TRABAJO DE FIN DE CICLO:

TFCBLOG

PABLO HERRANZ CANO

DESARROLLO DE APLICACIONES WEB

ÍNDICE

1. Abstract 3
2. Justificación del proyecto 4
3. Objetivos 6
4. Desarrollo 7
    A. Revisión bibliográfica 7
    B. Materiales y métodos 8
    C. Resultados y análisis 13
5. Conclusiones 14
6. Líneas de investigación futuras 16
7. Bibliografía 17

1. ABSTRACT

Resumen

Este proyecto trata de transformar una idea, como la creación de un blog, en un producto accesible desde cualquier parte del mundo. Se desarrolla una aplicación Web que dispone de registro de usuarios y control de sesiones, peticiones a la base de datos mediante las funciones CRUD, elaboración de *posts* y de categorías, y la conexión entre ambos, permisos de usuarios, un sistema de comentarios que permite interactuar con cada uno de los *posts* por separado, y también un formulario de contacto que nos permitirá atender las sugerencias de los posibles usuarios que generarán tráfico Web a nuestro sitio. En definitiva, una aplicación creada completamente desde cero, desarrollando tanto la parte *Front End* como *Back End* con lenguajes aprendidos en clase: *HTML, CSS, JavaScript, PHP* y *MySQL*. Es un auténtico ejercicio de tolerancia a la frustración en el que se hace frente a todo tipo de problemas provenientes de una aplicación Web como ésta, así como el ejercicio final de aprendizaje imprescindible para conseguir las aptitudes necesarias para el mundo laboral del desarrollo de aplicaciones Web y de la programación en general.

Abstract

This project tries to transform an idea, such as creating a blog, into a product accessible from anywhere in the world. A Website that includes a user registration and a session control system, requests to the database through the CRUD functions, publishing of *posts* and categories, and the connection between both, user permissions, a comments system that allows us to interact with each of the *posts* separately, and also a contact form that will allow us to respond to the suggestions of all the potential users that will generate Web traffic to our Website. Basically, an application created completely from scratch, developing both the Front End and Back End with languages learned in class: HTML, CSS, JavaScript, PHP and MySQL. It is a true exercise in frustration toleance in which all kinds of problems arising from a Web application like this are faced, as well as the final learning exercise essential to achieve the necessary skills for the world of development and programming in general.

2. JUSTIFICACIÓN DEL PROYECTO

En este apartado voy a intentar justificar de manera sencilla el porqué de la elección de hacer un blog.

A día de hoy, en pleno siglo XXI, existen muchas formas de crear un blog o un sitio *Web*. Los denominados CMS (*Content Management System*, o Sistema de Gestión de Contenido) como pueden ser *Wordpress*, *WebFlow*, *Joomla*, *Drupal*, *Prestashop*, *Moodle* y un largo etcétera, nos permiten hacer una página *Web* como si de un *Word* se tratara.

Gente que no ha estudiado ni sabe nada de programación sería capaz de crear una página *Web* en una de esas plataformas, obteniendo unos resultados más que convincentes. Y, aunque tiene una cierta complejidad la creación de un sitio *Web* con un *CMS*, todo el proceso, desde la idea hasta la publicación, resulta sencillo en términos generales.

La cantidad de *templates* disponibles en internet para darle un toque personal a tu *Web* es inmensa. También el número de *plugins* disponibles es muy amplio. Si tienes una necesidad que quieras implementar en tu página *Web*, posiblemente ya exista *online*, porque seguramente no serás ni el primero ni el último en haber pensado en ello.

Además, el uso de este tipo de servicios nos permite ahorrar tiempo en el desarrollo y dedicar más tiempo a otros aspectos del que negocio que tengamos entre manos.

Por lo general, el uso de un sistema de gestión de contenidos (*CMS*) tiene muchísimas ventajas:

- No es necesario tener una formación en desarrollo Web ni saber de programación.
- El desarrollo y el tiempo de implementación son procesos bastante rápidos, ya que muchas de las funcionalidades están listas para usar.
- Los CMS de código abierto son gratuitos para descargar e instalar. El único gasto sería el del hosting y el dominio.

- La mayoría de los sistemas permiten la implementación rápida de formularios, encuestas, calendarios, mapas, etc.
- El desarrollo con un CMS es definitivamente menos costoso que uno desde cero.
- La mayoría de los CMS están desarrollados para ser fáciles de mantener y actualizar.
- La comunidad de desarrollo detrás de los CMS más conocidos es enorme, lo que les da robustez y fiabilidad.
- Las posibilidades de personalización son enormes.

¿Pero entonces... por qué hacer un blog desde cero si ya existen todos estos servicios que te permiten hacer un blog de forma rápida y sencilla?

La respuesta es muy simple: porque he estudiado dos años, junto a otras 15 personas, el módulo de Desarrollo de Aplicaciones Web. Así que ¿de qué me (nos) serviría haber hecho esto si a la hora de la verdad utilizamos uno de esos servicios?

No podemos pasar por alto que todos esos *CMS* de los que hablo los ha creado alguien que ha dedicado su tiempo en facilitar la vida de los demás. A mí, personalmente, no me gusta mucho lo fácil, así que prefiero aprender a **hacer** un blog desde cero, y practicar todo lo aprendido durante estos dos años, que aprender a **instalar y mantener (o actualizar)** un Wordpress, por ejemplo.

Además, he hablado de las ventajas, que son muchas, pero no de las desventajas de usar este tipo de servicios. Y aunque son menos, también las hay, y a lo mejor son más importantes a tener en cuenta:

- En cuestiones de seguridad, no es que sean lo mejor de Internet. Si hay algún agujero de seguridad en el *CMS*, también lo tendrá toda página Web que hayas creado con ese *CMS*. En este caso, tú, a la hora de descargarte e instalar uno de estos sistemas estás aceptando cualquier riesgo proveniente de la utilización de ese software. El desastre puede ser monumental.
- La utilización de un gestor de contenido inhibe tu habilidad de programar de forma estructurada. Básicamente porque vas a ver muchísimo menos código que si no lo utilizas.

- Muchos de estos sistemas introducen *bloatware* que puede hacer que se ralentice la carga de tu página *Web*.

En definitiva, aunque creo que es una tecnología muy útil y que ayuda a mucha gente, si hubiera querido simplemente actualizar un *Wordpress*, me habría hecho algún curso online de utilización de un *Wordpress*, y no un FP de dos años de Desarrollo de Aplicaciones Web.

3. OBJETIVOS

Con este proyecto los objetivos que se pretenden alcanzar son:

- Ser capaz de desarrollar todo el *Front End* de una aplicación desde cero, sin utilizar plantillas Web.
- Ser capaz de desarrollar todo el *Back End* de una aplicación desde la base.
- Ser capaz de utilizar de forma eficiente y correcta un sistema de control de versiones, como *Git* o como *Bitbucket*.
- Desarrollar una visión analítica que me permita enfrentarme a problemas reales para, de este modo, ser capaz de solucionarlos.
- Ganar soltura a la hora de escribir código.
- Ser capaz de encontrar la información necesaria para seguir avanzando.
- Vencer a la frustración que, por seguro, aparecerá en algún momento del proyecto.
- Descubrir por qué camino seguir una vez lo termine (*Front* o *Back*).
- Aprender, a base de prueba y error, a crear un blog, o similar, totalmente funcional.
- Ser capaz de escribir un código legible, modularizándolo al máximo posible para conseguir estructurarlo de una manera eficiente.

4. DESARROLLO

4. A. Revisión bibliográfica, fundamentación teórica

A día de hoy, lo difícil es no encontrarse, en los dos ó tres primeros minutos navegando por internet, blogs o foros de discusión de una temática particular:

- Si lo que buscamos es información sobre **tecnología**, blogs como Xataka.com, HacksDigitales.com o Genbeta.

- Si queremos blogs de **cocina**, Webosfritos.es, eladerezo.com, o cocinarpara2.com son muy buenas opciones.

- En el mundo del **motor** tenemos motorpasion.com, autonocion.com o actualidadmotor.com, entre otros.

- Y en el mundo del **off-topic**, de todo un poco, podemos encontrarnos blogs tan arcaicos como emocionantes como el de montonesdepapeles.com o foros de discusión como el mundialmente conocido forocoches.com, que es de todo menos un foro de coches.

Y no sólo hay blogs de estas cuatro temáticas, hay blogs de todo tipo de contenido que uno pueda llegar a pensar. Y, al igual que la temática, también los diseños difieren un poco, o mucho, entre sí. Incluso las tecnologías utilizadas son diferentes dependiendo de si accedes a uno u otro. En definitiva, hay todo un mundo de oportunidades en la red.

4. B. Materiales y métodos

Una vez vistos los objetivos que se pretenden conseguir, conviene explicar un poco cómo se pretende llevarlos a cabo de manera satisfactoria y a través de qué tecnologías se pretende llegar a la meta.

Las tecnologías elegidas son *HTML*, *CSS*, *PHP*, *JavaScript* (con JQuery*)* y *MySQL*. De este modo, este trabajo servirá como un repaso general de lo visto en clase durante los dos años que ha durado el ciclo formativo de grado superior.

HTML y CSS para generar el diseño que tendrá la Web. PHP y JavaScript para darle dinamismo y poder interactuar con ella haciendo una serie de llamadas CRUD a la base de datos a través de comandos MySQL.

Funcionamiento y estructura

Lo primero de todo, y menos importante, para saber en qué Web estamos en todo momento, se muestra un *favicon* (Imagen 0) con mis iniciales.

Imagen 0. *Favicon*.

El proyecto en sí es extenso, pero un poco más simple de lo que parece con tanto archivo. Se divide en tres partes: por un lado tenemos la parte del *Front End*, por otra la del *Back End*, y por última, la base de datos.

Por la parte del ***Front***, la carpeta "*/assets*" (Imagen 1) es la que contiene todo lo relacionado con el diseño. Hay que diferenciar en dos estancias del blog: por un lado contamos con lo que el usuario que acaba de llegar puede ver y, por otro, el panel de administrador. Se utiliza el tipo de letra *Ubuntu* en toda la Web.

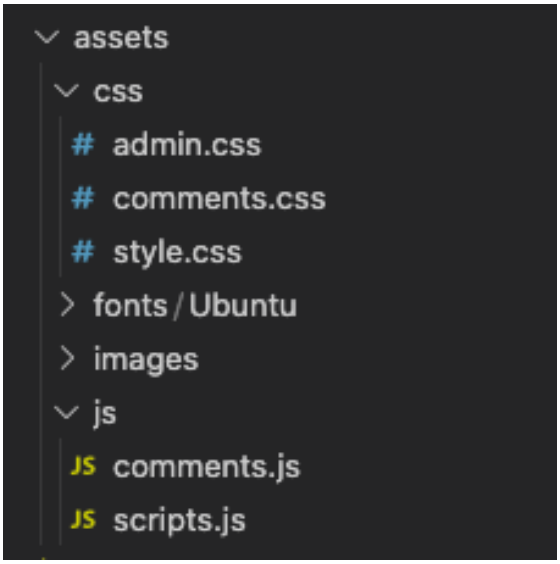

Imagen 1. Árbol carpeta "/assets".

En el home de la Web (Imagen 2), que es lo los usuarios ven según llegan, nos encontramos:

- ***Header***: un logo a la izquierda y un *navbar* a la derecha con un *dropdown* que se verá al pasar el ratón por encima.

- ***Page wrapper***: entre medias del *header* y el *footer*, tenemos el *page wrapper*. Dentro de éste, hay dos partes bien diferenciadas que son el *content* (izquierda, 70%) y el *sidebar* (derecha, 30%). Esta sección varía dependiendo del sitio en el que nos encontremos:
    - En el *index*, el *content* muestra una *preview* con imagen y texto de los diferentes *posts* del blog. El *sidebar* mostrará una pequeña sección "sobre mí", seguido de un buscador, que nos muestra los *posts* que contengan, en el título o en el texto del *post*, la palabra o expresión que le digamos; y de una lista con las diferentes categorías (*topics*) de los que se hablará en el blog.
    - En el *Single post*, que es el diseño de un *post* del blog, se mostrará a la izquierda el contenido del *post*, y a la derecha el *sidebar*, pero aquí desaparecen las secciones "sobre mí" y el buscador para dar paso a una sección que muestra todos los *posts* escritos pertenecientes a esa categoría. Debajo, la parte de las categorías sigue intacta.
    - En las páginas de registro y de *login* (*auth pages*) desaparece el *sidebar* y da paso a un div central con un formulario a rellenar por el usuario.

- **Footer**: en la parte de debajo de la página. Podemos verlo en la Imagen 3. Éste sólo se muestra en el *index* y en el *single post*. Consta de dos partes:
    - Una primera parte superior con información de contacto y un pequeño formulario.
    - Una segunda parte de inferior tamaño, con el copyright, situada en la parte más abajo.

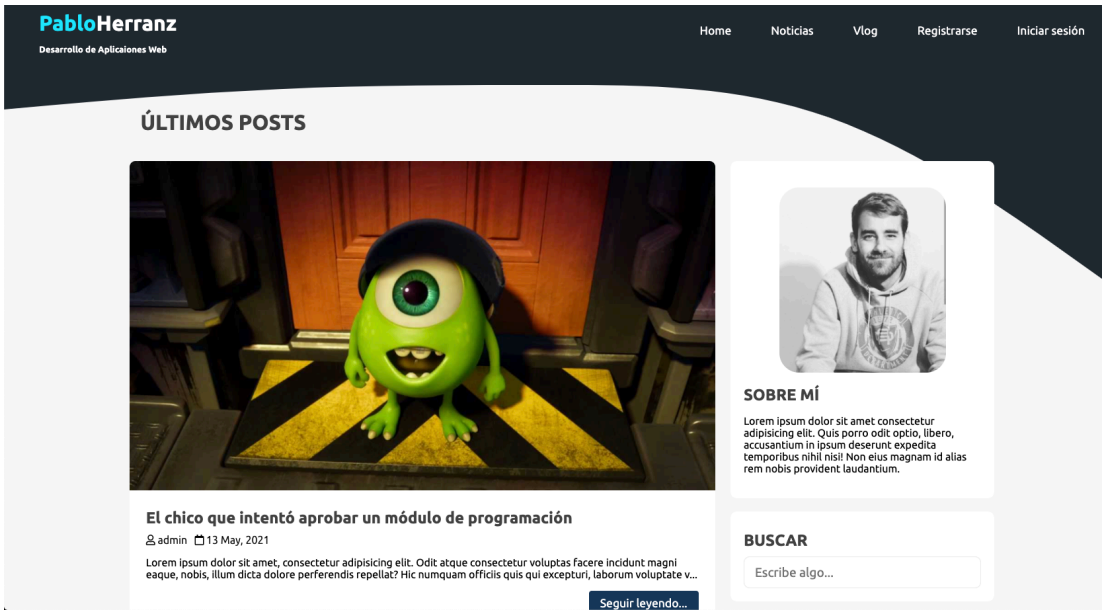

Imagen 2. Home del blog.

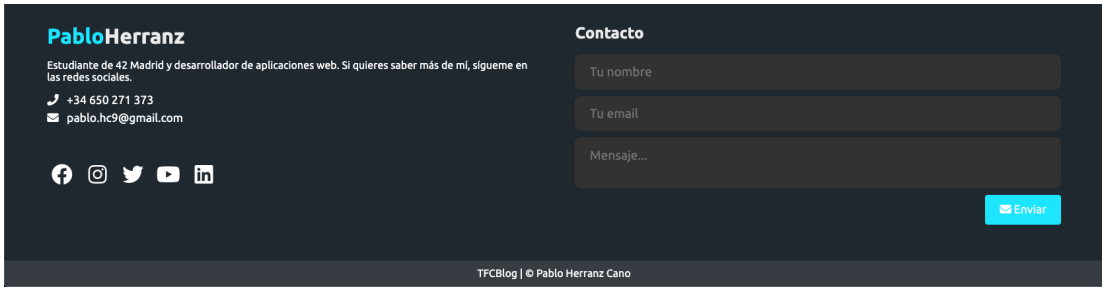

Imagen 2. *Footer*.

En el panel de administrador, tendremos el *sidebar* a la izquierda, y una zona a la derecha donde podremos administrar, mediante el uso de tablas, una cosa u otra dependiendo de dónde nos encontremos:

- Administrar usuarios.
- Administrar topics.
- Administrar posts.
- Administrar comentarios.

- Bandeja de entrada.
- Volver al dashboard, que en este caso sólo muestra un mensaje dándonos la bienvenida al panel de administrador.

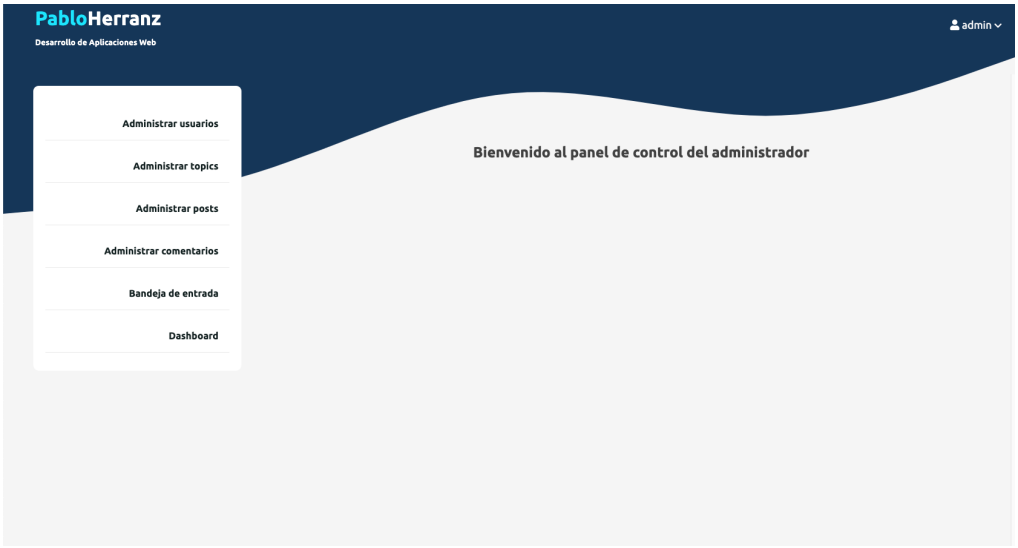

Imagen 3. Panel de adminisrador.

La forma más clara de diferenciar ambas zonas es por el *header*. En el del panel de administrador desaparecen los botones y sólo se puede ver el nombre de usuario con un *dropdown* para hacer *logout*. También hay en ambos un vector en forma de ola que forma parte del *header*. Ambas olas son diferentes, al igual que su color.

En cuanto al **Back**, la cosa se complica un poco. Al igual que todo lo del Front está en "/assets", el cerebro del blog se guarda en su mayoría en la carpeta "/app" (Imagen 4). Ésta contiene en su interior las carpetas "/controllers", "/database", "/helpers", e "/includes".

- **"/controllers"**: encontraremos los archivos con los que controlaremos las diferentes áreas del panel de administración: usuarios, comentarios, posts, topics y emails de contacto.

- **"/database"**: aquí se encuentran los dos archivos principales para poder interactuar con la base de datos. Por un lado, "connection.php" es en donde se encuentran las variables con las que realizaremos la conexión a la base de datos, y las que tendremos que modificar dependiendo de si queremos trabajar en local u online (*deployed*). Por otro, el archivo "db.php" es el cerebro de las funciones CRUD. Están definidas todas las funciones que desempeñarán el

papel de *create, read, update* y *delete*. Asimismo, están definidas funciones que nos permiten hace unas consultas específicas a la base de datos en donde se almacenan los diferentes, comentarios, categorías, *posts* y usuarios.

- **"/helpers"**: como su propio nombre indica, encontramos archivos que nos ayudan a ciertas cosas. Están los tres archivos de validación de *posts*, usuarios y categorías. Se encargan de validar que un registro no deje ningún campo en blanco, que el *captcha*, creado de manera aleatoria en el archivo "captcha.php", sea respondido correctamente, o que cuando estemos creando un *post* no dejemos el texto sin poner. También hay un archivo llamado "formErrors.php" que simplemente muestra cualquier posible mensaje de error en el momento del *login* o registro.

- **"/includes"**: esta carpeta es la encargada de resguardar fragmentos de código que nos permiten la modularización del mismo. Para tener más limpio el código HTML, separamos en diferentes archivos lo que podemos encontrar en esta capeta, como el *header*, o *adminHeader*, dependiendo de dónde nos encontremos en la página, el *footer*, o los mensajes de error o de éxito que saldrán dependiendo de qué acción hayamos llevado a cabo (loguearnos, editar un usuario o un *post*, o un *topic*... o eliminar un comentario, por ejemplo).

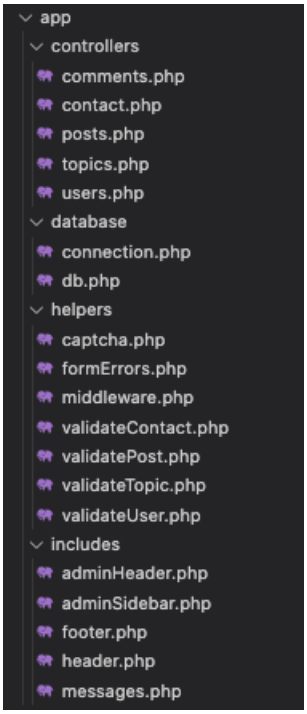

Imagen 4. Árbol carpeta "/app".

La **base de datos** (Imagen 5) contiene las tablas *coments*, *contact, users*, *posts* y *topics*, necesarias para poder guardar la información necesaria y llevar a cabo las consultas correspondientes que permiten al blog ser un producto totalmente funcional.

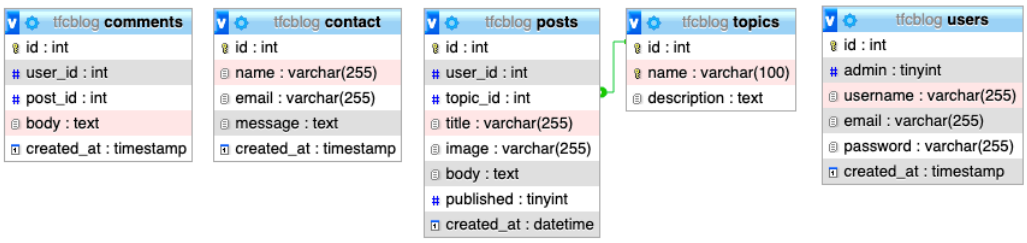

Imagen 5. Tablas de la base de datos.

En términos muy generales, el blog es esto. Si se quisiera profundizar más en el funcionamiento del mismo, bastaría con ir al código y leer los comentarios que intentan explicar cada sección del mismo.

4. C. Resultados y análisis

La manera de proceder para la realización del trabajo ha sido: crear primero la parte del diseño generando contenido estático, y después darle vida y dinamismo a todo con PHP y la base de datos.

El resultado final, que sirve como respuesta a esta pregunta, está disponible en los siguientes enlaces:

- Para descargar todo el proyecto: https://github.com/pabloherranzcano/TFC
- Para ver el resultado final: https://pabloherranzcano.herokuapp.com/

5. CONCLUSIONES

Después de haberme pasado más de diez días seguidos pegado a la pantalla del ordenador creando este proyecto, las conclusiones a las que he llegado son las siguientes:

- **Internet es un mundo maravilloso**: es una comunidad enorme en la que, al menos en este mundo de programación, la gente está dispuesta a ayudar de forma totalmente gratuita. Páginas como freecodecamp.org, o stackoverflow.com, o la mismísima *Youtube*, son una fuente inagotable de conocimiento al alcance de todos. Se puede aprender muchísimo, sobre cualquier tema, desde la silla de tu habitación. Y es que cuando tú tienes un error, lo más probable es que en todo el planeta Tierra no seas ni el primero, ni en segundo, ni el tercero, que se ha cruzado por delante de ese código de error o ese comportamiento inesperado.

- **Pegarse con el código es la mejor forma de aprender programación**: al menos en mi experiencia, cuantas más horas y más errores voy descubriendo, más aprendo. La técnica del prueba y error funciona.

- **Es importante tener bien estructurado el código**: no es de esta manera, tardas infinitamente menos tiempo en ir acotando un error para poder arreglarlo. Un código limpio y con comentarios (sé que los de este proyecto no son los mejores) no sólo ayudarán a los demás a entenderlo todo, sino a ti el primero.

- **Saber utilizar GIT, o cualquier sistema de control de versiones, es la clave del éxito**: aunque esté empezando con ello, he podido ver la importancia de ir haciendo pequeños *commits* que nos permitan volver a un estado determinado del código. No sabría decir cuántas veces he roto algo, he estado durante más de dos horas buscando la solución, y he sucumbido al "*git reset --hard*". No saber usarlo correctamente, como yo, también levanta muchos quebraderos de cabeza.

- **Saber inglés te abre muchísimas más puertas**: aunque no sea bilingüe, el hecho de poder entender lo que leo en la lengua de Shakespeare, me permite acceder a muchísima más información que la disponible sólo en español.

- **La frustración es real**: no sé cuántas veces he estado a punto de tirar el portátil por la ventana después de haber estado horas atascado en la misma historia.

- **La maquetación Web es horrible**: pasar de un "*margin: 10 px 0;"* a un "*margin: 5px 0;"* y ver cómo se produce un efecto cascada que descoloca todos los elementos de la Web es una sensación horrible.

- **No hay ningún lenguaje más aburrido de escribir que PHP**: al menos de los que he tocado. El hecho de tener que escribir "$" cada vez que queremos llamar a una variable lo hace bastante pesado. Aunque me ha sorprendido gratamente.

- **Soy más capaz de lo que pensaba**: cuando me incorporé por primera vez al módulo de Desarrollo de Aplicaciones Web, al mes de empezar las clases, y el profesor me preguntó que si sabía algo de programación, me tuve que reír porque entré sin tener ni idea de nada. Es alucinante lo que se puede aprender echándole ganas y horas a algo.

- **Las posibilidades de CRUD son muy amplias**: una vez se sabe hacer CRUD, imagino que no será muy diferente de uno lenguaje a otro. Y dándole unas vueltas a la cabeza, el sistema nos puede servir para hacer muchas cosas. Por ejemplo, es la estructura básica para hacer una página de reservas de un restaurante, o de un polideportivo, o también nos podría valer para crear una biblioteca online. Las posibilidades son infinitas.

6. LÍNEAS DE INVESTIGACIÓN FUTURAS

Si algo he aprendido en estos dos años, es que, en este mundo de la programación, hay que estar estudiando continuamente porque no paran de salir nuevas tecnologías que implementan nuevas funcionalidades. O sirven para una cosa, u otra, o se utilizan más que otras por un motivo, u otro.

Por lo tanto, el camino a seguir después de esto es ampliar al máximo la lista de lenguajes que ya conozco, y no estar nunca parado, para no quedarme atrás.

Pretendo estudiar mucho más a fondo *JavaScript* y *NodeJS* para desarrolla un perfil más *FullStack*. Y, sobre todo, voy a intentar evitar el *front* todo lo que pueda. A mi parecer, lo divertido de todo esto está en la lógica y las conexiones que se realizan en el back. Así que invertiré todos mis esfuerzos en estudiar lo que sea necesario para acabar trabajando en ese lado de la programación.

7. BIBLIOGRAFÍA

DESVENTAJAS AL USAR CMS/GESTORES DE CONTENIDOS. (2019). ZIPVISUAL. Recuperado 13 de abril de 2021, de https://www.zipvisual.com/blog/desventajas-al-usar-cmsgestores-de-contenidos/

VENTAJAS DE UTILIZAR UN CMS. (2020). TRAZADA. Recuperado 13 de abril de 2021, de https://trazada.com/ventajas-de-utilizar-un-cms/

5 RAZONES PARA ESTUDIAR FP DESARROLLO DE APLICACIONES WEB. (2017). IFP. Recuperado 13 de abril de 2021, de IFP Website: https://www.ifp.es/blog/5-razones-para-estudiar-fp-desarrollo-de-aplicaciones-Web

Cahuana, J. L. (2020). 10 RAZONES PARA USAR UN GESTOR DE CONTENIDO O (CMS). NETTIX. Recuperado 13 de abril de 2021, de Nettix Website: https://www.nettix.com.pe/blog/Web-blog/10-razones-para-usar-un-gestor-de-contenido-o-cms

Trigueros, R (2015). 7 VENTAJAS DE USAR UN CMS EN TU PÁGINA WEB. MLGDISENO. Recuperado 14 de abril de 2021, de https://www.mlgdiseno.es/7-ventajas-de-usar-un-cms-en-tu-pagina-Web/

VENTAJAS Y DESVENTAJAS DE USAR UN CMS PARA DESARROLLAR TU SITIO WEB. (2017). TACTIC-CENTER. Recuperado 27 de abril de 2021, de https://tactic-center.com/desarrollo-Web/ventajas-desventajas-usar-cms-desarrollar-sitio-Web/?locale=es

Grant, E. (2015). 9 REASONS YOU SHOULD NEVER USE A CMS. WEBDESIGNERDEPOT. Recuperado 27 de abril de 2021, de https://www.Webdesignerdepot.com/2015/07/9-reasons-you-should-never-use-a-cms

Larson, Q. (2014). FREECODECAMP. Recuperado 29 de abril de http://freecodecamp.org

Melvine, A. (2016). PHP CRUD CREATE, EDIT, UPDATE AND DELETE POSTS WITH MYSQL DATABASE. CODEWITHAWA. Recuperado 02 de mayo de 2021 de https://codewithawa.com/posts/php-crud-create,-edit,-update-and-delete-posts-with-mysql-database

YOUTUBE CHANNEL PACKETCODE. Recuperado 03 de mayo de 2021 de https://www.youtube.com/channel/UCaO2mN0IoM9WPE3Y6L7KS0w

YOUTUBE CHANNEL AWA MELVINE. Recuperado 03 de mayo de 2021 de https://www.youtube.com/channel/UCjOVC0tPI078a_DNeYs7ETQ

Penuel (2010). INTEGRATING FCKEDITOR FILEMANAGER IN CKEDITOR. MIXEDWAVES. Recuperado 08 de mayo de 2021 de http://www.mixedwaves.com/2010/02/integrating-fckeditor-filemanager-in-ckeditor/

Adams. D. (2015). COMMENTING SYSTEM WITH PHP, MYSQL, AND AJAX. CODESHACK. Recuperado 10 de mayo de 2021 de https://codeshack.io/commenting-system-php-mysql-ajax/