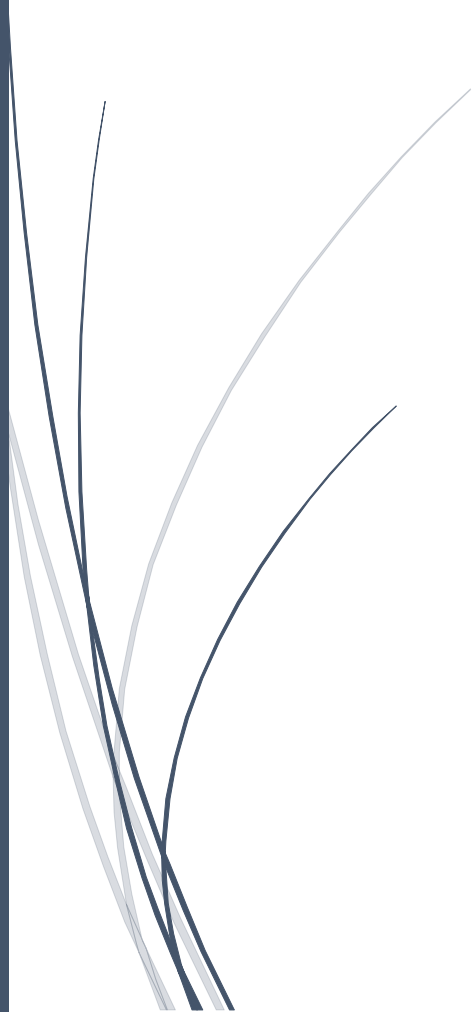
10-6-2020

MEMORIA

WWW.FUONLE.COM



2º DAW

JUAN FERNÁNDEZ GALÁN

ÍNDICE

JUSTIFICACIÓN DEL PROYECTO

En un principio me plantee hacer una aplicación web de chat y llamadas online. Justo antes del comienzo del proyecto vi que había un grave problema con los padres, madres y profesores de primaria y secundaria al mandar tareas mediante documentos (PDF, Word, etc…) al alumnado.

Me di cuenta que en la mayoría de casos que conocí de mi entorno cercano, los profesores se ponían en contacto con un padre o madre, pasándole las tareas para que este mismo lo pasara a través del grupo de WhatsApp a los demás padres. Eso era un incordio, la mitad de padres no sabían ni si quiera de la existencia de WhatsApp web para la descarga o visualización en pc del PDF y tener que estar con pendiente de un niño mientras estas con el móvil en la mano no era una tarea fácil.

Muchos institutos hacen uso de Moodle, como en el caso del I.E.S. Las Fuentezuelas, pero no es el caso de todos los institutos, o en algunos casos el buen uso o implementación de esta aplicación web.

Tras darle muchas vueltas pensé que ese problema podría tener una solución, una solución que no solo se hiciera cargo de las necesidades y comodidad de los padres y profesores, sino, que una academia de barrio, un grupo de amigos, un grupo de padres y madres, un grupo de profesores… en definitiva al alcance de cualquier necesidad educativa.

FUNCIONALIDADES EN LAS QUE SE DIVIDE

Mi aplicación web, www.fuonle.com te permite crear aulas virtuales protegidas mediante contraseña, whitelist para matricular a usuarios que desees y administrar tu propia aula virtual tú mismo o con la ayuda de los usuarios que designes como administradores. Un usuario deberá estar registrado tanto para crear un aula virtual como para acceder a ella.

Cada aula virtual podrá contener tantas secciones como desee el administrador, sección para matemáticas, sección de lengua, sección de historia del arte…

Cada sección dispondrá de documentos PDF, los cuales se podrán subir hasta un máximo de 10 a la vez y un máximo de 30 MB por PDF, los documentos se podrán subir bien arrastrando o bien seleccionando desde el cuadro de dialogo. Las secciones dispondrán de un chat en el cual se podrán resolver dudas entre alumnos y administradores, puntualizar algo, etc…

No todos son aulas virtuales, la web también dispone de una parte en las que los usuarios podrán subir documentos de manera anónima, siempre y cuando estés registrado en la web, consultar los documentos sí que se podrá hacer sin estar registrado, pero no añadirlos a favoritos.

Los documentos se podrán reportar si su contenido o título no se corresponden, o bien si su filtrado no está bien definido. Un usuario deberá de estar registrado para poder reportar un documento.

El sistema de filtrado se divide de la siguiente manera:

- Nivel de estudio
- Categoría
- Subcategoría
- Tipo de documento
- Nombre de documento

Los usuarios registrados también podrán hacer sugerencias para mejorar la web haciendo clic en '*suggestion*' localizado en el pie de página.

MÓDULOS DEL CICLO FORMATIVO QUE INTERVIENEN

En este proyecto lo módulos que han intervenido son los siguientes:

- Despliegue de aplicaciones web
    - Comprar un dominio para la aplicación.
    - Alquilar y configurar un servidor para poder desplegarla.

- Desarrollo web en entorno cliente
    - Por la utilización de la tecnología JavaScript (Angular) para el Front end.

- Desarrollo de aplicaciones web en entorno servidor
    - Por la utilización de node js en el Back end para la realización de consultas con la base de datos, manejo de información sensible.

- Diseño de interfaces web
    - Por la utilización de css y sass en los estilos del proyecto.

TECNOLOGÍAS ELEGIDAS Y JUSTIFICACIÓN DE DICHA ELECCIÓN

Las tecnologías elegidas para este proyecto han sido:

- MongoDB para base de datos
- Node js para Back end
- Angular 9 para Front end

MongoDB

MongoDB es una base de datos orientada a documentos. Esto quiere decir que, en vez de guardar los datos en registros, los guarda en BSON, que es una representación binaria del JSON. Por esto se puede decir que la velocidad de sus consultas es mucho mayor que en las bases de datos convencionales como SQL.

La base de datos se divide en documentos, que es lo que se podría llamar una *"tabla"* en SQL, los documentos no siguen un tipo de esquema, ya que un insert de un usuario podría tener nombre y apellido y el siguiente insert podría tener nombre, apellido y DNI, por lo que no lanzara ningún error.

Los esquemas de cada documento se controlarán desde Node js, ya que desde ahí podemos definir qué tipo de dato cada propiedad (string, date, number…) y a su vez que propiedad es requerida o no, así como definir como valores por defecto y demás.

Node js

Unas de las ventajas de Node js son la rapidez, el bajo coste de recursos y la utilización de un único lenguaje, sin embargo, también lleva inherentes otras ventajas en cuanto a su funcionamiento. Una de ellas es que Node js puede correr en prácticamente todas las plataformas sin gran coste de configuración y de mantenimiento, lo que ya es un punto a la hora de elegir Node js. Además, cambia el concepto de funcionamiento de servidor. En los servidores tradicionales, se genera un nido por cada petición que se hace, en cambio Node js funciona de otra manera. Node js genera un evento por cada petición que recibe, podemos decir que se parecen mucho a las tradicionales llamadas "Ajax" y es en su núcleo donde se genera esa gestión multiproceso de todas las operaciones que se refieren a partir de este canal único. Esto que parece un problema, realmente es lo que diferencia a Node js. Con esto se

consigue que sea tan sumamente rápido y sea perfecto para aplicaciones en tiempo real. Además, presenta el Node js un gran repositorio de paquetes que pueden aumentar sus funcionalidades y sus capacidades con una amplia comunidad que los soporta, en grandes empresas como por ejemplo Google. Junto con la librería “Mongoose” hace un equipo perfecto con MongoDB por la gran rapidez de consultas que forman.

Angular 9

Angular es uno de los frameworks de código abierto líderes en el mercado para el desarrollo de aplicaciones, tanto web como móvil. Y no podría ser de otra manera, pues el equipo de Angular (perteneciente a Google) nos tiene acostumbrados a una versión mayor cada 6 meses, incluso manteniendo la API estable desde el lanzamiento de Angular 2.

Angular también posee TypeScript, **TypeScript es un lenguaje de programación construido por encima de JavaScript**. Este superconjunto de JavaScript dota al lenguaje de varias características adicionales que hacen que podamos escribir código con menos errores, más sencillo, coherente y fácil de probar. Unas de las características de TypeScript son la programación orientada a objetos, promesas, programación asíncrona, símbolos, etc…

ETAPAS POR LAS QUE HA PASADO EL PROYECTO

En mi caso divido el proyecto en 3 etapas:

- Definición BD
- Creación de servidor
- Creación de Cliente

Definición de BD

Como he mencionado antes, MongoDB no es una base de datos relacional ni sigue ningún tipo de esquema, bien es cierto que antes de nada me pare un par de días para definir los diferentes tipos de esquemas que iba a almacenar en la base de datos (creados con Node js) para poder dar una relación mediante PK a los diferentes documentos que pudieran llegar a tener una relación en el futuro.

Creación de servidor

Node js es un servidor totalmente independiente del cliente, por lo que su desarrollo y testeo se puede hacer sin ningún tipo de cliente. Con este tipo de tecnología no había trabajado, por lo que le dedique un tiempo extra a su desarrollo.

Esta etapa se puede dividir en 3 fases:

- Modelos
    - Creación de los modelos (objetos) con lo que vamos a trabajar a lo largo de la aplicación, cada modelo va a tener sus propias propiedades y funciones como podemos ver en el siguiente ejemplo:

```
const UserSchema = new Schema({
    email: { type: String, unique: true, lowercase: true, required: true
},
    password: { type: String, select: false, required: true },
    userName: { type: String, unique: true, lowercase: true, required: tr
ue },
    token: String,
    _id_docs_favorites: Array,
    avatar: String,
    _id_rol: { type: String, default: "user" },
    signupDate: { type: Date, default: Date.now() },
    lastLogin: Date
})

UserSchema.pre('save', function(next) {
    if (!this.isModified('password')) return next()

    bcrypt.genSalt(10, (err, salt) => {
        if (err) return next(err)

        bcrypt.hash(this.password, salt, null, (err, hash) => {
            if (err) return next(err)

            this.password = hash
            next()
        })
    })
})
```

- Controladores
    - En los controladores vamos a poder definir todo tipo de acciones que van a realizar los modelos, en los que puede contener funciones para hacer un insert, update, remove, etc…

- Enrutamiento
    - En el enrutamiento definiremos las direcciones url y sus parámetros si fuese necesario, por las que el cliente se comunicara con el servidor. Así como poner un sistema de autentificación por token para cortar peticiones no deseadas.

Como he mencionado antes, el servidor es totalmente independiente del cliente, por lo que una vez terminado cada modelo con su controlador y enrutador podemos llevar a cabo

pruebas para asegurarnos de que todo funciona correctamente o en su caso hacer algún cambio si fuese necesario. Las pruebas las realice con la aplicación Postman.

Postman es una aplicación que te permite hacer todo tipo de peticiones al servidor, como pueden ser POST, GET, PUT, DELETE, etc… También nos permite generar peticiones con Params, Headers y Body. Esto siempre nos va a permitir tener una certeza casi garantizada de buen funcionamiento del servidor cuando generemos peticiones desde el cliente.

Creación del cliente

Tercera y última parte, Angular nos va a permitir plasmar de manera gráfica la aplicación. En esta parte tendremos todos los componentes (fragmentos de código) y sus correspondientes estilos, que mostraran los datos creados mediante HTML y los recibidos del servidor mediante servicios.

PROBLEMAS ENCONTRADOS Y SOLUCIONES APORTADAS

Unos de los grandes problemas que me han surgido es con que tipos de documentos iba a trabajar. En primera instancia, la idea era trabajar con .pdf, .docx y .doc, pero surgieron problemas. Los navegadores saben interpretar los .pdf abriéndolo y mostrándolos tal cual están redactados incluyendo imágenes y formatos de letra. No es así el caso de los documentos de office, ya que los navegadores no lo saben interpretar, de tal manera que habría que renderizar un documento office en HTML y es aquí donde está el quiz de la cuestión.

Tras probar varias librerías, ninguna mantenía el formato de texto y algunas ni mostraban imágenes. Otra opción era compra una API de alguna empresa que ofrecen este servicio de convertir documentos office en HTML, tras varios días de investigación me di cuenta que esta opción tampoco llegaba a respetar los formatos e imágenes de este tipo de documento, por lo que me he decido por manejar solo archivos pdf e intentar esperar a encontrar una solución que pueda respetar estos archivos en su totalidad.

Otro problema que he tenido ha sido la compatibilidad de Angular Material con todos los navegadores, Angular Material son estilos proporcionados por el mismo framework, más concretamente los cuadro de dialogo modales, tras ir probando repetidamente la compatibilidad de los estilos con Microsoft Edge, Firefox, Opera y Google Chrome, Estos modales cortaban el background-image que tiene la aplicación de fondo en los navegadores de Google Chrome y Firefox.

Siempre he estado acostumbrado a tratar las alturas con porcentajes y pixeles, no es el caso cuando se está trabajando con sass y Angular Material, en este caso, los estilos son totalmente fieles cuando las medidas se tratan con vh “viewport height”, este desconocimiento me hizo tener que llevar a cabo una investigación sobre posicionamiento en esta tecnología.

NIVEL DE CONSECUCIÓN DEL MISMO

Hasta el momento el proyecto se encuentra totalmente funcional para el uso público de usuarios, en lo que se podrá crear aulas virtuales, añadir y eliminar administradores de un aula, permitir la entra de alumnos a un aula, crear secciones en un aula, añadir y quitar documentos en una sección, chat incorporado para cada sección, subir documentos públicos y anónimos que cualquier usuario podrá consultar, añadir a favoritos y reportarlos en el caso de que no se corresponda a la realidad o no esté bien filtrado, también se ha incorporado un buzón de sugerencias para que cualquier usuario aporte ideas sobre mejoras en la web.

Actualmente me encuentro trabajando en la parte privada de la aplicación en la que solo se podrán acceder usuario con el rol de administrador. En este apartado se podrá administrar todo lo referente a la aplicación, aulas virtuales, secciones, documentos privados y públicos, usuarios, chat, etc…

RECURSOS NECESARIOS PARA SU IMPLEMENTACIÓN Y PRESUPUESTO ECONÓMICO

Los recursos necesarios para su implementación de manera local serían las dependencias de node para la interpretación del código, y las librerías de angular para el desarrollo de la aplicación. En cuanto a presupuesto económico hace falta ningún presupuesto para su desarrollo en local.

Llegado el momento del despliegue de la aplicación tendríamos que comprar un dominio y alquilar un servidor. Los dominios varían en precio según su extensión, los más comunes que son .com y .es rondarían entre los 10€ - 25€ dependiendo de la empresa a los que se le compre, por otro lado, la elección del servidor no es fácil, ya que no todos soportan una tecnología de node js, por lo que he podido comprobar un servidor puede ir desde 2€/mes hasta +60€/mes. Mi investigación me ha llevado a la elección de servidores, los cuales su precio es de unos 7€/mes y con muy buenas opiniones de sus clientes. También sería imprescindible que tenga acceso SSH para poner configurar el servidor nosotros mismos con las dependencias que hicieran falta instalar.

CRITERIOS DE CALIDAD

El proyecto tiene cada una de las funciones documentadas, aunque de por si el nombre de cada función es muy intuitivo con respecto a su función, las variables también están documentadas, algunas más específicamente porque así lo requieren.

El código del proyecto esta lo más diferenciado y separado posible, se ha intentado hacer funciones lo menos extensas posibles de tal manera que cada función haga un trabajo específico, así conseguimos que el encontrar un fallo no sea un caos y tener localizado el curso del código en cada trabajo que lo requiera.

Otro aspecto importante del que me he documentado es la vista, la vista tienda a cansarse más con la exposición prolongada de colores con mucho brillo, mi idea fue darle un tema tenue a mi aplicación. El estudio en una pantalla de ordenador puede llevar a un gran cansancio en la vista, por lo que si desde mi aplicación puedo aportar relajación en la tensión ocular y así intentar prevenir la vista cansada.

La composición de las etiquetas DOM son bastantes claras e intuitivas para no hacer pensar en exceso al usuario, así cuando necesite hacer algo no dude, tenga claro donde este cada objetivo y cómo hacerlo, simplificando al máximo cada acción que pudiera querer hacer el usuario.

En Fuonle no tomamos muy en serio la opinión de los usuarios, por ello se ha incorporado un buzón de sugerencias, que se atenderá diariamente y se estudiará las propuestas comunes de los usuarios por si hubiese hacer algún cambio en la aplicación que beneficie a toda nuestra comunidad.

MEJORAS QUE SE PODRÍAN REALIZAR EN EL FUTURO

El proyecto Fuonle tiene preparadas muchas mejoras en el futuro, no se han podido implementar de momento porque requiere más tiempo, pero sí que el servidor ya está preparado para algunas de ellas.

Amistades entre usuarios

Se implementará un sistema de amistades para que los usuarios se puedan agregar entre sí.

Chat para amigos

Al igual que los usuarios se podrán agregar como amigos, podrán chatear.

Más extensiones de documentos

No solo se podrán compartir archivos PDF, se ampliará a documentos de Office y archivos de video.

Versión móvil

Esta es la mejora más importante y la que se implementara a corto plazo, ya que hoy en día los dispositivos móviles están al alcanza de cualquier usuario y en cualquier lugar. También el acceso prioritario será a través de las RRSS por lo que es importante que la aplicación esté preparada esté preparada para su acceso desde un dispositivo reducido en pixeles.

Control administrador

Aunque ya se está trabajando en ello, es muy importante que la aplicación tenga un control privado para administradores.

Enviar email desde un aula

Ya que Gmail, actualmente limita el servicio de correos enviados a 500 por día, en poco tiempo se creará un servidor privado de correo para Fuonle, para que así los administradores de las puedan enviar correos a sus alumnos a través de Fuonle, eligiendo si enviar a todos sus alumnos o solo a los seleccionados.

BIBLIOGRAFÍA

- https://openwebinars.net/academia/portada/mongodb/
  - Bases y utilización de MongoDB

- https://carlosazaustre.es/como-crear-una-api-rest-usando-node-js
  - Bases y utilización de Node JS
  - Interactuar con MongoDB

- https://openwebinars.net/academia/portada/angular/
  - Bases y utilización básica de Angular 8

- https://openwebinars.net/academia/aprende/estructura-proyecto-angular/8079/
  - Estructuración y buenas prácticas de Angular

- https://stackoverflow.com/
  - Consulta de dudas y errores

- https://github.com/
  - Consulta de dudas y errores