2ºDAW | CURSO 2023-24

Proyecto Café

IGNACIO REVILLA ITALIYANKINA
KEVIN BRUNO GEISENHOF SÁEZ

1. Introducción

El Proyecto Café fue una iniciativa impulsada por la jefatura de estudios para poder ayudar a la docencia a registrar sus horarios de trabajo de manera más intuitiva y menos costosa a nivel manual: tras tiempo recurriendo a medios como el papel y el bolígrafo, ahora por fin se podrá usar una aplicación sencilla de unos clics en la que un usuario simplemente inicia sesión y podrá manipular o registrar su horario como desee.

Esta aplicación se desarrolló con la intención de poder mejorar la calidad de vida de los docentes en cuanto a horarios: informará mejor a estos de qué módulos pueden impartir además de, obviamente, poder eliminar, editar o añadir alguno. También mostrará que módulos ya se tenían asignados de antes.

Cabe destacar que los jefes de departamento podrán ver con facilidad los horarios de los profesores de su propio departamento, además de rellenar / modificar el suyo propio.

Es casi el mismo caso para la jefatura de estudios, que podrá ver **todos** los horarios según el departamento, además de rellenar / modificar el suyo propio.

2. Diseño e implementación

La arquitectura que se ha seguido para este proyecto ha sido la general seguida en el patrón de diseño Modelo-Vista-Controlador (MVC), lo que supone una capa de acceso a datos para las vistas, una capa de lógica para los controladores, y una capa de almacenamiento de datos para los modelos.

Los componentes a destacar son:

- **Laravel Sail.** Entorno de desarrollo local que simplifica la gestión de contenedores Docker.
- **API propia.** Conjunto de rutas fijas y personalizadas que sirven para conseguir datos. Devuelven datos para luego ser usados en las vistas.

Las tecnologías usadas fueron:

- **Laravel Framework.** Framework usado por Laravel para iniciar y manejar sus proyectos. Se basa en el patrón de diseño MVC.
- **Docker.** Gestor de contenedores para aplicaciones. Permite hacer que las aplicaciones se compacten en una imagen con todo lo necesario para funcionar.
- **MariaDB.** Gestor de bases de datos derivado de MySQL que usa el lenguaje SQL.

En cuanto a seguridad, se han usado los siguientes recursos:

- **Laravel Sanctum.** Sistema de autenticación de usuarios basado en el uso de *tokens* (cadenas de texto encriptadas que identifican a un usuario), que permiten y niegan ciertas acciones por parte de usuarios. Esto ayuda a limitar que o que no puede hacer alguien en nuestra aplicación. Estos tokens previenes ataques XSS.
- **Eloquent.** Eloquent viene incorporado con Laravel y permite usar consultas SQL preparadas, que previene la inyección SQL.
- **Blade.** Motor de plantillas de Laravel. Blade, al escapar de cualquier salida de variables, previene la inyección de scripts maliciosos a nivel de cliente.

3. Problemas encontrados

Al desarrollar la aplicación, se han encontrado ciertos problemas que podrían ser comunes en una aplicación que usa una API para obtener datos:

- **Lenguaje asíncrono vs. lenguaje síncrono.** JavaScript, por defecto, usa un lenguaje síncrono. Sin embargo, *ciertas acciones requieren que sean asíncronas, como usar la Fetch API.* La Fetch API es la API usada para poder sacar datos de nuestra propia API.

  Solución: realizar las acciones que sean necesarias dentro de la función “fetch”, dentro de un contexto asíncrono. Un truco común es pasar los datos como parámetro a una función ya construida de antes.

- **Ambigüedad en sintaxis y funcionalidad en lenguajes.** Ciertas veces, algunas propiedades o funciones de JavaScript reclaman hacer una misma operación a pesar de que sus funcionalidades conducen a un fin diferente. *Ejemplo: las propiedades “defaultSelected” y “selected” de un elemento “select” HTML pueden ser supuestamente usadas para cambiar el estado del atributo “selected” de dicho HTML, pero “defaultSelected” para ser la única opción para este fin.*

  Solución: simplemente, probar que opción es más efectiva y funciones, usando mensajes de consola para ver los resultados.

- **Sistema de seguridad vulnerable.** La razón especifica por la cual no se pudo implementar seguridad fue debido al cambio de laravel/ui a Sanctum a mitad del proyecto, lo cual afecto en cómo se tenía en mente la seguridad de la aplicación.

  Solución: a pesar de la complejidad, se puede combinar políticas, roles y middleware de manera manual para conseguir que la seguridad mejore.

4. Trabajo futuro y conclusiones

Algunas posibles mejoras a implementar:

- **Mejorar eficiencia de API.** A pesar de que la API está protegida con tokens de Sanctum, es escalable y compatible con más lenguajes, esto hace que las consultas puedan llegar a apilarse y la complejidad aumente para implementarla y mantenerla.
- **Uso de consultas SQL normales.** Derivado del anterior punto, se debería considerar usar consultas SQL normales si la complejidad del proyecto llega a ser un impedimento grande para que la API funcione eficientemente.

Ciertas lecciones aprendidas han sido:

- **Tener un back-end sólido antes de un front-end.** Cuando el front-end tiene un problema, normalmente no afecta al back-end, pero si el back-end lo tiene, en este caso si suele afectar al front-end. Tener un back-end constante evita problemas de obtención de datos para el front-end.
- **Trabajar front-end y back-end por separado vs. Trabajar cada uno conjuntamente.** Existen ciertas ventajas y desventajas de trabajar cada parte de una manera u otra. La primera permite que cada persona sea más eficiente y rápida en su especialización, pero la segunda asegura que todo sea más controlado y constante cuando se vaya a probar.

Las conclusiones que se derivan de esta experiencia han sido:

- Usar una API es una manera segura y escalable para poder construir una aplicación.
- Es importante tener en cuenta de qué manera se va a trabajar en equipo para luego afrontar las dificultades que puedan surgir.
- Afrontar que un lenguaje puede tener sus peculiaridades
- Tener un diseño, luego un back-end y, finalmente, un front-end

5. Documentación

- **Acceso al proyecto en público:** https://cafe-odiy.onrender.com/
- **Enlace al repositorio:** https://github.com/KamaliKevin/proyecto-cafe
- **Licencia:**

FIN