:	
	Nombre:	

Trabajo: Creación de una extensión para Visual Studio Code

Objetivos

En este ejercicio proponemos crear una extensión para el conocidísimo editor de texto de Microsoft: **Visual Studio Code**. Esta extensión va a hacer una cosa muy sencilla: insertar una línea en blanco cada *N* líneas.

https://code.visualstudio.com/

El objetivo es **practicar la programación en Javascript** (y conocer de paso TypeScript) y aplicarlo de manera novedosa. De manera paralela, vamos a trabajar con otras tecnologías modernas muy usadas hoy en día en el desarrollo web, tales como generadores y gestores de paquetes.

Descripción

VS Code se está convirtiendo en el editor de facto en el desarrollo web. Este *software* puede ser ampliado mediante las llamadas **extensiones**. Y lo mejor de todo es que estas extensiones pueden escribirse en JavaScript o TypeScript. Nosotros utilizaremos este último lenguaje.

Accede a más información sobre las extensiones:
https://code.visualstudio.com/docs/editor/extension-gallery

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos:	
	Nombre:	

TypeScript es un lenguaje basado sobre JavaScript (un superset) y desarrollado también por Microsoft (al igual que Visual Studio Code). Se parece enormemente a JavaScript, ya que solo añade lo justo para que JavaScript sea un lenguaje más seguro. Nosotros vamos a crear una pequeña extensión para Visual Studio Code y lo vamos a hacer en este lenguaje. TypeScript se está imponiendo de manera muy contundente en el mundo del desarrollo web.

Accede a más información sobre TypeScript:

http://www.typescriptlang.org/

No es necesario que conozcas a fondo TypeScript, tan solo una característica: la adición de tipos sobre JavaScript. En esta página tienes también un pequeño resumen del lenguaje: https://github.com/lakshaydulani/typescript-summary.

Accede a más información sobre tipos de JavaScript:

https://www.typescriptlang.org/docs/handbook/basic-types.html

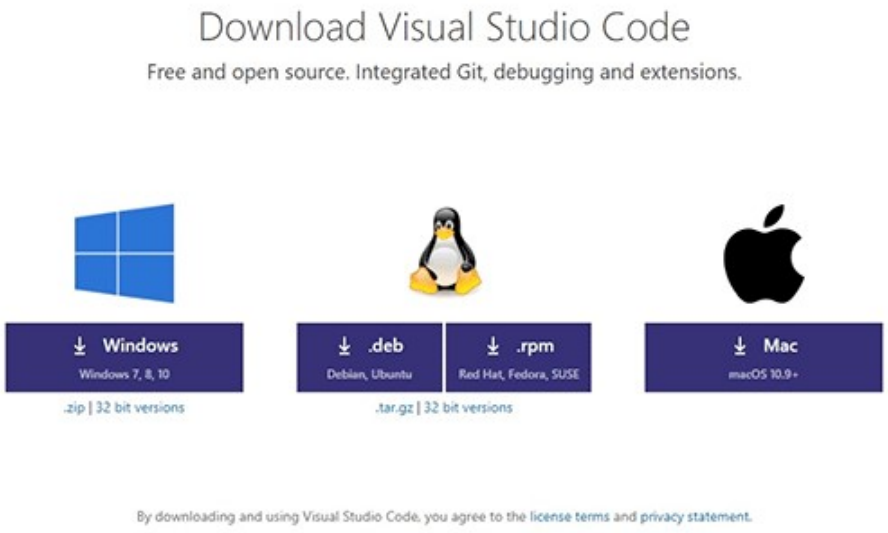
Software a instalar

La actividad simplemente necesita de dos paquetes de *software* fáciles de instalar y disponibles para todos los sistemas operativos.

Visual Studio Code (VS Code)

Como no podía ser de otra manera, vamos a emplear este editor para desarrollar una extensión para sí mismo. Ve a la página de Visual Studio Code (VS Code para los amigos) y descárgatelo: encontrarás una versión específica para tu sistema operativo:

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos:	
	Nombre:	



Accede a la descarga de VS Code:

https://code.visualstudio.com/#alt-downloads

NodeJS

Node (o NodeJS) es un motor JavaScript para el servidor. Se está convirtiendo poco a poco en la tecnología de servidor por excelencia. Además, muchas utilidades ya están siendo distribuidas como paquetes Node a través de su respositorio centralizado de software: NPM.

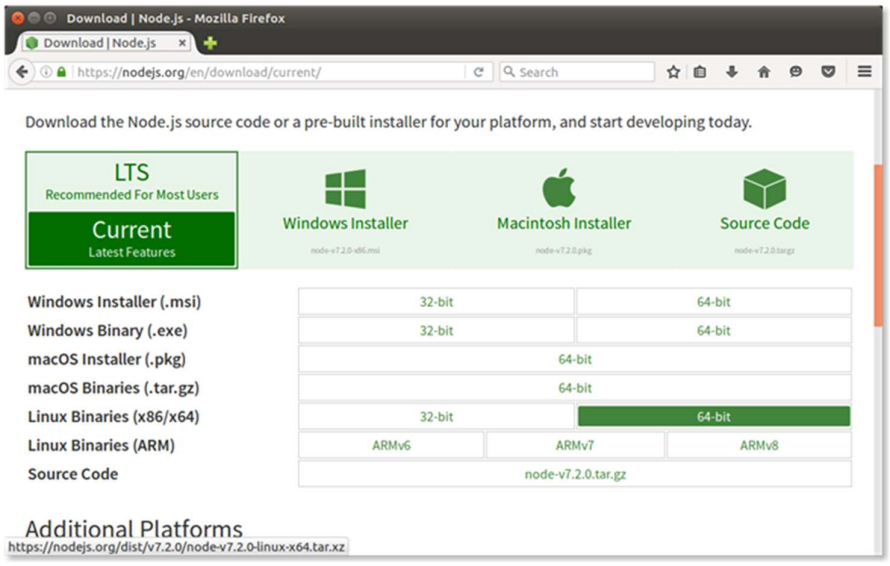
Accede a la descarga de:

Node: https://nodejs.org/en/download/current/

NPM: https://www.npmjs.com/

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos:	
	Nombre:	

Ve a la página de descarga de Node e instala la versión pertinente para tu sistema operativo.



La instalación es directa e inmediata. Una vez completada, ya deberías tener acceso al comando node desde un terminal (`cmd.exe`, `PowerShell` o `Terminal.app`, según tu sistema operativo). Cuando se ejecuta este comando sin opciones, se abre una sesión REPL: un intérprete de comandos JavaScript en tiempo real.

Accede a más información sobre REPL:

https://es.wikipedia.org/wiki/REPL

No vamos a usar esta modalidad de Node, solo se trata de asegurarnos de que funciona. Puedes cerrar el programa pulsando `Control+c` dos veces o cerrar la ventana del terminal.

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos:	
	Nombre:	

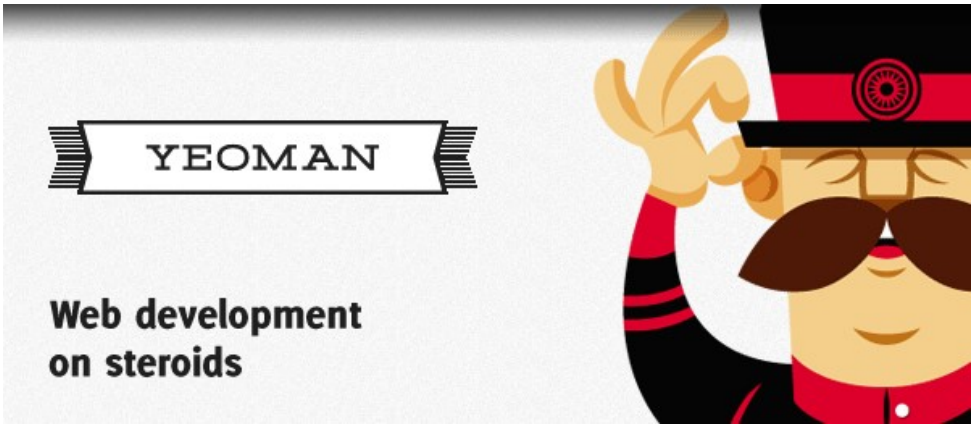
Paquetes NPM

Es necesario instalar además dos utilidades basadas en Node (dos paquetes NPM): Yeoman y Generator-Code.

- Yeoman es una aplicación para generar el esqueleto inicial de un proyecto web. Utiliza a su vez generadores para cada tipo de proyecto (web estáticas, páginas que usen jQuery, programas basados en Node, etc.). Yeoman está escrito, a su vez, en JavaScript.

Accede a más información sobre Yeoman:

https://yeoman.io/



- generator-code es uno de estos generadores para Yeoman y sirve para comenzar el desarrollo de una extensión para Visual Studio Code.

Accede a más información sobre generator-code:

https://www.npmjs.com/package/generator-code

- Para instalar ambos paquetes, ejecuta la siguiente instrucción en un terminal:

`npm install -g yo generator-code typescript.`

En esta línea, el comando `npm` instala de manera global (`-g`) los siguientes elementos:

- El comando `yo` (que no es otra cosa que Yeoman).
- El generador `generator-code`, un generador para Yeoman que sirve para crear el esqueleto inicial de extensiones para VS Code.
- El compilador (o transpilador) de TypeScript:
  - https://en.wikipedia.org/wiki/Source-to-source_compiler

Ejecución de la actividad

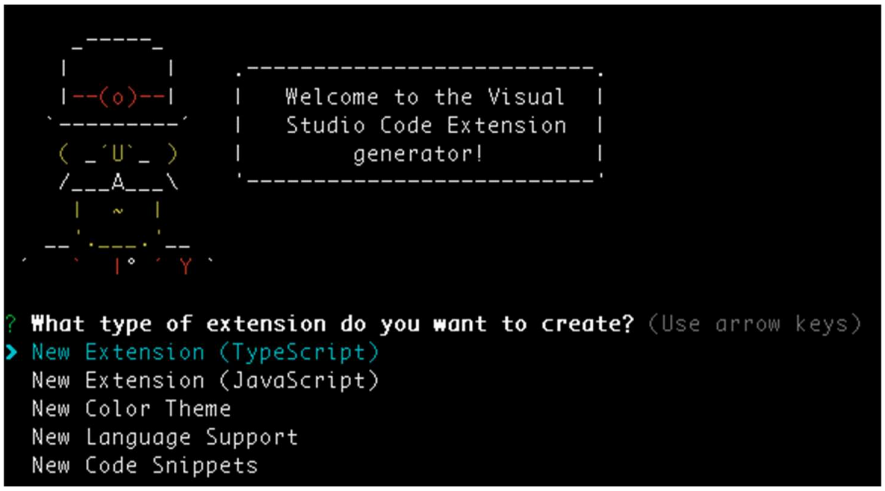
Generación del esqueleto del proyecto

Para empezar, genera el esqueleto del proyecto con Yeoman (comando `yo`, que deberías ya tener a tu disposición por el simple hecho de haber instalado el paquete `yo`).

```
yo code
```

Con la opción `code` le estamos indicando a Yeoman que queremos usar el generador de extensiones para VS Code (que hemos instalado previamente mediante el paquete `generator-code`). Al ejecutar este comando, Yeoman nos hará varias preguntas para configurar nuestro proyecto.

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos:	
	Nombre:	



Elige TypeScript como tipo de lenguaje de desarrollo: `New Extension (TypeScript)`. Luego dale un nombre a la extensión (`Line Gapper`) y un identificador (`gapline`). La opción de `publisher name` es por si vamos a publicar la extensión en el repositorio de extensiones de VS Code, pero nosotros no vamos a hacerlo, así que podéis elegir un nombre cualquiera. Por último, cuando se nos pregunte si queremos iniciar un repositorio Git, decimos que no (Git se utiliza para control de versiones, que no vamos a utilizar).

Accede a más información sobre Git:

https://git-scm.com/

Todas estas opciones acaban registradas en el fichero `package.json`. Si se te preguntara algo más, puedes elegir la opción por defecto y pulsar intro.

Además del fichero `package.json`, Yeoman también habrá generado una carpeta de trabajo con otros archivos y carpetas. Esta es la carpeta de nuestro proyecto.

Apertura del proyecto con VS Code

Ejecuta VS Code y dile que quieres abrir la carpeta que acabamos de crear. Nada más hacerlo, VS Code la «escaneará» y sabrá inmediatamente de qué tipo de desarrollo se trata (una extensión, en nuestro caso). Incluso es posible que nos proponga instalar ciertos componentes apropiados. Dile que sí a todo. Si finalmente instala algo, es mejor que reinicies el editor (y vuelve a abrir la carpeta del proyecto).

Código de la extensión

El código de la extensión se encuentra en el fichero `extension.ts` dentro de la carpeta `src`. Inicialmente, el generador de Yeoman nos pone un código genérico, un «¡Hola Mundo!». Este ejemplo hace precisamente eso: cuando se ejecuta la extensión, como luego veremos, sale un sencillo mensaje mostrando esta bien conocida frase.
Para añadir la funcionalidad que se solicita en este ejercicio, solo tienes que copiar y pegar el siguiente texto (sustituyendo el del ejemplo):

```
'use strict';
import * as vscode from 'vscode';

export function activate(context: vscode.ExtensionContext) {
    let disposable = vscode.commands.registerCommand('extension.gapl
ine', () => {
        var editor = vscode.window.activeTextEditor;
        if (!editor) { return; }
        var selection = editor.selection;
        var text = editor.document.getText(selection);
        vscode.window.showInputBox({ prompt: 'Lineas?' }).then(value
=> {
            let numberOfLines = +value;
            var textInChunks: Array<string> = [];
            text.split('\n').forEach((currentLine: string, lineIndex
) => {
                textInChunks.push(currentLine);
                if ((lineIndex+1) % numberOfLines === 0) textInChunk
s.push('');
            });
            text = textInChunks.join('\n');
            editor.edit((editBuilder) => {
                var range = new vscode.Range(
                    selection.start.line, 0,
                    selection.end.line,
                    editor.document.lineAt(selection.end.line).text.
length
```

```
                );
                editBuilder.replace(range, text);
            });
        })
    });
    context.subscriptions.push(disposable);
}

export function deactivate() { }
```

Asegúrate de que el archivo `tsconfig.json` tiene la opción `strict` marcada como `false`. Si no estuviera, añádela:

```
"compilerOptions": { …, "strict": false,
```

Ejercicio inicial:

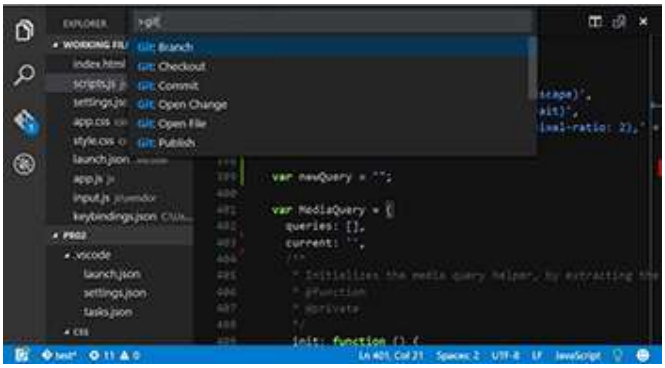
- Comenta cada línea del código anterior teniendo en cuenta el propósito descrito al principio de este enunciado.
- Busca en el API de las extensiones de VS Code (https://code.visualstudio.com/api/references/vscode-api) y averigua para qué sirve las funciones `activate`, `deactivate` y el resto de funciones y propiedades específicas de las extensiones de VS Code.
- Es importante que entiendas cada línea de código. Si tienes alguna dificultad, puedes preguntar en el foro *ad hoc* para la resolución del ejercicio. ¿Qué elementos específicos de TypeScript identificas? Recuerda que el profesor está para ayudarte y que esta ayuda continua en el foro.

Segundo ejercicio:

- Comprueba que tu extensión funciona. Para ello, primero asegúrate de que sustituyes todos los `sayHello` (que inserta por defecto el generador `code` que hemos aplicado con Yeoman) por el nombre de la función que hemos usado (`gapline` en nuestro caso) en el fichero `package.json`. Como hemos comentado antes, este fichero contiene las opciones e información básica de nuestra

extensión y, entre otras cosas, cómo ejecutarla. Es especialmente importante las secciones `activationEvents` y `commands`. Vigila que el nombre del comando sea el correcto.

- Para ejecutar la extensión, pulsa **F5** (o la tecla función + F5, dependiendo de tu teclado y sistema) o selecciona la opción de menú `Start debugging` del menú `Debug`. Verás que se abre una nueva ventana de VS Code (que ya tiene tu extensión precargada). Abre un fichero de texto cualquiera, selecciona su contenido y pulsa **control/comando + shift + P**. Se abrirá el visor de comandos de VS Code, el command palette:
  - https://code.visualstudio.com/docs/getstarted/userinterface#_command-palette
- Busca el nombre de tu extensión (tendrá el mismo nombre que hayas puesto en el campo `contributes -> commands -> title` en el fichero `package.json` o el que le diste cuando la creaste con el generador de Yeoman).



Debería aparecer un nuevo cuadro de diálogo preguntándote cada cuántas líneas quieres insertar: una línea en blanco (o el texto que hayas puesto en la propiedad `prompt` del código anterior). Selecciona un número apropiado y pulsa intro. Verás cómo el texto seleccionado es sustituido por uno nuevo, pero que contiene líneas en blanco.

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos:	
	Nombre:	

Rúbrica

- La extensión funciona correctamente y se demuestra mediante captura de pantalla. Peso: 40%
- Cada paso y línea de código está correctamente justificada y narrada en el documento final. Peso: 30%
- La presentación final del ejercicio es limpia, cuidada y acorde a un nivel de master universitario. Peso: 30%

Entrega

Se valorará una correcta narración de la ejecución del ejercicio paso a paso (instalación de componentes, edición de los ficheros, ejecución de la extensión con un ejemplo, etc.). Puedes incluir una captura de pantalla demostrando que la extensión funciona correctamente. Contesta a todas las preguntas de manera ordenada y limpia. Se trata de un ejercicio propio de un master universitario y se tendrá en cuenta una correcta explicación y cuidado en la presentación. Cuida la ortografía y los nombres propios de las tecnologías usadas (no es *vs Code*, ni *vs code*, sino *VS Code*). Presenta tu ejercicio en la plataforma en PDF o en un HTML inline (con todo incrustado: imágenes, estilos, etc.). Recomiendo el uso de Markdown como estándar industrial para registrar información de tipo técnica.