Universidad Autónoma de Madrid

Departamento de Informática

Proyecto de Sistemas Informáticos
Práctica - 2

Roberto Marabini
Alejandro Bellogín
Jose A. Macías
Alejandro Moya

Registro de Cambios

Versión[1]	**Fecha**	**Autor**	**Descripción**
3.0	01.11.2024	AB	Adaptación para el curso 2024-25
3.1	17.02.2025	JAMI	Concreción de aspectos varios para este curso, así como de la rúbrica de corrección
4.0	01.09.2025	AB	Adaptación para el curso 2025-26
4.1	30.09.2025	RM	Eliminación de algunas referencias a la practica del año pasado
4.2	09.02.2026	AM	Modificación menor sobre los test a pedir para obtener el Apto

[1]La asignación de versiones se realizan mediante 2 números $X.Y$. Cambios en Y indican aclaraciones, descripciones más detalladas de algún punto o traducciones. Cambios en X indican modificaciones más profundas que o bien varían el material suministrado o el contenido de la práctica.

Índice

1. Objetivo

En esta práctica aprenderás a crear una aplicación usando *Vue.js*. Veremos primero los conceptos fundamentales de *Vue.js* así como la anatomía de las aplicaciones creadas con este framework. Seguidamente aprenderás a crear una aplicación CRUD paso a paso capaz de conectarse a un API REST. Cuando finalice la práctica, habrás aprendido a crear **componentes**, **eventos**, **métodos**, **propiedades computadas** y **formularios**, además de saber gestionar el **estado de los componentes**, acceder a **APIs** externos a la aplicación y gestionar el **enrutado**.

2. Introducción

Vue.js es un entorno de código abierto desarrollado en JavaScript que se encarga de proporcionar las herramientas necesarias para la creación de las vistas en una arquitectura MVC.

Para esta práctica necesitarás tener instalado tanto `Node` como `npm`. Nota: `npm` se instala automáticamente al instalar `Node`. Las versiones a utilizar serán la **18.3** o superior de Node (comprobar con `node -v` y `npm -v`) y la **3.4** de Vue.

Listado 1: Cómo instalar `npm` desde la línea de comandos (no necesario en los laboratorios). La flecha roja indica un salto de línea introducido para mejorar la visibilidad del listado.

```
    # install node (and npm)
    # node is already installed in the lab computers (linux partition)
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash &&
    ↪ sudo apt-get install -y nodejs
```

En esta práctica se creará una aplicación de gestión con *Vue.js*. La aplicación permitirá gestionar los datos de diferentes personas. Veremos **cómo instalar *Vue.js*** y cómo configurarlo. Seguidamente verás cuál es la **estructura de un archivo *Vue.js***. Luego aprenderás a crear componentes, propiedades, métodos, eventos y diferentes tipos de sentencias de *Vue.js*.

Para crear la interfaz de la aplicación, usaremos tablas y formularios, agregando

campos a estos últimos junto con las validaciones pertinentes. Por último, también aprenderás a desplegar una aplicación creada con *Vue.js* en *Render.com* y cómo conectarla a un API, tanto a una externa como a una creada por vosotros en Django.

En `https://tutorialvue.onrender.com/` puedes ver el resultado de la aplicación que vas a crear.

3. Organización de la Práctica

La práctica se ha dividido en dos partes que se describen por separado en dos ficheros pdf adicionales. En la primera (parte 2.1) se crea la aplicación con toda la funcionalidad requerida, incluyendo tests, excepto la habilidad de persistir los datos y la existencia de varias páginas. En la segunda parte (2.2) la aplicación accederá a un API `REST` creada en *Django* que permitirá la persistencia de los datos, también se verá cómo almacenar información de sesión y cómo se puede navegar entre diversas URLs sin necesidad de contactar con el servidor.

4. Asignación de tiempo para las diferentes tareas

Aproximadamente se espera que la mitad del esfuerzo se dedique a la primera parte y la otra mitad a la segunda parte.

5. Material a entregar al finalizar la práctica

1. Se debe indicar claramente, y en el momento de la entrega en *Moodle*, las URLs donde están desplegados el proyecto de *Vue.js* y el proyecto de *Django* en *Render.com* así como la dirección en la que se encuentra la base de datos utilizada, la cual debe estar albergada en `https://neon.tech`. Esta información debe aparecer en un fichero llamado `.env` situado en la raíz del proyecto de *Django*. El fichero `settings.py` entregado con

el proyecto de *Django* debe tener las variables `CORS_ORIGIN_WHITELIST` (*Vue.js*) y `ALLOWED_HOSTS` (*Django*) configuradas adecuadamente de forma que se pueda desplegar tanto localmente como en *Render.com* sin necesidad de modificarlo (pudiendo depender, como se indica en el enunciado, de otras variables de entorno). Finalmente, la interfaz de administración de *Django* debe ser accesible mediante el usuario y contraseña habituales (`alumnodb`).

2. En esta práctica se crean dos proyectos, uno en *Vue.js* y otro en *Django* llamados `tutorial-vue` y `persona` respectivamente. Estos dos proyectos deben crearse **dentro del mismo repositorio** Git en distintas carpetas. Para la entrega en *Moodle*, subid el fichero obtenido al ejecutar el comando `zip -r ../psi-lab2.zip .git` desde el directorio raíz del repositorio, teniendo cuidado de haber incluido los ficheros `.env` para que funcione al descomprimir. Recordad que hay que añadir (*git add*) y enviar (*git commit*) los ficheros al repositorio de *git* antes de ejecutar el comando. Comprobad que el contenido del fichero zip es correcto ejecutando la orden: `cd /tmp; rm -rf tmpDir; unzip psi-lab2.zip; git clone . tmpDir; ls tmpDir`. El directorio `tmpDir` contendrá los dos proyectos.

   La estructura que deberíais tener es la siguiente:

```
psi-lab2/
├── tutorial-vue/
│   ├── cypress/
│   ├── src/
│   └── ...
└── persona/
    ├── persona/
    ├── api/
    ├── manage.py
    └── ...
```

3. Se debe prestar especial atención a las variables de entornos, importándolas correctamente desde los respectivos ficheros `.env`.

4. No subáis, ni añadáis al repositorio el entorno virtual de python (*Django*) ni el directorio `node_modules` (*Vue.js*) puesto que sólo son válidos en el ordenador en el que se han creado.

6. Criterios de evaluación

Para aprobar con APTO es necesario satisfacer TODOS los criterios siguientes:

- El fichero subido a *Moodle* contiene un repositorio de git.
- El código se ha guardado en un repositorio de git y este repositorio es privado.
- El "log" del repositorio muestra "commits" semanales de todos los miembros del grupo.
- Los datos de la aplicación deben persistirse en una base de datos PostgreSQL almacenada en `https://neon.tech`.
- La aplicación dedicada a la gestión de personas (proyectos `tutorial-vue` y `persona`) debe estar desplegada en *Render.com*, y debe ser accesible a través de las URL suministradas por los estudiantes. Si no se entregan las dos URLs, se calificará automáticamente como NO APTO. Las URLs deben seguir el formato: `http://práctica_proyecto_pareja_grupo_año_versión.onrender.com` (en este caso: práctica = 2, proyecto = {tutorial-vue, persona}; se añade el campo versión para permitir a los estudiantes añadir un sufijo arbitrario).
- *Devtools* debe estar accesible en la aplicación desplegada en *Render.com*. Esta medida facilita la corrección de vuestro trabajo pero no es una buena práctica para aplicaciones web desplegadas para producción.
- La aplicación desplegada en *Render.com* debe pasar los test de cypress que comienzan con "dynamic" ejecutados sobre la aplicación desplegada en *Render.com*.

- El código creado para esta práctica debe ejecutarse correctamente usando las versiones de los diferentes programas que se describen en la memoria. En particular Python 3.11, *Django* 5.2.2 y *Vue.js* 3.4.

NOTA: En caso de realizar la entrega fuera de plazo, se substraerá un punto de la nota del examen por cada día (o fracción) de retraso en la entrega.