Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Plataforma web de automatización de caracterización electrónica.

Autora: Ana María Brázquez Villarán
Tutores: Fernando Muñoz Chavero,
José María Hinojo Montero

Departamento de Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Plataforma web de automatización de caracterización electrónica.

Autora:
Ana María Brázquez Villarán

Tutores:
Fernando Muñoz Chavero
José María Hinojo Montero
Profesor titular
Fernando Muñoz Chavero

Departamento de Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Proyecto Fin de Carrera: Plataforma web de automatización de caracterización electrónica.

Autora: Ana María Brázquez Villarán

Tutor: Fernando Muñoz Chavero,

José María Hinojo Montero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mis seres queridos

A mis profesores

A mis compañeros

Agradecimientos

Hoy, a punto de finalizar una bonita etapa, considero primordial escribir este apartado de agradecimientos. Ha sido un período de aprendizaje intenso, no solo en el campo científico, también a nivel personal. Vivir esta etapa ha tenido y tendrá un gran impacto y enseñanza en mí y es por eso que me gustaría agradecer a todas aquellas personas que me han ayudado y apoyado durante este proceso.

En primer lugar, me gustaría agradecer a mis padres. Me habéis apoyado enormemente y siempre habéis estado ahí para ayudarme cuando lo necesitaba. Pocas personas se vuelcan tanto en la educación de sus hijos y, no hay nada que me enorgullezca más que podáis apreciar los frutos de tanto esfuerzo. Aquí quiero incluir a mi hermano, del que me siento y me sentiré siempre tan orgullosa.

En segundo lugar, me gustaría hacer un agradecimiento especial a mi pareja por sus consejos, su constante apoyo, comprensión y cariño. Siempre será un placer compartir contigo el interés por el mundo de la ingeniería y disfrutar de nuestras charlas eternas, debates y risas toda la vida.

Además, me gustaría darles las gracias a mis tutores Fernando Muñoz y José María Hinojo, por su valiosa ayuda. Muchas gracias por vuestra cooperación, por escuchar siempre nuestras propuestas y brindarnos todas las herramientas necesarias para completar nuestro trabajo de fin de grado satisfactoriamente.

También me gustaría agradecer a mis compañeros/as de carrera, tanto mis niñas como mis niños, aquellos que me habéis dado tantísimos momentos buenos hasta en las situaciones más duras.
Finalmente, a mi compañero de TFG y amigo, Manu. Gracias por compartir conmigo estos años y por ser una persona que sé que siempre va a formar parte de mi vida. Ha sido un placer cerrar esta etapa a tu lado.

Siempre habéis estado ahí para mí y yo lo estaré siempre para vosotros.

Ana María Brázquez Villarán

Escuela Técnica Superior de Ingeniería

Sevilla, 2020

Resumen

Son muchas las personas, profesionales y estudiantes, dedicadas al mundo de la electrónica. En este ámbito, cobra un papel importantísimo el manejo y conocimiento de los distintos instrumentos de laboratorio y la configuración de parámetros de señal en ellos.

En ocasiones, es necesario enfrentarse a circuitos tediosos de controlar y configurar.

Hemos querido aportar un pequeño grano de arena para facilitar este proceso a nuestros futuros compañeros.

Para ello, se va a implementar un entorno frontend – backend mediante el cual, con tan solo rellenar un formulario web, el usuario podrá transformar una señal como le plazca (por ejemplo, calcular la FFT).

Además, se propone una segunda opción en la web para calcular parámetros como amplitud, frecuencia, periodo etc de una señal dada.

En este documento, se explicará la parte frontend de esa plataforma de automatización medidas con supervisión remota a la hora de caracterizar un dispositivo.

En un futuro, esta web se ampliará hacia tantas pruebas y configuraciones como se deseen proporcionar al usuario que la utilice.

Abstract

There are many people, professionals and students, dedicated to the electronic's world. In this area, handling and knowledge of the different laboratory instruments and the configuration of signal parameters in them take on a very important role.

Sometimes it is necessary to face tedious circuits to control and configure.

We wanted to contribute a small grain of sand to facilitate this process for our future colleagues.

To do this, a frontend - backend environment is going to be implemented through which, just by filling a web form, the user can transform a signal as they wish (for example, calculate the FFT).

In addition, a second option is proposed on the web to calculate parameters such as amplitude, frequency, time etc. of a given signal.

In this document, we will explain the frontend part of this automation platform measured with remote supervision when characterizing a device.

In the future, this website will be expanded to as many tests and configurations as you want to provide to the user who uses it.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Estar preparado es importante, saber esperarlo es aún más, pero aprovechar el momento adecuado es la clave de la vida

- Arthur Schnitzler -

A lo largo de estos años, son muchos los instrumentos, circuitos, herramientas, dispositivos y componentes que hemos trabajado o, a veces, contra los que hemos combatido. Nuestra principal motivación era, por tanto, facilitar cualquier tipo de procesamiento de señal o cálculo de parámetros mediante una plataforma web capaz de automatizar la caracterización de señales e instrumentos electrónicos.

1.1 Motivación

Desde un principio se pensó en las numerosas ventajas que puede ofrecer una plataforma de automatización de medidas con supervisión remota a la hora de caracterizar un dispositivo.

En primer lugar, aporta un gran ahorro de tiempo, ya que consistiría en una aplicación web lo más cómoda y rápida posible para la interacción con usuarios, en la cual se pudiesen configurar señales, calcular parámetros y realizar varias operaciones de manera sencilla, sin necesidad de interactuar con el instrumento electrónico o usar MatLab de por medio.

Además, la generación de esta plataforma web que automatizara la caracterización de un circuito electrónico concreto conllevaría otro tipo de ventajas como la generación de informes automáticos (con solo completar cuadros de texto sobre las propias gráficas que genere la web, se podría imprimir un documento totalmente válido como aporte de documentación) o la opción de llevar a cabo ensayos de fiabilidad.

Otra gran ventaja sería la posibilidad de realizar campañas de medida con una duración extensa o supervisión online de experimentos que se encuentran en entornos donde no puede existir intervención humana (aceleradores de partículas o ensayos donde se prueba la tolerancia de los dispositivos electrónicos a radiación).

1.2 Objetivo

Como se ha explicado en el apartado anterior, la motivación principal de este proyecto es crear una plataforma de automatización de medidas. Por tanto, el objetivo principal será diseñar una interfaz de usuario basada en una aplicación web para la adquisición de medidas de forma remota.

Esta interfaz permitirá al usuario configurar distintos instrumentos, representar las señales adquiridas y los datos obtenidos de su procesamiento. Para ello, se buscará una interfaz sencilla que otorgue al usuario la oportunidad de realizar configuraciones y obtener la información de respuesta con solo rellenar un formulario.

Cabe destacar que la plataforma desarrollada se empleará incialmente para la caracterización de convertidores sigma-delta, dada la complejidad que presenta este caso de uso.Por esa razón, nos hemos centrado en el desarrollo de un frontend que muestre la respuesta temporal de la señal adquirida y su respuesta en frecuencia, gracias a la estimación de su transforamda de Fourier.

1.2.1 Convertidor Sigma-Delta

La modulación **Sigma-Delta** (ΔΣ) es un tipo de conversión analógica-digital o digital-analógica. Un circuito conversor analógico-digital (ADC) que implemente esta técnica puede ser realizado fácilmente usando sistemas de bajo coste del tipo CMOS, semejantes a los usados para fabricar circuitos integrados digitales.

Es por esta forma de construirlos, por lo que, pese a haber sido propuestos por primera vez a principios de los años 60, solo se ha generalizado su uso en los últimos años, gracias al empleo de tecnologías basadas en el silicio. La mayor parte de los fabricantes de circuitos integrados analógicos ofrecen moduladores sigma-delta.

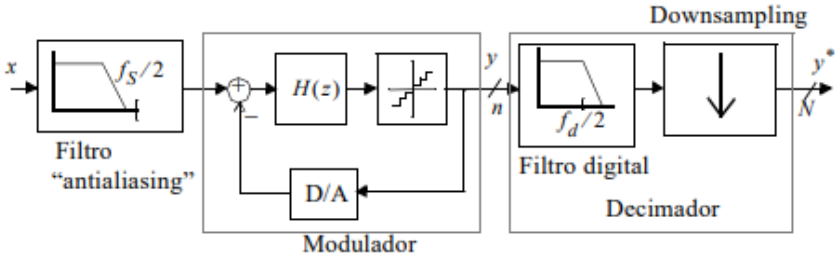

Figura 1-1. Diagrama de bloques convertidor sigma-delta.

La Fig. 1.1 muestra el diagrama de bloques de un convertidor A/D de sobremuestreo con modulación SD que consta de los siguientes elementos:

(a) Filtro analógico "antialiasing". Se encarga de eliminar de la señal de entrada todas las componentes espectrales por encima de la mitad de la frecuencia de muestreo. La operación de sobremuestreo permite flexibilizar los requisitos de este filtro de forma que incluso filtros pasivos de primer orden son suficientes para implementar el primer bloque del convertidor.

(b) Modulador. En este bloque se muestrea y cuantiza la señal. Además, es posible filtrar el error de cuantización, haciendo que su densidad espectral de potencia quede fuera de la banda de la señal, de donde es eliminado mediante filtrado digital. Este hecho ha dado origen al término "noise-shaping", que se usa también para denominar a los moduladores SD ("noise-shaping coders").

La salida del modulador consiste en un número reducido de bits (normalmente es sólo uno) a la frecuencia de muestreo.

(c) Decimador. En este bloque, 100% digital, tras un filtrado que elimina todos los componentes fuera de la banda de la señal, incluido gran parte del error de cuantización, se reduce la frecuencia de muestreo mediante un proceso de decimación. Como resultado, se obtiene la señal de entrada, codificada con un gran número de bits, a la frecuencia de Nyquist.

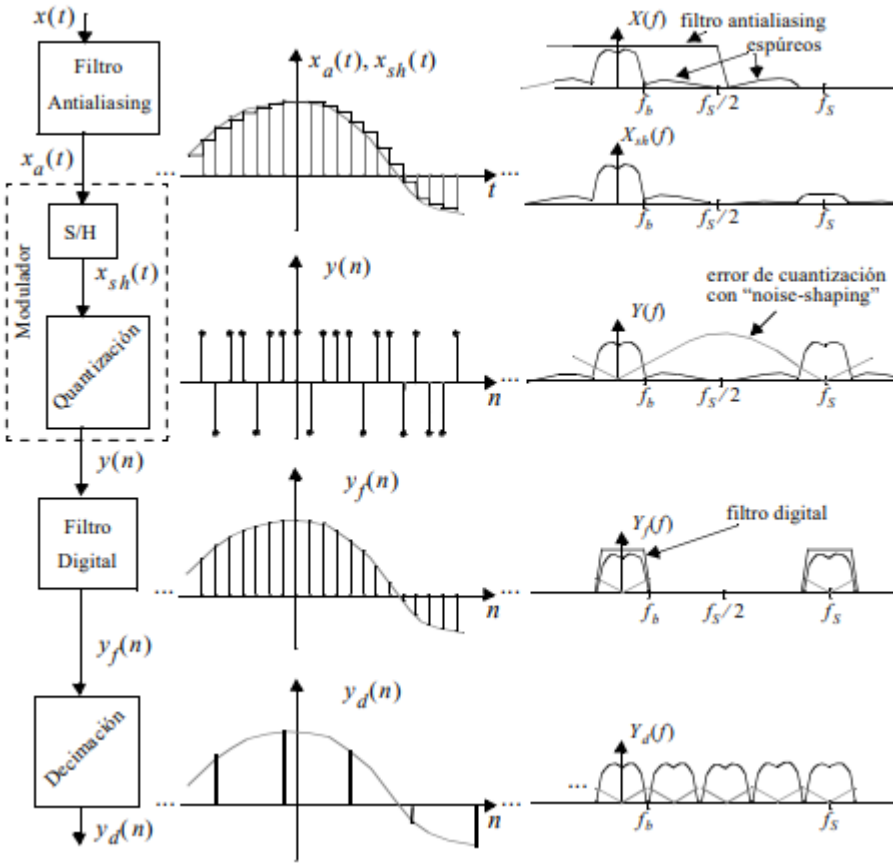

Figura 1-2 Procesado de señal en convertidor sigma-delta

Por otro lado, a Fig. 1.2 muestra el procesado de la señal realizado por los distintos bloques del convertidor.

En esta imagen, se ha separado en el modulador la operación de muestreo de la de cuantización, que consta de un solo bit. Del mismo modo, en el decimador se han aislado las operaciones de filtrado y decimación. Tener en cuenta que esta última operación no supone pérdida de información dado que sólo se elimina la información redundante que resulta del sobremuestreo.

De los tres bloques del convertidor, el modulador suele ser el que mayor dificultad plantea al diseñador. Por un lado, el sobremuestreo reduce el filtro antialiasing a un sencillo filtro RC paso de baja. Por otro, el decimador, es un bloque puramente digital cuyo diseño, al igual que el de otros bloques de procesado digital de señal, se encuentra bastante estructurado y resulta abordable mediante herramientas de CAD ampliamente conocidas.

En el modulador, situado en la frontera entre el plano analógico y el digital, podemos encontrar los mecanismos de error que degradan el comportamiento del convertidor: por una parte, el error de cuantización y por otra, toda una serie de condiciones no ideales del circuito que en mayor o menor medida afectan la funcionalidad de los bloques analógicos que lo forman.

El impacto de estas últimas, así como las interferencias causadas por los bloques digitales vecinos, deben tenerse en cuenta en el diseño de convertidores con especificaciones exigentes y velocidad de operación.

1.2.2 Flujo de trabajo

SwaggerHub

Puesto que estamos ante un trabajo frontend comunicado con un backend (no englobado en el alcance de este proyecto), durante el desarrollo del mismo ha existido una constante comunicación y claridad entre las necesidades de frontend hacia backend y viceversa. Por este motivo, se propuso el uso de una herramienta muy útil a la hora de interactuar con el resto de equipos de desarrollo: SwaggerHub.

Figura 1-3. SwaggerHub

Cualquier API de alta calidad comienza con estándares de diseño que se alinean con los objetivos determinados. Con SwaggerHub, se puede acelerar el proceso de diseño de un equipo de trabajo y, al mismo tiempo, garantizar la calidad y la coherencia del estilo. El editor de API cumple con Swagger, ahora conocido como las Especificaciones OpenAPI (OAS), simple e intuitivo.

Para actualizar el flujo de trabajo de diseño de API al trabajar con un conjunto de personas, podemos implementar:

- Comentarios inteligentes de errores y autocompletado de sintaxis dentro del editor
- La capacidad de crear simulacros de API automáticamente mientras diseñamos
- Reglas de diseño de API integradas que refuerzan los estándares en tiempo real
- Dominios para catalogar y reutilizar la sintaxis OAS común en las API

Nada ralentiza el desarrollo tanto como la falta de comunicación. Por ello, decidimos usar SwaggerHub, ya que ha sido diseñado para fomentar la colaboración API y la estandarización en múltiples equipos. Al aprovechar SwaggerHub como su única fuente para las definiciones de API, hemos podido definir los distintos métodos usados en Angular (frontend) y Python (backend), especificando el tipo de método (GET, PUT, POST, DELETE, etc), los parámetros de entrada y sus tipos, el path del método, la salida deseada, etc. De esta forma, tanto la parte backend como la frontend ha tenido constante acceso a esta plataforma para no tener duda de qué necesitaba el uno del otro en cada momento.

Todo esto, por supuesto, ha ido acompañado de una comunicación constante vía oral y mensajería.

Tras trabajar de este modo, nos queda claro que algunas de las ventajas de esta plataforma son:

- Organización y gestión de equipos para que sepa quién posee qué.
- Permite bifurcar, comparar o fusionar con una API existente.
- Comentarios en tiempo real y seguimiento de problemas adyacentes al editor
- Sincronización de definiciones de API con repositorios de control de origen y puertas de enlace

Un ejemplo de definición de los métodos usados en Swagger sería el siguiente:

```
paths:
  /signals/signalConfiguration:
    post:
      tags:
      - signalConfiguration
      summary: Configuración de la señal del osciloscopio
      operationId: signalConfiguration
      consumes:
      - application/json
      produces:
      - application/json
      parameters:
      - in: body
        name: valoresConfiguracion
        description: JSON que contiene canal, voltaje, tiempo y frecuencia por tiempo de muestreo
        schema:
          type: object
          required:
            - canal
            - voltaje
            - tiempo
            - fMuestreo
            - puntos
```

Como puede verse en el ejemplo, se está describiendo el detalle de una señal cuya llamada sería {dominio}/signals/signalConfiuration, usado para configurar una señal en el osciloscopio basándonos en los parámetros de configuración que el usuario introduzca en el formulario de la web.

Para realizar la definición de nuestro método, es necesario especificar:

Tipo de dato que consume: JSON.

Tipo de dato que devuelve: JSON.

Cuerpo de la petición post: en un JSON de nombre "valoresConfiguración", incluimos los parámetros canal, voltaje (eje x), tiempo (eje y), fMuestreo (frecuencia de muestreo) y puntos (número de puntos por muestra, para el cálculo de la FFT).

```
properties:
        canal:
          type: string
          enum:
           - "1"
           - "2"
           - "3"
           - "4"
        voltaje:
          type: number
        tiempo:
          type: number
        fMuestreo:
          type: number
    responses:
      200:
        description: successful
operation
        schema:
          type: object
          items:
            $ref:
'#/definitions/GraficaRespuesta'
      500:
        description: Invalid
      503:
        description: Invalid
      404:
        description: Not found
      405:
        description: Validation exception
```

Por otro lado, es necesario detallar cada parámetro por searado: el tipo (en este caso usamos string y number) y en caso de ser un enumerado como nuestro parámetro “canal”, especificamos que el valor puede ir del canal 1 al canal 4.

Además, especificaremos los tipos de código de respuesta http y su significado.

2 ESTADO DEL ARTE

Siempre puedes editar una mala página, pero no una página en blanco.

- Jordi Picoult -

La investigación realizada con el objetivo de adquirir conocimientos para este proyecto se ha basado en conocer el estado de las tecnologías actuales que podían servirnos de referencia para nuestra elección de framework y demás herramientas a utilizar.
Además, se ha hecho una breve investigación sobre aquellos proyectos que ua usan plataformas de automatización de medidas para la caracterización de dispositivos.

2.1 Comparación de frameworks para front-end actuales

Una de las primeras decisiones que tomamos fue qué framework era más accesible, útil y adecuado a nuestras necesidades. Durante dicha meditación, fue necesario hondar en una pequeña investigación entre los frameworks para frontend más populares.

En primer lugar, vamos a comparar algunos de los frameworks de JavaScript más utilizados. Para ello, nos centraremos en examinar cinco aspectos diferentes de estos frameworks, aquellos que generalmente facilitan la toma de decisión.

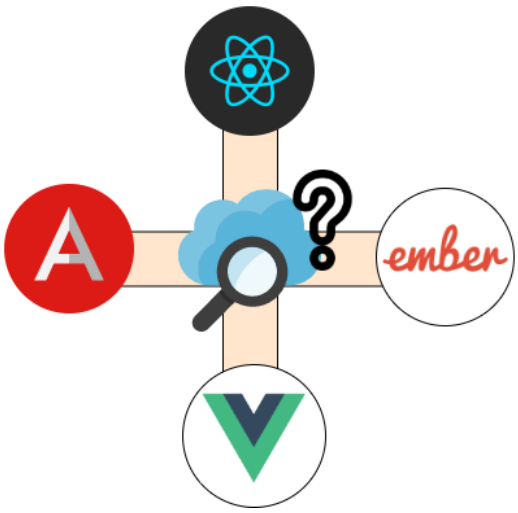

Figura 2-1. Frameworks de frontend

El framework de front-end que se elija puede afectar considerablemente al proyecto a largo plazo, por tanto, vamos a hacer una comparación de varios frameworks dentro de distintos ámbitos o puntos clave que consideramos importantes en el mundo del frontend.

Un listado de los puntos clave a la hora de elegir un framework para frontend es el siguiente:

- Disponibilidad de recursos de aprendizaje
- Popularidad
- Características ofrecidas
- Usabilidad
- Facilidad de integración (con otras librerías)

Vamos a describir cada uno y a indicar por qué razones son tan importantes para nuestra elección.

2.1.1 Disponibilidad de recursos de aprendizaje

Aunque parece obvio, a menudo es un punto que se pasa por alto. En rasgos generales, solemos dejarnos llevar por la gran página de inicio de algunos frameworks sin pararnos a pensar en que se necesitan cursos, libros, tutoriales y artículos adicionales además de documentación para comenzar a usarlos.

Debemos entender que es fácil que los grandes frameworks nos impresionen a primera vista, pero lo realmente importante es que esos mismos ingenieros que han creado frameworks impresionantes, sepan comunicar de forma clara y precisa todas las ideas que existen detrás. En este punto, uno de los errores más comunes es usar un framework demasiado nuevo o aún no adoptado por la comunidad. Aunque es positivo innovar, debemos tener muy en cuenta la documentación online, bibliográfica y comunidades que existan detrás de nuestro framework.

En el caso de los frameworks que queremos comparar, cabe destacar que **Angular, Vue** y **React** poseen grandes cantidades de recursos debido a su fama y antigüedad. El caso de **Ember** es diferente, ya que existe mucha documentación peor por lo general, presta una pésima calidad.

En definitiva, en mi opinión es crucial que puedas encontrar de manera rápida y sencilla gran cantidad de

información en cualquier formato (artículos, vídeos, cursos, libros, foros…). Cada fuente aporta nuevos y diferentes conocimientos que te harán ahorrar una de las cosas más valiosas a la hora de desarrollar cualquier proyecto: el tiempo.

2.1.2 Popularidad

Tiene un gran mérito y valor aprender algo muy nuevo, tecnologías recién salidas, pero también presenta un gran riesgo. Si lo observamos desde una perspectiva comercial, una empresa, cliente o proyecto en el que trabajes probablemente prefiera un conjunto de herramientas ya probados y cuya eficiencia está asegurada desde la experiencia de otros antes que usar el framework más innovador del mercado.

Una segunda razón para usar un framework conocido es: ¿qué ocurre si su framework es tan inutilizado que deja de respaldarse, es decir, de mantenerse o actualizarse con los años? En este caso, nuestro proyecto podría quedar obsoleto y no tener más remedio que migrarlo a otro framework con el paso de los años.

Otra razón por la que optar por un framework popular y ampliamente utilizado es: ¿Qué sucede si te encuentras atascado con algún problema y no hay una comunidad a la que puedas pedir ayuda? Ya que está solo con la documentación, es probable que se pierda mucho tiempo.

En definitiva, todos los que nos hemos dedicado un mínimo al mundo de la porgramación, sabemos que recurrir a blogs, videos, cursos, foros, comunidades, libros y vídeos es crucial para cualquier problema al que te estés enfrentando. Por tanto, vale más usar un framework cuyos problemas puedas resolver con facilidad y que esté altamente respaldado por una comunidad, antes que ser el más innovador.

Los líderes absolutos aquí son **Angular y React**. La mayoría de los grandes programadores que trabajan en front-end usan uno u otro. Están respaldados por Google y Facebook, respectivamente, y por lo tanto, haces que siempre te sientas seguro sobre tu elección.

Normalmente, existen una serie de temas más buscados y por tanto, cruciales en una documentación. Es decir, debemos asegurarnos que nuestra documentación siempre posea información sobre:

- Creación de plantillas
- Administración de estado
- Comunicación HTTP
- Procesamiento
- Validación de formularios
- Enrutado

Se trata de temas a los que día a día se enfrenta un desarrollador. No todos estos puntos tienen por qué ser positivos en el framework que elijamos, también influye mucho el problema específico al que nos tengamos que enfrentar.

2.1.3 Características ofrecidas

He aquí uno de los puntos a tratar más específicos e importantes. ¿De qué serviría elegir el framework más popular y con más documentación si no posee las características que se ajustan a nuestro proyecto a desarrollar?

Por ese motivo, es importante hacer un análisis previo a desarrollar un proyecto y obtener un listado de características fundamentales que deba tener nuestro framework elegido. Para ello, vamos a definir brevemente las características de aquellos frameworks que estamos comparando.

Comenzaremos con **React y Vue**. No son realmente frameworks; sólo representan la capa de vista de su aplicación. Significa que todas las otras partes como comunicación HTTP, validación de formulario, etc

dependen 100% de la mano del programador. Esto puede ser un arma de doble filo, ya que podemos terminar construyendo nuestro propio framework personalizado, algo muy costoso en tiempo.

Por el contrario, **Ember** posee casi todas las características. Pero, sorprendentemente, el núcleo de Ember no proporciona procesamiento de formularios avanzado, algo muy utilizado y crucial en nuestro proyecto.

Por otro lado, **Angular** es un framework rico en funciones. Al igual que Ember, posee una gran cantidad de módulos, por lo que tiene una amplia variedad de herramientas a nuestra disposición. Además, su uso intensivo de observables de la biblioteca Rx es realmente útil. Casi cualquier cosa puede ser representada como observable y, esto hace que se le pueda aplicar operaciones de alto nivel como mapas, filtros, etc.

A continuación, se muestra un resumen de las características principales de los cuatro frameworks que estamos discutiendo:

Tabla 2-1. Comparación frameworks actuales

	Angular	React	Ember	Vue
Vistas y plantillas	Sí	Sí	Sí	Sí
Procesado de formularios	Sí	Sí	Sí	Sí
Validación de formularios.	Sí	No	Sí	No
Enrutado	Sí	No	No	No
Comunicación HTTP	Sí	No	Sí	No

2.1.4 Usabilidad

El siguiente aspecto a tratar será la usabilidad de un framework, es decir, su utilidad según nuestras condiciones y el proyecto en el que lo implementemos. Puede que un framework sea una buena opción para una persona gracias a la formación previa que posea sobre él. Por el contrario, puede que sea un poco diferente a lo que está acostumbrada y suponga un gran reto.

Por tanto, es muy importante probar varios frameworks y decidir, de forma totalmente subjetiva, cuál se adecua más a nosotros y con cuál nos sentimos más cómodos.

Una buena forma de familiarizarse con un framework es utilizarlo en algún pequeño proyecto. Esto brinda la oportunidad de resolver problemas cotidianos, genéricos y previamente resueltos por otros desarrolladores. Una vez inmerso en ese pequeño proyecto, un buen método para determinar si un framework es usable es hacerte ciertas preguntas a la vez que trabajas con él.

Mientras trabajas en tu proyecto, reflexiona si está siendo un desarrollo productivo o no. ¿Cómo de fácil es lograr el resultado que deseas? ¿Tienes que buscar muchas librerías externas? Tal vez necesites algunos complementos

de la comunidad. ¿Hay alguna estructura o directrices convencionales dentro del contexto del framework? ¿Existe un CLI para acelerar el proceso de desarrollo?

En este aspecto Ember se considera un framework muy productivo, al menos para sus usuarios principales. Viene con un CLI, realmente útil. Puede generar rutas, controladores, componentes y modelos con sus propios conjuntos de test. Hacer todo eso manualmente sería una tarea bastante tediosa.

Pero, existe un inconveniente, **Ember** es muy obstinado; a pesar de todas estas ventajas, puedes acabar frustrado al tratar de realizar tareas comunes porque todo debe desarrollarse a "modo Ember", de forma muy poco genérica y objetiva.

Por otro lado, **React** y **Vue** también tienen sus tipos de CLI, create-react-app y vue-cli. Pero además de generar un proyecto inicial con algunas opciones, no ofrecen mucho en comparación con Ember o Angular

Por último, vamos a destacar una de las principales ventajas de **Angular** en este punto y es que con TypeScript todo es más sencillo: nosotros mismos definimos los diferentes tipos y esperamos los datos apropiados.

2.1.5 Facilidad de integración (con otras librerías)

Por último, pero no menos importante, hablaremos de la facilidad de un framework para instegrarse con otras librerías.

No importa qué tan rico en funciones sea el framework seleccionado, es probable que nos enfrentemos a problemas donde se necesitan herramientas adicionales. Existen múltiples librerías enfocándose en numerosos tipos de problemas, ya sea manipulación de DOM, procesamiento de datos, formateo de tiempo, edición de texto enriquecido, etc. Si intentas integrar una de estas librerías y te conlleva muchas horas, quizás ese framework no sea la mejor opción.

¿Cómo funcionan en este aspecto los frameworks que estamos comparando?
En el caso de **Angular**, framework basado en TypeScript, no todas las librerías admiten este lenguaje, pero lo positivo es que debido a su popularidad, la mayoría de veces no necesitarás ninguna librería, gracias a que casi todo se puede resolver usando sus propias herramientas.

En el caso **Vue** y **React**, el propio desarrollador es responsable de casi todo, y el uso de otras librerías no es una excepción.

Tras este análisis basado en ámbitos, a continuación, se describirán los tres principales frameworks según su uso actual:

a) Angular

En primer lugar, presentemos a nuestro elegido, Angular, y las numerosas ventajas que nos decantaron a usarlo.

Podemos enfocar Angular como una de las tecnologías front-end ampliamente difundidas en 2020 desarrollada por un gran equipo de Google.

En un primer momento se lanzó la versión 1 de Angular: AngularJS, y un tiempo después fue lanzado Angular como lo conocemos hoy. Es decir, Angular como un marco completamente nuevo con una nueva lógica y características para abordar los nuevos desafíos del desarrollo web. Este lanzamiento creó una gran revuelta en la comunidad de desarrollo, ya que la ausencia total de compatibilidad con AngularJS causó una gran controversia. Aquellos que confiaron en Google y la comunidad Angular no pudieron simplemente actualizar su aplicación a la nueva versión de marco front-end de Google para obtener todas las nuevas herramientas y características de desarrollo, si no que tuvieron que reescribir la aplicación desde cero con Angular.

Figura 2-2. Angular

A pesar de que los usuarios de Angularjs presentaros numerosas quejas, este fracaso se abordó y todas las versiones de Angular desde entonces tienen compatibilidad con versiones anteriores, lo cual es muy importante para poder tenerlo en cuenta en nuestra lista de marcos de front-end más destacables en 2020.

Angular es una solución que lo abarca todo y su principal virtud es que los desarrolladores pueden centrarse más en realizar sus tareas en lugar de buscar librerías y soluciones para éstas (personalmente, es una de las cosas que más agradezco como programadora de Angular). Esto es definitivamente lo que hace que Angular sea uno de los mejores marcos de interfaz de usuario actualmente.

Además, Angular se creó sobre TypeScript, que ofrece numerosas ventajas: funciones de flecha, asincronía, sintaxis de clase, etc. En eneral, Typescript se define como Javascript mejorado.

Por último, destacar que Angular ha sido creado para el trabajo en grandes equipos, ya que aprovecha los módulos para que cada parte del equipo pueda trabajar en su parte del código independiente sin tener miedo de romper nada en el código de otra persona. Esto también reduce la carga para la superposición de trabajo y el control de calidad. Es decir, es perfecto para que trabajen más de una persona a la vez, como es el caso de cualquier proyecto profesional o educativo.

Angular es un framework con numerosas virtudes, pero al igual que el resto, posee sus pros y sus contras.

La primera ventaja de Angular radica en su arquitectura. Posee una arquitectura basada en elementos que permite crear la interfaz de usuario con partes individuales (componentes) y reutilizarlos en la aplicación. Los elementos también simplifican las pruebas y el mantenimiento del usuario.

Otra de sus ventajas más importantes está basada en el lenguaje sobre el que está soportado. TypeScript es el lenguaje central para Angular y compila a JavaScript, lo que facilita el proceso de codificación para muchos ingenieros debido a su navegación mejorada y servicios de refactorización.

En tercer lugar y una de las ventajas más importantes para muchos es su alto rendimiento. Angular Universal Support ayuda a procesar las aplicaciones en un servidor, y Google ya ha creado el conjunto de herramientas para este propósito en particular. El alto rendimiento también está predeterminado por el Soporte a largo plazo de Google (LTS) que mantiene el ecosistema angular y lo desarrolla.

Por otro lado, aunque esta vez se trata de una ventaja bastante específica, Angular Material es una herramienta de Angular de uso opcional que optimiza la ingeniería de la interfaz de diseño de materiales, es decir, ofrece componentes diseñados que el equipo de Angular actualiza constantemente.

Por último, Ecosistema grande: los conocidos recursos de angular incorporan entornos de interfaz de usuario, IDE, herramientas de análisis, instalaciones para ASP.NET, etc.

Ahora bien, Angular también posee algunos inconvenientes como la migración que algunos tienen que hacer de AngularJS a Angular, su complejidad (a pesar de la estructura basada en la web, es bastante difícil administrar

los componentes: por ejemplo, los desarrolladores deben tener varios archivos para un componente angular y mantener las interfaces del ciclo de vida de los elementos) o la dificultad de su aprendizaje debido a la posesión de elementos complejos como la inyección de dependencias.

b) React

React es el principal marco front-end creado y desarrollado por Facebook. El equipo quería construir una interfaz de usuario que les permitiera obtener un alto rendimiento. Por tanto, Jordan Walke propuso crear la librería basada en la combinación de XHP y JavaScript. Como resultado, obtuvieron una librería para construir interfaces web con JavaScript donde las actualizaciones se hacen al mismo tiempo que los clientes usan el chat.

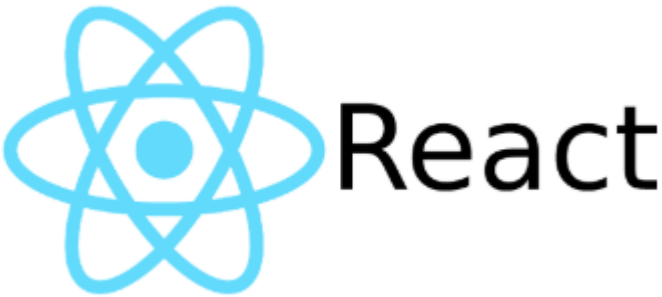

Figura 2-3. React

Son varias las características distintivas de React:

En primer lugar, funciona con el DOM virtual y aplica copias abstractas del DOM real. Por lo tanto, actualiza todos los cambios del usuario, pero no tiene efecto en otras partes de la interfaz. En segundo lugar, este marco front-end web hace que las actualizaciones sean bastante rápidas, ya que React conecta DOM a la funcionalidad de la interfaz de usuario.

Por otro lado, tiene la capacidad especial de reutilizar los componentes de código de cada nivel sin necesidad de modificarlos.

En tercer lugar, el flujo de datos en una dirección proporciona un código estable. Al aplicar las actualizaciones, solo se modificarán los componentes elegidos.

Además de Facebook, empresas como Instagram y Netflix han adoptado React, que optimizó su rendimiento web y esfuerzos de desarrollo para aumentar la eficiencia.

React se caracteriza por gran cantidad de ventajas que hacen de él un framework muy usado por todos los desarrolladores de frontend. En primer lugar, posee una librería de código abierto con gran variedad de herramientas en la que además, se lanzan actualizaciones para toda la comunidad.

En segundo lugar, cabe destacar que React trata con componentes aislados, lo que permite reutilizarlos en cualquier momento y ahorrar gran cantidad de tiempo. Por otro lado, contiene las ventajas del virtual DOM, es decir, ayuda a actualizar los cambios de cualquier usuario sin la interferencia de las otras partes mediante la aplicación de componentes aislados, por lo que ofrece una mejor experiencia al usuario y al desarrollador.

Por último, a la hora de programar, el trabajo directo con cada componente requiere un flujo de datos en una dirección y hace que el código sea realmente estable.

Puesto que no todo iba a ser bueno, React también presenta pequeños inconvenientes. Entre ellos, el más

importante puede ser que presenta una curva de aprendizaje relativamente larga, es decir, es poco probable que se aprenda en un día. Requiere un poco más de tiempo para estar al tanto de todos los detalles que, por ejemplo, Angular.

Otra desventaja respecto a otros frameworks puede ser la falta de documentación debido al desarrollo de alto ritmo. La popularidad de este marco front-end web es extremadamente mayor que la de otros, por lo que tiene tantas actualizaciones e innovaciones que a veces es difícil encontrar toda la información detallada.

A pesar de estas ventajas es un framework altamente recomendable. Entonces, ¿Por qué no hemos elegido React para nuestro proyecto?

Muy sencillo, no es un framework 100% independiente de backend como nuestro elegido, Angular. Además, su mayor eficiendia recae en webs con grandes recargas (no es nuestro caso).

c) Vue.js

Vue.js es el marco web para construir interfaces de usuario. Es una herramienta independiente que crea interfaces web y no requiere la extensión adicional. Vue.js fue creado por Evan You y lanzado inicialmente en febrero de 2014.

Muchas características accesibles atraen al desarrollador para usarlo. Vue.js es fácil de aprender e integrar. Vue.js puede aplicarse tanto para componentes representados como para aplicaciones completas de una página. Además, incluye diseño de archivo de componentes y estructura lógica.

Este marco de JavaScript trata el enlace de datos reactivo bidireccional y no exige librerías adicionales.

Figura 2-4. Vue.js

Al mismo tiempo, es extremadamente flexible. Vue.js puede integrarse fácilmente con diferentes librerías y aplicarse para proyectos más grandes. El equipo de Vue.js ha lanzado algunas excelentes extensiones de devtools del navegador para su marco. Éstas permiten verificar la situación actual de los componentes y cambiar manualmente algunos. Los desarrolladores se sienten atraídos por la posibilidad de combinar la estructura de la manera que han elegido.

Un detalle importante que no debemos pasar por alto es el tamaño de Vue.js, ya que es bastante pequeño en comparación con otros. Como resultado, esto acelera el tiempo de rendimiento general.

Por último, Vue.js es conocido por su documentación con todo incluido, algo que hace que los desarrolladores ahorremos tiempo aprendiendo este marco front-end. Para inciarse en Vue.js es crucial comenzar con JavaScript y HTML básicos, ya que Vue.js aplica patrones de plantilla basados en HTML.

En cuanto a los pros y contras de Vue.js podemos destacar algunos como su claridad y simplicidad a la hora de usarlo (este marco front-end tiene el área de superficie API más pequeña. Es bastante fácil comenzar no solo a aprender sino a usarlo en poco tiempo), la facilidad para encontrar documentación necesaria que esté actualizada y bien escrita o la simplicidad que ofrece a la hora de integrarlo (debido a su estructura lógica, los desarrolladores

de la interfaz de usuario tienen la capacidad de crear componentes flexibles y reutilizarlos más tarde, incluso en los otros proyectos).

Otra de sus ventajas puede ser que Vue.js no exige ninguna librería adicional, pero posee extensiones de devtools del navegador. Por lo tanto, pueden cambiarse componentes separados manualmente cuando sea necesario.

Por otro lado, algunas de sus desventajas están relacionadas con sus propias ventajas, como la excesiva flexibilidad que ofrece. Algunos expertos han señalado que mucha flexibilidad no es buena. Todos los ingenieros de front-end web pueden contribuir a su desarrollo, por lo que puede causar ciertas irregularidades.

Además, posee una comunidad de desarrolladores más pequeños; ya que es un marco relativamente nuevo y en constante mejora.

¿Por qué hemos elegido Angular y no Vue.js?

Como comentaba al principio, es primordial que un framework que vamos a usar para un proyecto fin de carrera haya sido muy conocido, usado y probado con anterioridad. Ésta es una de las características que Vue.js no tiene respecto a Angular. Además, personalmente, la primera desventaja de Vue.js en este documento nombrada (excesiva flexibilidad) es crucial desde mi punto de vista. Puede hacer que utilicemos un código aparentemente fiable y que no lo sea.

2.2 Framework elegido

Además de las numerosas ventajas descritas en las secciones anteriores, este framework ha sido el elegido por ocho razones principales:

1. Es gratuito

Angular es un framework JavaScript, gratuito y Open Source, creado por Google y destinado a facilitar la creación de aplicaciones web modernas de tipo SPA (Single Page Application).

2. Ofrece una amplia base de serie

Dado que Angular es un framework, ofrece muchas más opiniones y funcionalidades de serie que una simple librería. Con otro software similar, lo más común es tener que tirar de varias bibliotecas de terceros a la hora de desarrollar una app. Lo más probable es que se necesiten algunas adicionales para hacer el enrutado, para la gestión de dependencias, para realizar llamadas a APIs REST, para hacer el testing, etc...

Algo bueno en Angular, es que ofrece más opciones de serie, ayudándote a arrancar un proyecto sin necesidad de pararte meses para la toma de decisiones. Es decir, con Angular ya sabes desde el primer momento cómo organizar el código, cómo se realizan las diferentes tareas que necesitas, la arquitectura de la aplicación, ya que tiene un "modus operandi" muy común entre todos sus desarrolladores.

3. TypeScript, la versión mejorada de JavaScript

Aunque se puede programar en ECMAScript puro, el equipo de Angular decidió en su momento que haría todo el desarrollo con el lenguaje TypeScript, y el 95% de la documentación y los ejemplos que encuentras por en internet utilizan este lenguaje.

A mucha gente esto le parecerá un error, pero el criterio que siguió el equipo de Angular sobre la variedad de JavaScript a utilizar tiene muchas ventajas.

Una de las primeras es la consistencia en la documentación. Si navegas por internet intentando encontrar

ejemplos y tutoriales de otras librerías de JavaScript vas a encontrar de todo, pero la única constante es la inconsistencia que existe. Con TypeScript esto no pasa, ya que toda la sintaxis y la manera de hacer las cosas en el código es la misma, lo que añade coherencia a la información y a la forma de leer el código.

Esta consistencia ayuda a evitar la confusión y la sobrecarga en la toma de decisiones derivadas de empezar con Angular, es decir, lo que ya comentamos en el punto anterior sobre el ahorro de tiempo al comenzar un nuevo proyecto con Angular.

4. Reutilización gracias a componentes web

Un componente en Angular es una pequeña parte de código que reutilizable en otros, lo que permite un desarrollo ágil y un gran ahorro de tiempo. Es decir, podemos tratar a dichos componentes como un puzle de componentes que encajas de maneras distintas según te convenga.

Otro punto para destacar es que los componentes que creas en Angular son fáciles de convertir en componentes web nativos**.** A largo plazo esto es una gran ventaja pues permitirá reutilizar componentes creados en Angular en otro tipo de aplicaciones. Cuanta más reutilización de nuestro trabajo, mucho mejor.

5. Un futuro estable en tu aplicación

Uno de los grandes problemas del mundo JavaScript actual es que hay tantos cambios y tan frecuentes, que todo el mundo acaba saturado intentando seguir el ritmo de las novedades. De hecho, es uno de los motivos por los que mucha información actual de internet está obsoleta.

Angular, por el contrario, siempre ha sido menos propendo a los grandes cambios en las migraciones d eversiones (exceptuando el paso de Angularjs A Angular hace unos años). Es decir, podemos asegurar que Angular posee un equipo detrás que está continuamente tomando decisiones meditadas sobre el futuro, lo que evita las prisas y errores que caracterizan a otras librerías y frameworks.

Además, en otras librerías y frameworks, debido al gran uso de librerías de terceros se dispone de mucha lógica que no depende solo de nosotros mismos y está en continuo riesgo de cambios de versiones o incompatibilidades.

En conclusión, con Angular tenemos un framework de duración a largo plazo y es uno de los motivos por los que se ha elegido Angular para este proyecto, ya que la idea principal será que futuros compañeros lo continúen mejorando y cumplimentando sin necesidad de encontrarse con problemas provocados por la tecnología.

6. Gran soporte de herramientas

Cuando un desarrollador programa, casi nunca lo va a hacer en un editor de texto plano. Usará editores avanzados, IDEs y otras herramientas relacionadas. Por ejemplo, en nuestro caso concreto hemos usado Visual Studio Code, una herramienta de la que hablaremos más adelante y a la que se le aplican varios plugins consiguiendo adoptar así buenas prácticas en la programación.

Las plantillas de Angular almacenan por separado el código de la Interfaz de usuario (archivos html) y el de la lógica de negocio (archivos ts), por lo que se le puede sacar partido a las muchas herramientas ya existentes para editar este tipo de archivos. Otros frameworks como React, por ejemplo, mezclan en un mismo archivo todo el código. Si bien esto puede tener sus ventajas, dificulta el uso de herramientas estándar de desarrollo.

Además, gracias a la popularidad de Angular, los principales editores e IDEs ofrecen ya extensiones para

poder trabajar con este framework de la manera más cómoda posible.

7. Angular permite un buen trabajo en equipo

La plataforma está diseñada para permitirnos compartir código y dividir el trabajo entre distintos roles (diseñadores, ingenieros, analistas, programadores, testers, etc). El modelo basado en componentes utilizado en Angular ha sido diseñado para separar todos estos roles, y permitir la participación en desarrollos colaborativos**.**

8. Angular tiene detrás un soporte fiable

Debido a que Angular es un producto de Google, es capaz de aprovechar la infraestructura de pruebas del gigante de Internet. Cada cambio que se hace en Angular se valida contra cada proyecto Angular dentro de Google. Esto significa que antes de que cualquier versión se libere públicamente, el framework ya está en uso en cientos de proyectos, maximizando la posibilidad de que no haya cambios de prototipo o retrocesos no intencionados.

2.2.1 Versión elegida

Actualmente, Angular se encuentra en la versión 10.0.2, si contamos solo versiones estables. Sin embargo, nosotros hemos decidido usar la versión 8.0.0 por varias razones.

Figura 2-5 . Angular versión 8

Angular v8 añade a versiones anteriores una gran cantidad de funcionalidades para navegadores modernos. Entre ellas, mejoras para las rutas, el núcleo, el compilador, el editor de estilos, rendimiento en el procesamiento de imágenes, sintaxis, compilador, API pública, formularios, uso del renderizador Ivy y mejoras en rendimiento.

Vamos a listar algunas de las ventajas técnicas y añadidos que se ha hecho en la versión 8 de Angular respecto a otras:

- Typescript 3.4 contiene una flag llamada --incremental que detecta la forma menos costosa de verificar y emitir cambios, eventos en un proyecto Angular.
- Angular CLI está mejorando continuamente. Ahora, los comandos ng build, ng test y ng run están equipados internamente con librerías y herramientas de terceros.

- Ya no tendremos una forma específica de importar *lazy modules* (módulos de Angular configurados para cargarse en el momento necesario y no siempre al inicio de la aplicación, lo que provee de una gran eficiencia), sino una forma estándar mucho más cómoda.

  Lo que antes era:

  { path: '/student', loadChildren: './student/student.module#StudentModule' }

  Ahora es

  { path: '/student', loadChildren: () => import('./student/student.module').then(s => s.StudentModule) }

- JavaScript tiene un solo subproceso, por lo que es común que las tareas más críticas, como las llamadas de datos, se realicen de forma asincrónica. Angular 8 implementa Web Workers, que facilita la ejecución de los cálculos intensivos de CPU en el subproceso en segundo plano, liberando el subproceso principal para actualizar la interfaz de usuario. Por tanto, ganamos eficiencia.
- Se ha implementado una corrección de errores en la que se ha aumentado el límite de memoria de nodos para ng-module para evitar problemas de memoria insuficiente con módulos grandes. Cuando la memoria no se libera correctamente o un proceso continúa usando más y más, puede producirse un fallo en el proceso.

- Angular Router
  Se añade un modo de retrocompatibilidad para simplificar la actualización del path de proyectos grandes, permitiendo usar $route para los lazy loadings de AngularJS.
- Carga Diferencial de JavaScript
  Generar un proyecto con el CLI producirá paquetes de Javascript antiguo (ES2015) y moderno (ES2015+), esto ayudará a los navegadores modernos a cargar más deprisa las páginas al tener ya el compilador para ES2015+.
- Rutas
  Se ha agregado una opción de navegación basada en hash a *setUpLocationSync*. Con este cambio, los desarrolladores ahora pueden pasar una opción a setUpLocationSync para garantizar que los cambios de ubicación se ejecuten en aplicaciones basadas en hash.
- CLI
  Se ha agregado soporte para tokenizar un subconjunto de una cadena de entrada y tokenizar cadenas de escape. CLI ya no soporta símbolos externos por defecto.
- CSS
  Con el respaldo de SASS para Bazel, las reglas se agregan al área de trabajo para un proyecto que requiere la extensión SASS a CSS. Con SASS, los desarrolladores pueden escribir estilos visuales para un sitio web en un lenguaje más avanzado que se compila en CSS.
- Ivy
  Con la versión 8 podemos elegir entre el motor Ivy o el View Engine clásico para generar los proyectos, aunque Ivy no será viable para todos los usos, ya que su lanzamiento definitivo será en versiones posteriores.

  Las ventajas de Ivy en esta versión son:

  - Hacer que el código angular sea más fácil de depurar, incluso a medida que las aplicaciones crecen.
  - Mejora de la comprobación de tipos de plantillas.
  - Mejora de la retrocompatibilidad.

Resumen de mejoras:

- Mejora del rendimiento interno del framework
- Mejora del compilador
- Facilidades para las migraciones de versiones anteriores
- Soporte y generador de webworkers
- Mejoras en los Service Workers
- Mejoras en Ivy
- Mejoras en los Formularios
- Nuevas funcionalidades en el routing de Angular
- Cambio en las versiones de las dependencias de Angular
- Cambios en HttpClient y RXjs
- Cambio en la versión de node, nos sugieren la 12
- Se incluye Browserslist

Por último, cabe destacar que tras una investigación, observamos que a pesar de ser una versión inferior a la actual, la versión 8 es la más descargada actualmente. Podemos observar la siguiente gráfica donde tenemos número de descargas por usuario en el eje x y fechas en el eje y. Puede observarse que a finales de 2019 la línea amarilla (perteneciente a angular 8) comenzó a superar incluso al cláseco y altamente utilizado AngularJS. Así ha continuado hasta hoy, en 2020, siendo la versión más usada en proyectos de todo tipo.

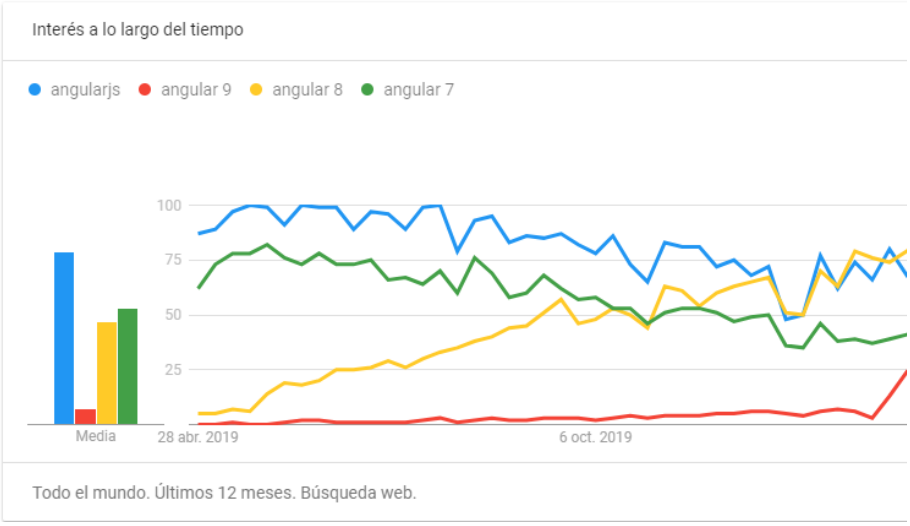

Figura 2-6. Versiones mas descargadas de Angular

3 SOLUCIÓN TÉCNICA

Vale más actuar exponiéndose a arrepentirse de ello,
que arrepentirse de no haber hecho nada

Giovanni Boccaccio

En este apartado se explicará de forma muy detallada la solución técnica adoptada para poner resultado a un problema que venía aconteciendo desde hace bastante tiempo, el constante enfrentamiento a sistemas tedioso y no automatizados.

Consistirá básicamente en una plataforma frontend-backend para la automatización de caracterización electrónica.

En primer lugar vamos a explicar a alto nivel la funcionalidad de nuestra web, abordaremos los distintos elementos que contiene la web y como deben utilizarse, detallaremos el flujo que debe seguir un usuario y ahondaremos un poco en la arquitectura de la web y las librerías usadas.

Por último, presentaremos una breve descripción del backend implementado con el objetivo de aportar una visión global del sistema propuesto. No obstante, su diseño y desarrollo está fuera del alcance de este proyecto.

3.1 Estructura del proyecto

Todas las funcionalidades del frontend ofrecidas en este proyecto están estructuradas mediante módulos y componentes, una estructura muy útil y ordenada que provee Angular.

Consiste en un módulo padre llamado “app” conteniendo dos componentes principales “Inicio1” (que dentro contiene a los componentes “Configuración” e “Informe”) e “Inicio2” que contiene formularios y muestra de datos resultados.

Además contiene “Header” y “Footer”, componentes correspondientes a la cabecera y pie de la plataforma.

A su vez, dentro del componente “Informe” podemos encontrar los componentes “Chartjs” y “ChartjsOrigen”, correspondiente a los dos tipos de gráficas que vamos a visualizar.

A continuación, lo explicaremos con más detalle:

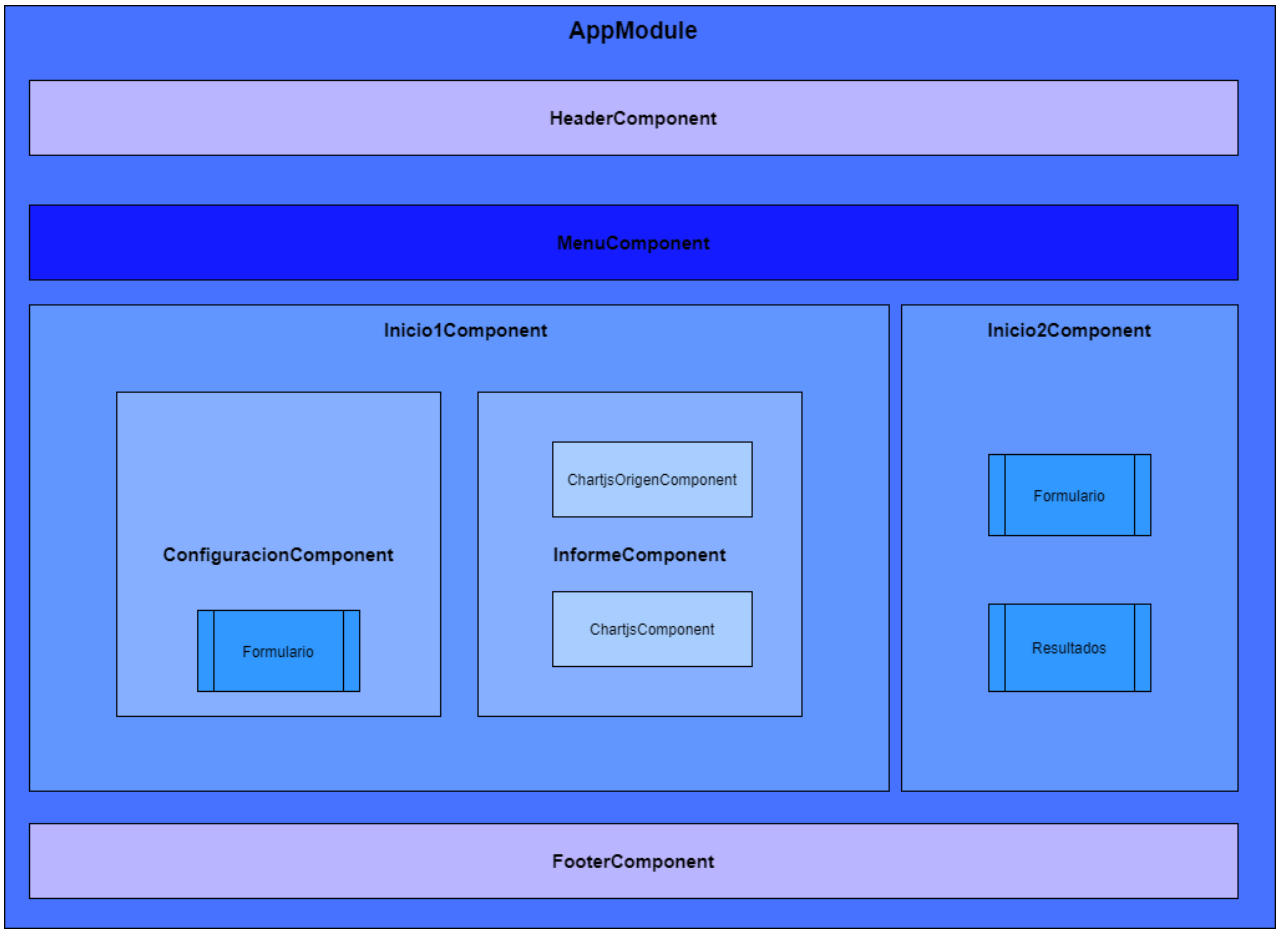

Figura 3-1. Estructura del proyecto

AppModule es el módulo principal de la aplicación. En otro tipo de aplicaciones de mayor envergadura suele ser necesario definir más de un módulo (PrivateModule, PublicModule, SharedModule, etc), pero en nuestro caso un solo módulo permite una estructura organizada de nuestra aplicación.

Podemos reconocer nuestro AppModule mediante la etiqueta @NgModule y contiene:

- Declaraciones: todos aquellos componentes que quieran usarse dentro de un módulo en angular, deben declararse en el array "declarations" del mismo.
- Importaciones: aquellos módulos que quieran usarse en la aplicación, deben importarse en el array "imports". En nuestro caso hemos importado:
    - **BrowserModule**: módulo necesario si queremos ejecutar nuestra aplicación en un navegador.
    - **AppRoutingModule**: se trata de un módulo crucial, ya que engloba todo el enrutado de la aplicación, es decir, los paths existentes y los componentes a los que corresponden su visualización.

      En una aplicación de una sola página, constantemente cambiamos lo que ve el usuario al mostrar u ocultar partes de la pantalla que corresponden a componentes particulares, en lugar de salir al servidor para obtener una nueva página.

      A medida que los usuarios realizan tareas de aplicación, necesitan moverse entre las diferentes vistas que ha definido. Para implementar este tipo de navegación dentro de la única página de

su aplicación, es necesario usar el enrutador angular.

Por otro lado, para gestionar la navegación de una vista a la siguiente, también debemos el enrutador angular. El enrutador permite la navegación interpretando la URL de un navegador como una instrucción para cambiar la vista, donde se puede agregar o quitar controladores en tiempo de ejecución.

Gracias a este módulo podemos usar RouterLink, .forRoot() y .forChild(). Funciones muy importantes en nuestro proyecto para importar componentes hijos dentro de padres y hacer distintos enrutados.

- **ReactiveFormsModule**: uno de los componentes principales de nuestra web es el formulario de configuración, correspondiente al módulo de reactive forms de Angular, uno de los más famosos y usados por todos los desarrolladores. Se importa a través de @angular/forms.

  Los formularios reactivos utilizan un enfoque explícito e inmutable para administrar el estado de un formulario en un momento dado. Cada cambio en el estado del formulario devuelve un nuevo estado, que mantiene la integridad del modelo entre cambios. Los formularios reactivos se construyen alrededor de flujos observables, donde las entradas y los valores de los formularios se proporcionan como flujos de valores de entrada, a los que se puede acceder síncronamente.

  Los formularios reactivos también proporcionan una ruta directa a las pruebas porque tienen la seguridad de que sus datos son consistentes y predecibles cuando se solicitan. Cualquier consumidor de las transmisiones tiene acceso para manipular esos datos de manera segura.

  Los formularios reactivos difieren de os formularios basados en plantillas de distintas maneras. Los formularios reactivos proporcionan más previsibilidad con acceso síncrono al modelo de datos, inmutabilidad con operadores observables y seguimiento de cambios a través de flujos observables.

  Los formularios basados en plantillas permiten el acceso directo para modificar datos en su plantilla, pero son menos explícitos que los formularios reactivos porque dependen de directivas incrustadas en la plantilla, junto con datos mutables, para rastrear cambios de forma asíncrona.

- **HttpClientModule**: módulo correspondiente a la librería http de angular 8: @angular/common/http. Éste módulo permite hacer todo tipo de peticiones get, post, put y delete mediante el protocolo http a backend.

  La mayoría de las aplicaciones front-end necesitan comunicarse con un servidor a través del protocolo HTTP, para descargar o cargar datos y acceder a otros servicios back-end. Angular proporciona una API HTTP de cliente simplificada para aplicaciones angular, la clase de servicio HttpClient en @ angular / common / http.

  El servicio de cliente HTTP ofrece las siguientes características principales.

  - La capacidad de solicitar objetos de respuesta escritos.
  - Manejo simplificado de errores.
  - Características de testabilidad.
  - Solicitud y respuesta de intercepción.

- **NgbModule**: módulo que te provee Bootstrap para Angular, una librería que facilita el diseño, permite personalizar rápidamente vistas responsive para dispositivos móviles y webs con Bootstrap. Además, proporciona el kit de herramientas de código abierto front-end más popular del mundo, que presenta variables y mixins Sass, sistema de cuadrícula sensible, componentes

pre compilados extensos y complementos potentes de JavaScript.

- Providers: se trata de instrucción para el sistema de Inyección de dependencias sobre cómo obtener un valor para una dependencia. La mayoría de las veces, estas dependencias son servicios que nosotros mismos creamos y proporcionamos. En nuestra aplicación GlobalService es un servicio bastante importante que incluimos en nuestros providers.
- Bootstrap: en esta instrucción tenemos incluidos AppComponent e InicioComponent, los dos componentes principales. En esta instrucción debemos declarar los componentes boostrap que tengamos en nuestro proyecto, por eso incluimos nuestro componente principal InicioComponent.

Tas entender nuestro módulo principal, nos centramos en sus componentes. El resto de componentes tendrán la misma estructura:

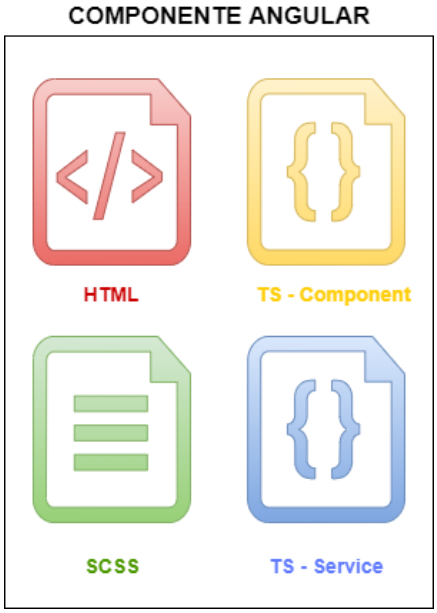

Figura 3-2. Ficheros componente Angular

La estructura de un componente en Angular es siempre la misma. Dentro de un directorio /nombre tendremos los archivos: nombre.component.ts, nombre.component.html, nombre.component.scss y en ocasiones nombre.service.ts.

- Nombre.component.ts contendrá la lógica, las funciones typescript que deseemos ejecutar.
- Nombre.component.html contendrá toda la estructura html incluyendo la inyección de componentes hijos.
- Nombre.component.scss contendrá las clases de estilo que definamos y a las que haremos referencia en el html.
- Nombre.service.ts contendrá la definición de las llamadas http a backend.

En referencia al archivo Nombre.component.scss, cabe destacar que, como puede observarse, no se trata de la extensión normal de css (archivo.css).

Esto es debido a que SCSS es la extensión de estilo correspondiente a SASS. En Angular suele usarse SASS (*Syntactically Awesome Stylesheets*) en lugar de CSS (*Cascading Style Sheets*) como vemos habitualmente en otros frameworks.

La diferencia principal entre uno y otro es que CSS es un lenguaje de hojas de estilo, sirve para organizar la presentación y aspecto de una página web, ya sea en un ordenador, un móvil, una Tablet, etc.; mientras que

SASS se define como un CSS muy avanzado. Consiste en un metalenguaje de hojas de estilo basado en dos sintaxis:

La original, ".sass", que usa la indentación para separar bloques de código y el carácter nueva línea para separar las reglas.

La sintaxis más reciente, ".scss", que usa el formato de bloques como CSS, usa llaves para denotar bloques de código y punto y coma para separar las líneas dentro de un bloque.

Una vez aclarada la estructura genérica de un componente en Angular, hagamos un repaso de los componentes principales creados en este proyecto:

- *App.component*: componente padre del proyecto. Suele existir en toda app Angular y contiene a los elementos hijos. En este caso:
    - *Header.component*: component que contiene una imagen de fondo a modo de banner y el título del proyecto, de la web.
    - *Footer.component*: componente que contiene iconos, enlaces o algún texto identificativo de pie de página.
    - *Menu*.component: corresponde al menú principal que vemos en nuestra web.
    - Inicio1.component: componente que engloba las funcionalidades de la pestaña 1 del menú.
    - Inicio2.component: componente que engloba las funcionalidades de la pestaña 2 del menú.
- *Inicio1.component*: componente más importante y con más lógica contenida del proyecto. Además de el botón "Descargar resultados", contiene los siguientes components hijos:
    - *Configuracion.component*: componente que contiene un formulario con varios campos de configuración de la señal del osciloscopio.

      Además, contiene los botones "Refrescar" y "Generar FFT". Aunque se explicarán con más detalle, el botón refrescar permitirá al usuario tener la tasa de refresco de la señal obtenida que él desee y el botón de generación de la FFT, como su nombre indica, hace una petición para que en backend se genere la señal FFT mediante las funciones matemáticas necesarias.
    - *Informe.component*: contiene dos components hijos:
        - *ChartjsOrigen.component*: contiene la señal cuadrada origen del osciloscopio.
        - *Chartjs.component*: contiene la señal origen modificada por el usuario mediante el formulario o la FFT generada y en un futuro se ampliará a cualquier tipo de señal que el usuario quiera generar a partir de la señal origen.
- *Inicio2.component*: componente que contiene un formulario conformado por selectores para decidir qué parámetros se quiere calcular y, una zona donde se muestran los resultados.
- *Inicio.service*: servicio que contiene todos los servicios con las URL de llamada a backend.

El botón "Descargar resultados" nombrado anteriormente consiste en un botón que con solo un click permite al usuario imprimir o guardar como imagen la gráfica o gráficas generadas.

Por último, ya que la señal del osciloscopio va cambiando constantemente, pensamos en dos opciones: estar escuchando constantemente o lanzar una petición para escuchar cada x segundos mediante un timeout la señal del osciloscopio, algo muy costoso a nivel de rendimiento.

Analizamos la situación y concluimos que una solución intermedia podría ser un botón de refresco de la señal que el usuario pueda pulsar cada vez que lo desee. Dicho botón lanzará una llamada a backend solicitando de nuevo la última señal que se haya obtenido en la web.

En un futuro, se podría ampliar creando otras pestañas en la web correspondiente a distintas configuraciones. Por ejemplo, un generador de señal.

3.2 Funcionalidad

Se propone una solución basada en una interacción de usuario cómoda y rápida que le permita configurar cualquier tipo de señal.

El usuario se limitará a completar un formulario con los parámetros que desee, hacer click en un botón de envío de formulario y visualizar (o imprimir, si lo desea) el resultado mostrado en una gráfica.

Es decir, un usuario entrará por primera vez en nuestra web y visualizará una cabecera y pie (como en toda web) con información sobre la US, ETSI y el nombre del proyecto. En la parte superior encontrará un menú con dos pestañas:

- Primera pestaña

  En la primera pestaña, a la izquierda se mostrará un formulario y a la derecha una gráfica representando la señal origen que se está mostrando en el osciloscopio.

  Por otro lado tendrá botones con las opciones de refrescar (cualquiera que sea la última gráfica que está visualizando), limpiar formulario (resetearlo a valores 0), imprimir un pequeño documento (las gráficas calculadas junto a los inputs de texto escritos).

  El usuario deberá completar el formulario con la configuración que desee y pulsar el botón "Configurar" de envío de formulario. Una vez hecho esto, se mostrará una segunda gráfica debajo de la gráfica original que consistirá en esa gráfica cuyos parámetros han sido modificados por nosotros mismos.

  Por último, pulsará el botón de FFT para calcular la potencia de FFT y visualizarla también en una tercera gráfica.

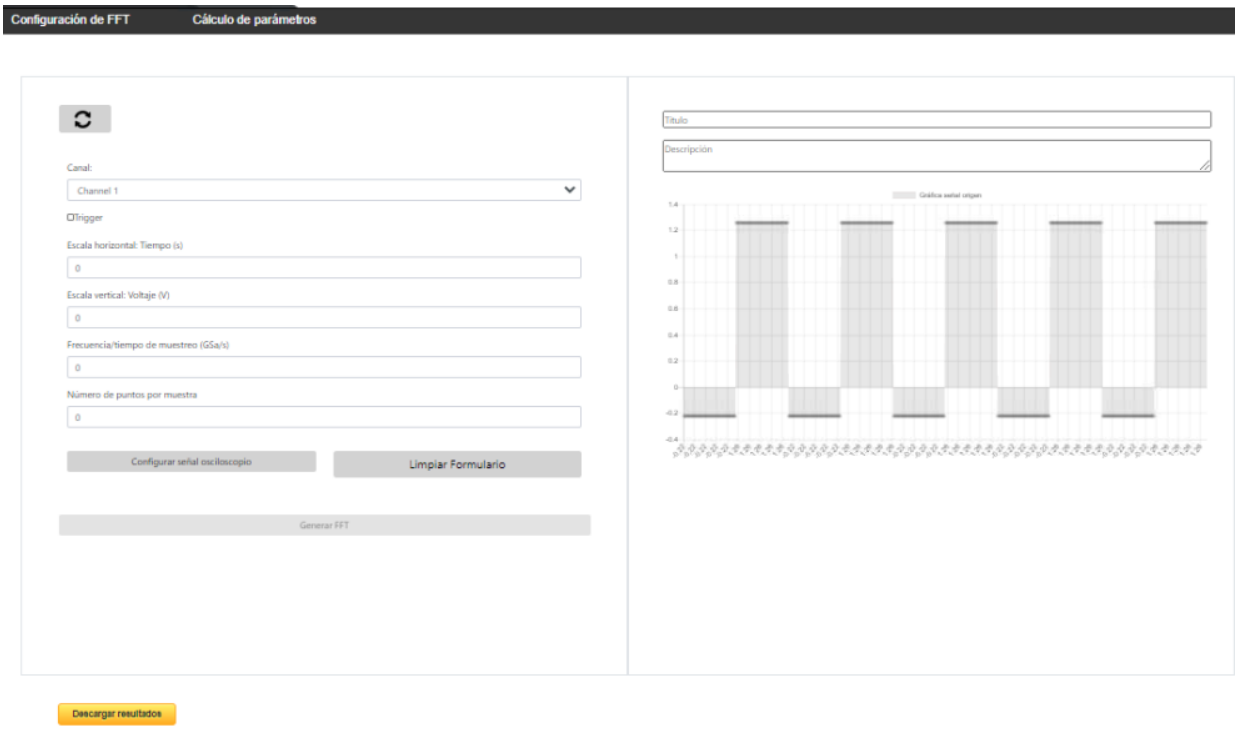

Figura 3-3. Pestaña 1

- Segunda pestaña

  En la segunda pestaña, el usuario podrá calcular parámetros como la amplitud, periodo, tensión máxima y mínima o la frecuencia de la señal deseada.

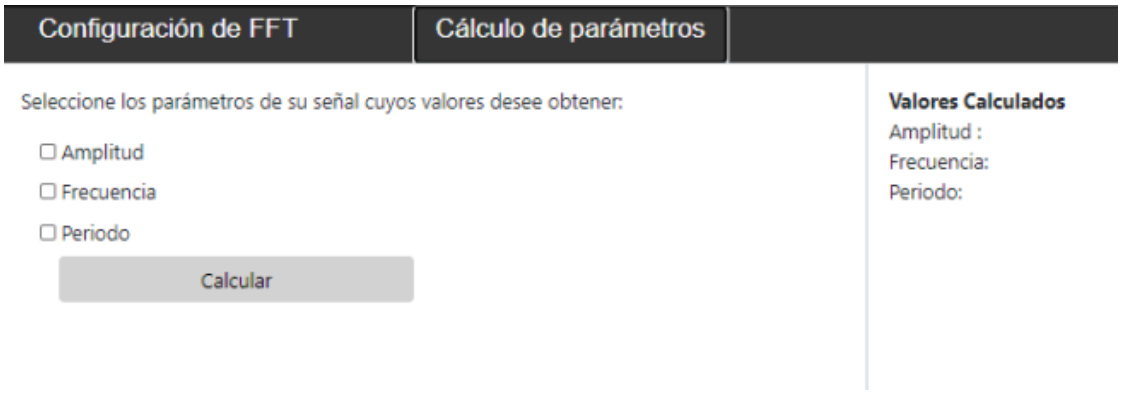

Figura 3-4. Pestaña 2

3.3 Flujo

Vamos a crear un entorno frontend – backend independiente y estructurado de tal manera que el flujo va a ser bastante claro:

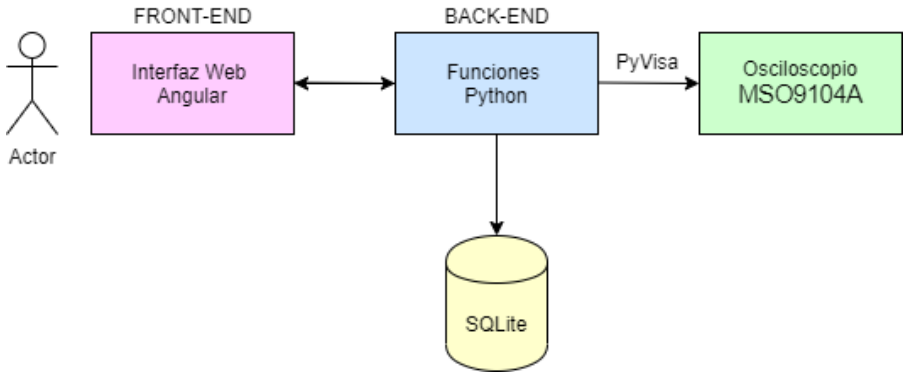

Figura 3-5. Flujo

El usuario interactuará en todo momento con la interfaz, la web programada en el framework Angular. En dicha interfaz encontrará en primer lugar, un formulario vacío a la izquierda y una señal (cuadrada) a la derecha, correspondiente a la señal inicial de nuestro osciloscopio.

En el formulario web, el usuario podrá determinar varios parámetros:

- **Canal**: indicará el canal que desea visualizar.
- **Trigger**: es un select que el usuario podrá seleccionar solo para 1 canal. Una vez seleccionado para uno de los canales y enviado el formulario, dicho select desaparecerá para evitar confusiones. De esta manera controlamos que el usuario elija un solo canal como trigger.
- **Voltaje**: indicará el voltaje deseado para su señal (eje y de la gráfica), teniendo una validación que comprueba que siempre se introduzcan números.

- **Tiempo**: indicará el voltaje deseado para su señal (eje y de la gráfica), teniendo una validación que comprueba que siempre se introduzcan números.
- **Frecuencia** por tiempo de muestreo: número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta.

Una vez enviado el formulario (botón "Configurar señal osciloscopio"), el usuario podrá observar a la derecha como la gráfica es modificada mostrando una señal configurada mediante los parámetros que él mismo a seteado.

Cuando el usuario verifique que la señal origen es la deseada podrá hacer click en un segundo botón "Generar FFT" (este botón se encuentra deshabilitado hasta que el usuario envía el primer formulario de configuración de la señal origen del osciloscopio), el cual inmediatamente mostrará una segunda gráfica a la derecha, justo debajo de la señal origen, con la FFT calculada.

A continuación, procederemos a describir la solución aportada.

3.4 Solución técnica a bajo nivel

La primera vez que el usuario accede a la web, entrará en dos componentes según la pestaña que seleccione: inicio.component.ts (primera pestaña) o inicio2.component.ts (segunda).

Dentro de la primera pestaña, cargará un método *ngOnInit()* (clase configuración.component.ts), encargado de realizar cualquier acción solo la primera vez que se entra en la web o cuando se produce una recarga (F5).

Dicho método, mostrará un formulario vacío a la izquierda y un spinner de carga a la derecha mientras hacemos la llamada a backend del servicio http://localhost:XXXX/signals/getOrigenSignal. Si la llamada no se realizara con éxito, mostraríamos un mensaje de error.

En el *ngOnInit()* también se llamará a la función que construye el formulario inicial buildFormularioConfig().

Una vez obtenida la primera señal se emitirá al servicio global (explicado con detalle más adelante) la respuesta (ejex, ejey) *globalService.signalEmitter$.subscribe(respuesta)*. De esta forma, haremos que la señal esté disponible en otros componentes de nuestra aplicación como informe.component.ts, donde se analizan esos valores y se configura chartjs para mostrar su gráfica correspondiente.

Se hará un tratamiento de los valores redondeándolos mediante funciones matemáticas y mostrándolo adecuadamente dentro del array *datasets* de cada chart definida.

El siguiente paso será

- Botón "Refrescar"

Lógica implementada para evitar estar refrescando nuestras gráficas cada x segundos y hacer de nuestro proyecto una web poco eficiente.

¿Qué hay detrás del botón Refrescar?

Tenemos una señal que comprueba cual fue la última señal obtenida, original, modificada por formulario o FFT. Volverá a llamar a su servicio correspondiente para actualizar sus valores y de nuevo se hará un EMIT al servicio global para ponerla a disposición de todos los componentes. Para ellos, se rellena un objeto con la respuesta (ejex y ejey), el tipo de señal(original, modificada o FFT) y un parámetro error que estará vacío en caso de éxito.

```
if (this.nombreGrafica == 'ORIGINAL'){
        this.respuesta = this.getOrigenSignal();
        tipo = 'ORIGINAL';
    } else if(this.nombreGrafica == 'MODIFICADA'){
        this.respuesta = this.setSignalConfig(this.valoresConfiguracion);
        tipo = 'MODIFICADA';
    } else if(this.nombreGrafica == 'FFT'){
        this.respuesta = this.generarFFT();
        tipo = 'FFT';
    }
     const compartirRes = {
        'ejes': this.respuesta,
        'nombreGrafica' : tipo,
        'error' : this.errorLlamada
        }
        this.globalService.emitAuthentication(compartirRes);
```

Función implementada para el botón "Refrescar"

El siguiente paso que realizará el usuario será completar el formulario (cada vez que rellene una casilla se procederá a hacer una validación y mostrar un mensaje de error si el formato no es correcto o supera un umbral. Para ello se usarán validaciones de tipo *Validators.required && numberValido() && Validators.min(0) && Validators.max(5).*

Si todo está correcto, se enviará el formulario haciendo una llamada al servicio http://localhost:XXXX/signals/signalConfiguration, pasando como parámetro el JSON:

```
this.valoresConfiguracion= {
        "canal" :  this.configFormulario.value.canal,
        "trigger":  this.configFormulario.value.trigger,
        "voltaje" : this.configFormulario.value.voltaje,
        "tiempo" : this.configFormulario.value.tiempo,
        "fMuestreo": this.configFormulario.value.muestreo,
        "puntos": this.configFormulario.value.muestreo
    }
```

JSON usado como parámetro entre front y back

Una vez más, desde la clase informe.component.ts se hará una suscripción al servicio global desde el que se ha emitido previamente el resultado devuelto por backend.

Por último, el usuario podrá calcular la FFT de su señal generada rellenando el número de puntos por muestra deseado y pulsando en "Generar FFT". Esto hará una llamada a http://localhost:XXXX/signals/fft pasando lo parámetros necesarios y obteniendo

- Formulario

La primera vez que enviamos el formulario con un tiempo configurado, dicho campo "tiempo" se bloqueará para el resto de canales. De esta forma, obligamos al usuario a que mantenga la misma escala de tiempo.

Respecto al select "Trigger", una vez seleccionado para el canal que deseemos y pulsado el botón "Configurar formulario", dicha opción desaparecerá para el resto de canales.

- Enviar formulario: llamada al servicio de backend y emisión al servicio global.
- Limpiar formulario: reset del formulario para poner valores a 0.

Para validar los valores introducidos en el formulario usaremos una directiva de angular number.validator.directive.ts que usará la expresión regular:

/^[.\d]+$/.test(voltajeValue) ? +voltajeValue : NaN

- Gráficas resultado

Obtendremos tres tipos de gráfica, las tres implementadas mediante una librería de javascript llamada Chartjs.

Desde la clase informe.component.js se hará continuamente *this.subscripcionGlobal = this.globalService.signalEmitter$.subscribe((res) => {}* para obtener la señal que se esté queriendo emitir desde otros componentes.

Es muy importante que cortemos esa comunicación constante con el servicio global cuando no sea necesaria, ya que de lo contrario tendríamos una constante comunicación innecesaria e ineficiente.

Para ello, también se implementará un método *ngOnDestroy()* donde nos desuscribiremos del servicio global *this.subscripcionGlobal.unsubscribe()*.

- Botón "Descargar Resultados"

Si el usuario desea conservar las gráficas generadas, tendrá la opción de descargar ese resultado mediante este

botón. Además, dispondrá de varis áreas de texto que podrá rellenar con títulos y descripciones para cada gráfica y disponer de ellos al imprimir.

¿Cómo controlamos qué elementos se imprimen y cuales no en nuestra página web?

Disponemos de una clase *printedPage.scss*, que define clases de estilo en *@media print {}* que solo se cumplirán al imprimir. De esta forma, mostramos y ocultamos lo que deseemos.

- GlobalService

Como hemos explicado anteriormente, en Angular se trabaja con estructuras muy esquematizadas: componentes padres, hijos, hermanos, etc.

A veces, un botón, un formulario o cualquier elemento se encuentran en un componente, pero el resultado de la lógica de dicho elemento es necesario para otro componente. En nuestro caso, el formulario que rellenamos o los botones de generación de señales se encuentran en un componente y el resultado de las gráficas generadas se muestran en otro componente.

En Angular, como ya hemos mencionado anteriormente, existe una fuerte estructura muy divida en componentes. Para establecer la comunicación entre esos componentes, se debe hacer uso de la herencia de variables en Angular.

La herencia de variables en Angular no es más que la comunicación entre componentes relacionados estructuralmente o no. Es uno de los conceptos más necesarios en Angular, ya que debido a su definición, cuantos más componentes distribuidos e independientes mejor. Y por supuesto, esos componentes necesitarán comunicarse y lo podrán hacer de varias formas según el tipo de comunicación.

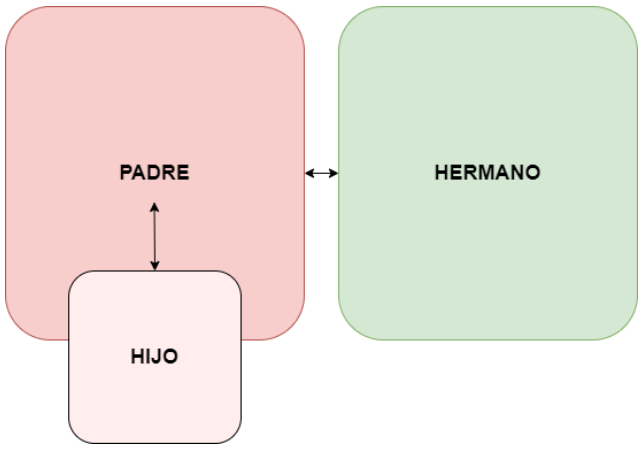

Figura 3-6. Estructura componentes

3.4.1 Comunicación de componente padre a hijo

Este es probablemente el método más común y directo para compartir datos. Funciona mediante el uso del decorador *@Input ()* para permitir que los datos pasen a través de la plantilla del padre al hijo.

Padre.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-parent',
  template: `
    <app-child [childMessage]="parentMessage"></app-child>
  `,
  styleUrls: ['./parent.component.css']
})
export class ParentComponent{
  parentMessage = "message from parent"
  constructor() { }
}
```

Figura 3-7. Caso1- padre

Hijo.component.ts

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-child',
  template: `
    Say {{ message }}
  `,
  styleUrls: ['./child.component.css']
})
export class ChildComponent {

  @Input() childMessage: string;

  constructor() { }

}
```

Figura 3-8. Caso1-hijo

3.4.2 Comunicación de componente hijo a padre

- Método 1: ViewChild

ViewChild permite que un componente se inyecte en otro, dando al padre acceso a sus atributos y funciones. Sin embargo, una advertencia es que el hijo no estará disponible hasta después de que se haya inicializado la vista. Esto significa que necesitamos implementar el enlace del ciclo de vida AfterViewInit para recibir los datos del hijo.

Padre.component.ts

```
import { Component, ViewChild, AfterViewInit } from '@angular/core';
import { ChildComponent } from "../child/child.component";

@Component({
  selector: 'app-parent',
  template: `
    Message: {{ message }}
    <app-child></app-child>
  `,
  styleUrls: ['./parent.component.css']
})
export class ParentComponent implements AfterViewInit {

  @ViewChild(ChildComponent) child;

  constructor() { }

  message:string;

  ngAfterViewInit() {
    this.message = this.child.message
  }
}
```

Figura 3-9. Caso2.1-padre

Hijo.component.ts

```
import { Component} from '@angular/core';

@Component({
  selector: 'app-child',
  template: `
  `,
  styleUrls: ['./child.component.css']
})
export class ChildComponent {

  message = 'Hola Mundo!';

  constructor() { }

}
```

Figura 3-10. Caso2.1-hijo

- Método 2: Output() y EventEmitter

Otra forma de compartir datos es emitir datos desde hijo, que el padre podrá listar o enumerar. Este enfoque es ideal cuando desea compartir los cambios de datos que ocurren en elementos como clicks de botones, entradas de formularios y otros eventos de usuarios.

En el padre, creamos una función para recibir el mensaje y establecerlo igual a la variable del mensaje.

En el elemento secundario, declaramos una variable messageEvent con el decorador de salida y la establecemos igual a un nuevo emisor de eventos. Luego creamos una función llamada sendMessage que emite llamadas en este evento con el mensaje que queremos enviar. Por último, creamos un botón para activar esta función.

El padre ahora puede suscribirse a este messageEvent generado por el componente hijo, luego ejecutar la función de recibir mensaje cada vez que ocurra este evento.

Padre.component.ts

```
    Message: {{message}}
    <app-child (messageEvent)="receiveMessage($event)"></app-child>
  `,
  styleUrls: ['./parent.component.css']
})
export class ParentComponent {

  constructor() { }

  message:string;

  receiveMessage($event) {
    this.message = $event
  }
}
```

Figura 3-11. Caso 2.2-padre

Hijo.component.ts

```
import { Component, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-child',
  template: `
    <button (click)="sendMessage()">Send Message</button>
  `,
  styleUrls: ['./child.component.css']
})
export class ChildComponent {

  message: string = "Hola Mundo!"

  @Output() messageEvent = new EventEmitter<string>();

  constructor() { }

  sendMessage() {
    this.messageEvent.emit(this.message)
  }
}
```

Figura 3-12. Caso 2.2-hijo

3.4.3 Comunicación entre componentes hermanos o no relacionados

Al pasar datos entre componentes que carecen de una conexión directa, como hermanos, nietos, etc., debemos utilizar un servicio compartido o global. Cuando se tienen datos que siempre deben estar sincronizados, el RxJS BehaviorSubject es muy útil en esta situación.

También se puede usar RxJS Subject para compartir datos a través del servicio, pero normalmente funciona mejor BehaviorSubject y por eso lo hemos elegido para generar el servicio global de nuestro proyecto Angular.

BehaviourSubject funciona de tal forma que siempre devolverá el valor actual de la suscripción; no es necesario llamar al siguiente. Tiene una función getValue () para extraer el último valor como datos sin procesar.

Además, asegura que el componente siempre reciba los datos más recientes.

En el servicio, creamos un BehaviorSubject privado que contendrá el valor actual del mensaje. Definimos una variable currentMessage para manejar este flujo de datos como un observable que será utilizado por los componentes. Por último, creamos una función que llama a continuación en BehaviorSubject para cambiar su valor.

Los componentes padre, hijo y hermano reciben el mismo tratamiento. Inyectamos el DataService en el constructor, luego nos suscribimos al observable currentMessage y establecemos su valor igual a la variable del mensaje.

Ahora si creamos una función en cualquiera de estos componentes que cambia el valor del mensaje, cuando esta función se ejecuta, los nuevos datos se transmiten automáticamente a todos los demás componentes suscritos.

Padre.component.ts

```
import { Component, OnInit } from '@angular/core';
import { DataService } from "../data.service";

@Component({
  selector: 'app-parent',
  template: `
    {{message}}
  `,
  styleUrls: ['./sibling.component.css']
})
export class ParentComponent implements OnInit {

  message:string;

  constructor(private data: DataService) { }

  ngOnInit() {
    this.data.currentMessage.subscribe(message => this.message = message)
  }

}
```

Figura 3-13. Caso3-padre

Data.service.ts

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

@Injectable()
export class DataService {

  private messageSource = new BehaviorSubject('default message');
  currentMessage = this.messageSource.asObservable();

  constructor() { }

  changeMessage(message: string) {
    this.messageSource.next(message)
  }

}
```

Figura 3-14. Caso3-servicio

Hermano.component.ts

```
import { Component, OnInit } from '@angular/core';
import { DataService } from "../data.service";

@Component({
  selector: 'app-sibling',
  template: `
    {{message}}
    <button (click)="newMessage()">New Message</button>
  `,
  styleUrls: ['./sibling.component.css']
})
export class SiblingComponent implements OnInit {

  message:string;

  constructor(private data: DataService) { }

  ngOnInit() {
    this.data.currentMessage.subscribe(message => this.message = message)
  }

  newMessage() {
    this.data.changeMessage("Hello from Sibling")
  }

}
```

Figura 3-15. Caso3-hermano

- Librería para gráficas

La librería elegida es ChartJS. Se trata de una librería que provee gráficos JavaScript simples pero flexibles para diseñadores y desarrolladores, Se ha elegido esta librería por su capacidad para mostrar todo tipo de gráficos: tipo línea, área, barras, radar, burbuja, dispersión, mezcla… He aquí algunos ejemplos:

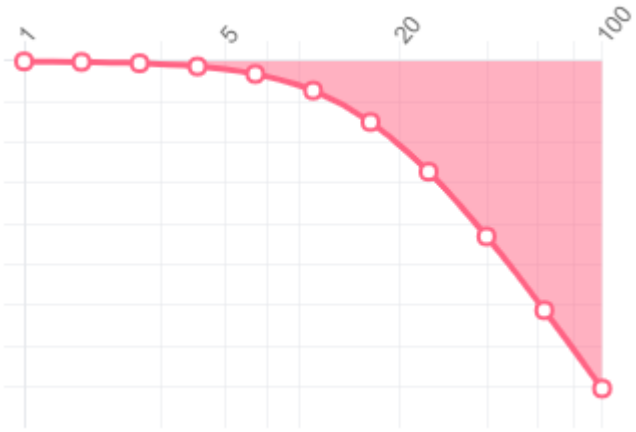

Figura 3-16. Gráficos Chartjs: tipo 1

Figura 3-17. Gráficos Chartjs: tipo 2

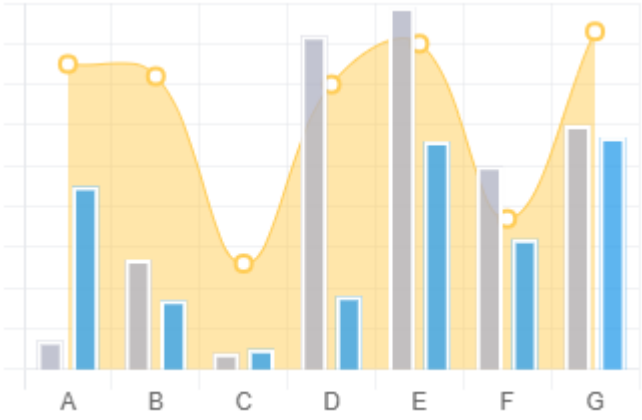

Figura 3-18. Gráficos Chartjs: tipo 3

3.5 Backend

Una vez entendida la estructura front-end, vamos a explicar brevemente la del back-end. La estructura backend de este proyecto no está incluida en el alcance de este TFG aunque, en cierta forma lo completa y complementa. Por tanto, me parece interesante hacer un breve resumen de lo que aquí se ha estructurado y analizado.

Consiste en un módulo Python conectado a una base de datos SQlite a la que hacemos consultas a través del framework Django. Además, se han usado programas como PyCharm, Spyder y keysight para ayudarnos con la constante comunicación con el osciloscopio y la prueba de distintos comandos.

Dentro de la estructura que engloba Django, se ha implementado la lógica de los comandos visa, es decir, se ha usado PyVisa, un paquete de Python que le permite controlar todo tipo de dispositivos de medición independientemente de la interfaz (por ejemplo, GPIB, RS232, USB, Ethernet). Esto nos permite leer sobre nuestros instrumentos electrónicos a través del protocolo Ethernet.

Toda esta estructura, a su vez, interactuará con un osciloscopio, fuente y generador de señal (que proporciona una señal cuadrada al inicio.

Cualquier llamada desde frontend a backend para solicitar datos, gráficas o información, leerá el osciloscopio gracias a los comandos de keysight. De la misma forma, cualquier modificación que el usuario solicite desde frontend, se hará sobre la señal en backend, devolviendo siempre una respuesta tipo JSON con dos arrays pertenecientes a los ejes X e Y.

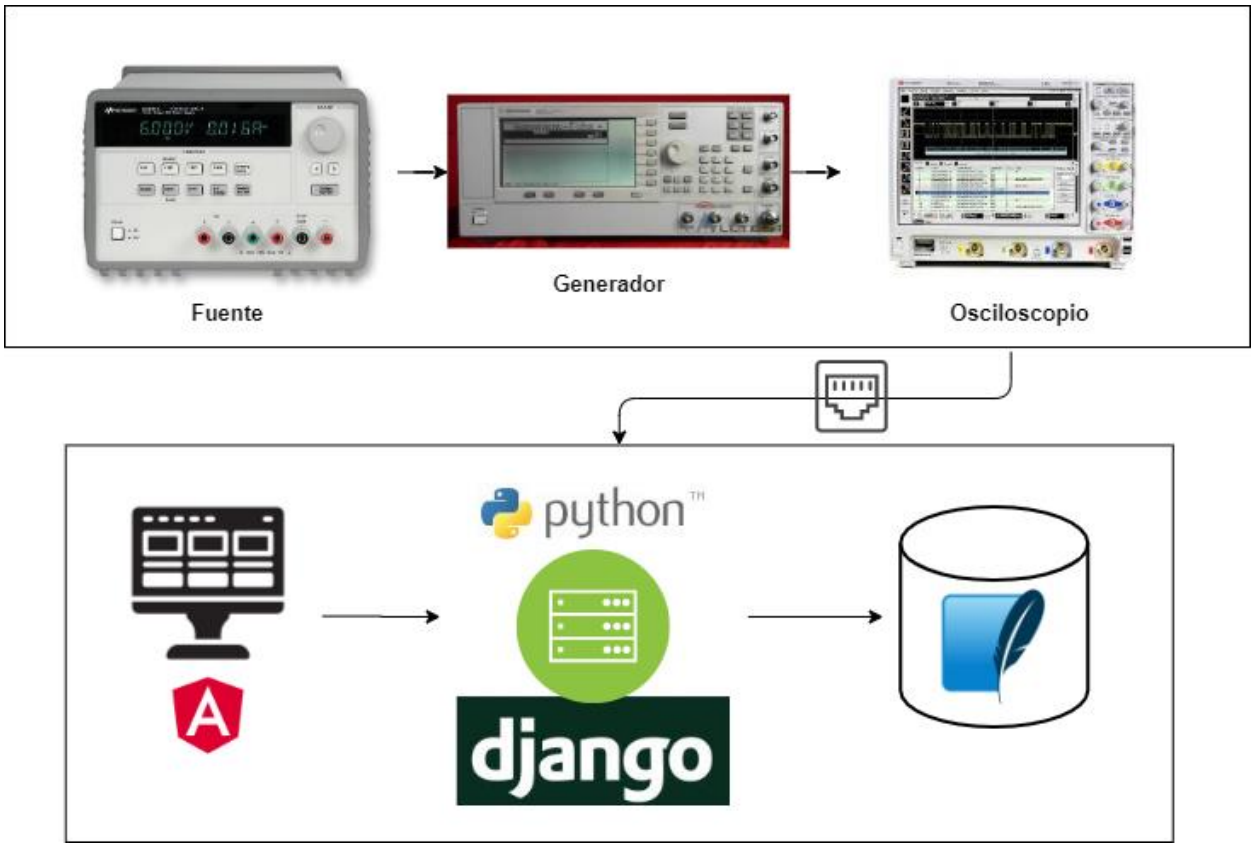

Figura 3-19. Backend

3.6 Instrumentación y programas

En este apartado vamos a detallar el conjunto de herramientas y programas usados en el desarrollo del frontend. Por otro lado, vamos a explicar de forma extensa cada instrumento que usemos, en este caso un osciloscopio.

3.6.1 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows , Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios podemos cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto.

Fue nuestra elección para trabajar con Angular desde el principio, ya que ofrece miles de plugins para: correcciones a la hora de no cumplir buenas prácticas, errores de sintaxis en javascript, espaciados, atajos propios de eclipse (son los más usados) y múltiples opciones que facilitan bastante el trabajo con Angular.

Otra de sus ventajas es que es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un lenguaje dado. Muchas de las características de Visual Studio Code no están expuestas a través de los menús o la interfaz de usuario. Más bien, se accede a través de la paleta de comandos o a través de archivos .json (por ejemplo, preferencias del usuario).

La paleta de comandos es una interfaz de línea de comandos. Sin embargo, desaparece si el usuario hace click fuera de él o presiona una combinación de teclas en el teclado para interactuar con algo que está fuera de él. Esto también se aplica a los comandos que requieren mucho tiempo. Cuando esto sucede, el comando en progreso se cancela. Cuando queremos que los cambios se refresquen y los errores nos aparezcan en consola sin necesidad de estar arrancando cada vez, usamos el comando *ng build –watch*.

Por otro lado, para arrancar el servidor en el host y puerto establecidos, usaremos en comando *ng serve*.

Figura 3-20. Visual Studio Code

Todas estas ventajas junto a la interfaz tan agradecida y usable que posee, ha hecho que sea nuestra mejor elección como editor para trabajar con el desarrollo frontend de este proyecto.

3.6.2 Spyder

Scientific Python Development Environment

Figura 3-21. Spyder

Consiste en un entorno de desarrollo interactivo para el lenguaje Python. Una de sus mayores ventajas es que dispone de numerosas funciones avanzadas de edición, pruebas, depuración e introspección. Todo esto es posible debido al uso de librerías de Python como NumPy o SciPy.

Algunas de las características principales de spyder son:

- Podremos disponer de un explorador de archivos y acceso al registro del historial.
- Su código fuente puede consultarse en GitHub, es decir, está al alcance de todos.
- El editor que integra este es multilenguaje.
- Vamos a poder explorar las variables creadas durante la ejecución de un archivo, al igual que en eclipse.
- Ofrece soporte de expresiones regulares.
- Dispone de una consola interactiva.

Instalación: sudo pip install spyder

Desinstalación: sudo pip install spyder

3.6.3 Keysight

El osciloscopio elegido para esta estructura es el *MSO9104A Agilent Oscilloscope*. En el formulario mostrado en frontend cada uno de los campos se corresponden con parámetros carácterísticos del osciloscopio, ya que es usado para configurar la señal del mismo.

Aunque forme parte directa de la comunicación backend, desde frontend se ha colaborado en la búsqueda de comandos parala interacción con el osciloscopio:

:STATus? CHANNEL1

Muestra si el canal especificado está encendido o apagado. Un valor de retorno de 1 significa encendido y un valor de retorno de 0 significa apagado.

<frequency1>, =: MEASure: FREQuency?

Instala una medida de la frecuencia de un pico FFT. Los picos están numerados de baja a alta frecuencia. Solo picos arriba del nivel de umbral especificado están numerados.

Para que este comando / consulta funcione, la fuente debe ser una función configurada en FFT o una memoria de forma de onda que contiene una FFT.

<originMinValue>, =:MEASure: VMIN?

Mide el voltaje mínimo absoluto presente en la forma de onda fuente seleccionada. Las fuentes se especifican con: :MEASure:SOURce o con el parámetro opcional que sigue al comando.

<offset>, =:MEASure:VMIDdle?

Mide el nivel de voltaje en el medio umbral de la forma de onda. Las fuentes se especifican con: MEASure: SOURce o con el parámetro opcional que sigue a: MEASure: VMIDdle.

<originMaxValue>, = :MEASure:VMAX?

Mide el voltaje máximo absoluto presente en la forma de onda fuente seleccionada. Las fuentes se especifican con el comando :MEASure:SOURce e o con el parámetro opcional siguiendo el comando MEASure:VMAX?

<originVPP>, = :MEASure: VPP?

Mide los voltajes máximos y mínimos en la fuente seleccionada, luego calcula el voltaje pico a pico como la diferencia entre los dos voltajes. Las fuentes se especifican con :MEASure:SOURce o con el parámetro opcional que sigue a MEASure: VPP?

<period>, = :MEASure:PERiod?

Mide el período del primer ciclo completo en la pantalla utilizando los niveles de umbral medio de la forma de onda (niveles del 50% con medidas estándar seleccionadas).

La fuente se especifica con el comando: MEASure: SOURce o con el comando opcional parámetro que sigue al comando: MEASure: PERiod.

El algoritmo es:

Si el primer borde de la pantalla está subiendo, luego periodo = segundo tiempo de flanco ascendente - primer tiempo de flanco ascendente más período = segundo tiempo de borde descendente - primer tiempo de borde descendente

:CHANnel<CHANnel<N>>:DISPlay 1

El comando: CHANnel <N>: DISPlay activa la visualización del canal especificado ON u OFF.

Se pueden especificar canales diferenciales y / o comunes usando la siguiente convención.

Si tenemos habilitados los canales en modo diferencial o común (utilizando : CHANnel <N>: DIFFerential o

: CHANnel <N>: comandos COMMonmode) y luego:

•: CHANnel1 se referiría al Canal 1 - Canal diferencial del Canal 3

•: CHANnel2 se referiría al Canal 2 - Canal diferencial del Canal 4

•: CHANnel3 se referiría al canal de modo común del Canal 1 + Canal 3

•: CHANnel4 se referiría al canal de modo común del Canal 2 + Canal 4

3.7 Compatibilidad en navegadores

Uno de los quebraderos de cabeza más común cuando tienes que desarrollar una aplicación frontend en la que intervienen estilos (CSS, SASS…), plugins, animaciones, etiquetas html, librerías como boostrap, etc es la compatibilidad de tu página web en los diversos navegadores existentes.

El soporte de navegadores de Angular, depende de la versión que utilices:

Tabla 3-1. Compatibilidad versiones navegadores

Navegador	Versión soportada
Chrome	Última versión
Firefox	Última versión y última versión de soporte extendido (ESR)
Edge	2 últimas versiones más recientes
Internet Explorer	11, 10*, 9* ("vista compatible" modo no soportado) *desaconsejado en v10
Safari	2 últimas versiones más recientes
IE Mobile	11 *desaconsejado en v10
iOS	2 últimas versiones más recientes
Android	X (10.0), Pie (9.0), Oreo (8.0), Nougat (7.0)

4 RESULTADOS

Una meta sin un plan, es solo un deseo.
Antoine de Saint-Exupéry

En apartados previos, se ha detallado el análisis de la situación inicial a la que nos enfrentábamos, las diferentes herramientas e instrumentos usados para solucionarla y la solución técnica que hemos decido aplicar. Tras este proceso, se han obtenido una serie de resultados técnicos y conclusiones teóricas y analíticas que nos permiten describir este apartado del documento.

4.1 Resultados visuales

Visualmente el resultado consistirá en una web con un menú, en la que se podrán realizar dos configuraciones distintas según la pestaña del menú seleccionada.

4.1.1 Contenido de la web

Dentro de la web, podremos ver un menú que diferencia dos opciones de configuración:

Pestaña 1: un formulario web junto a varios botones de configuración explicados anteriormente. Con este formulario se podrán generar señales cuadradas de distintos valores, FFT y PSD.

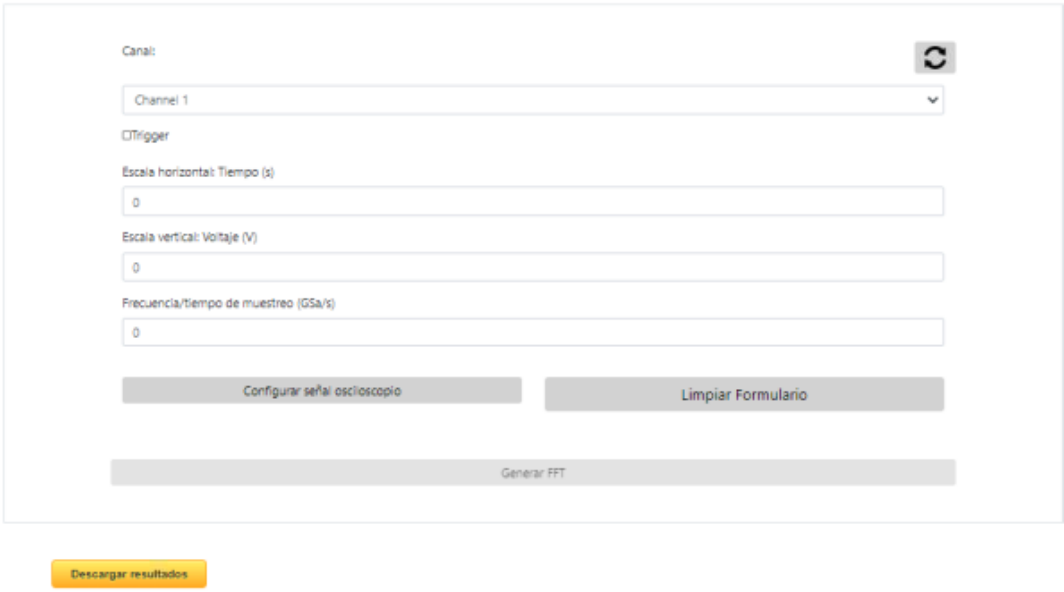

Figura 4-1. Pestaña 1

Pestaña 2: una lista de selectores en los que el usuario marcará los parámetros cuyo valor desee obtener. Estos parámetros se calcularán en backend sobre la señal generada y se mostrarán a la derecha, en "valores calculados".

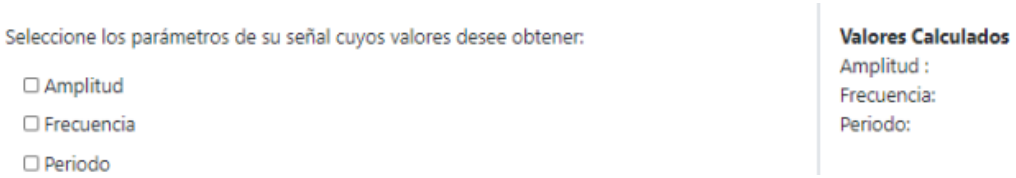

Figura 4-2. Pestaña 2

Por último, el resultado final de la web sería el mostrado en la Fig. 4-3., donde la cabecera y pie pueden personalizarse y se muestra un menú con dos pestañas: una para configurar señales y generar la FFT y otra para calcular parámetros de la señal del osciloscopio.

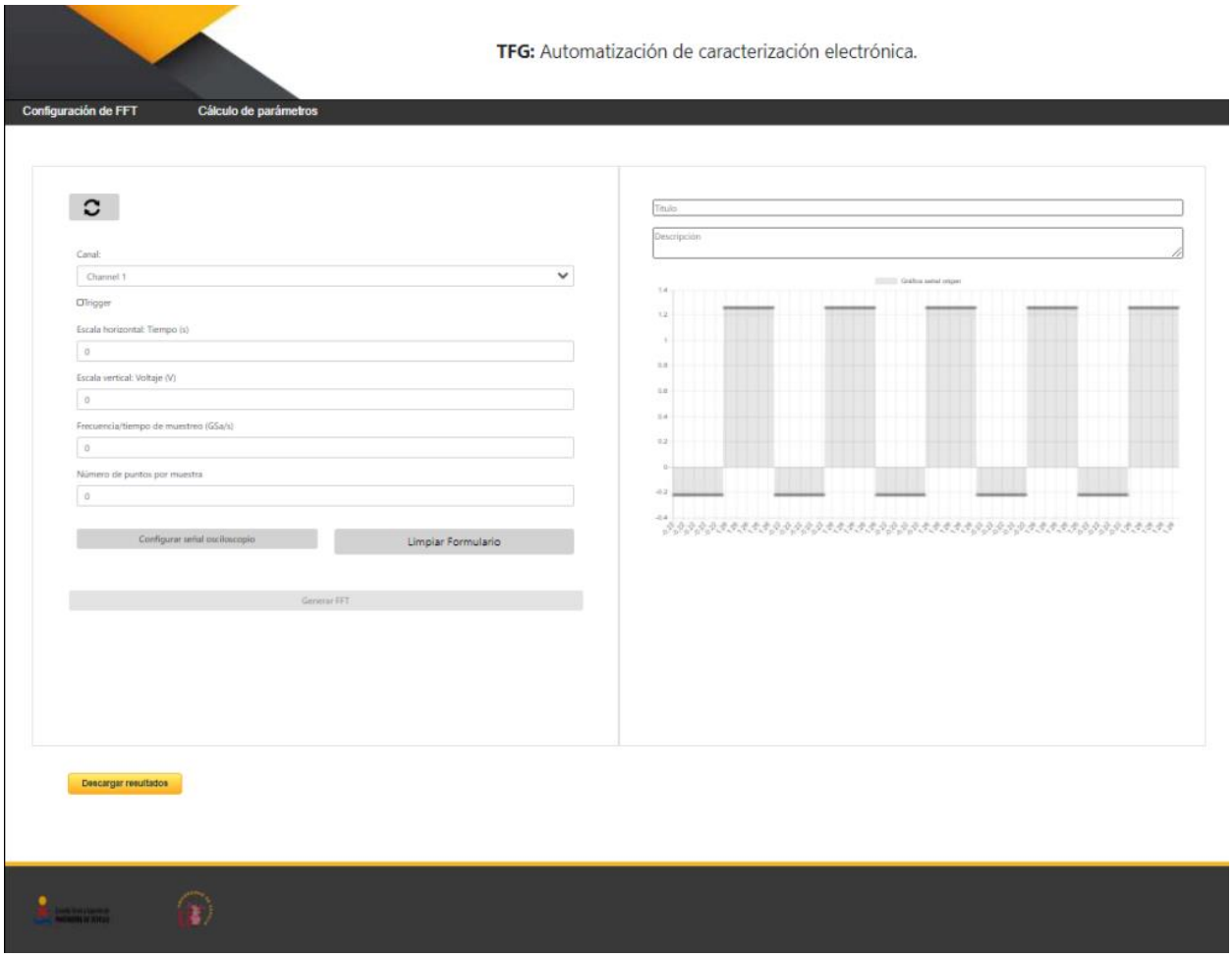

Figura 4-3. Web

4.1.2 Gráficas a modo de resultado

Como resultado, obtendremos varios tipos de gráficas.

Al entrar en la web, se generará y mostrará una señal cuadrada representando a la señal origen del osciloscopio.

Dicha señal se mostraría en frontend con el siguiente formato, incluyendo las áreas de texto para escribir y ejes de valor decimal cuya escala es configurada según los requerimientos de backend.

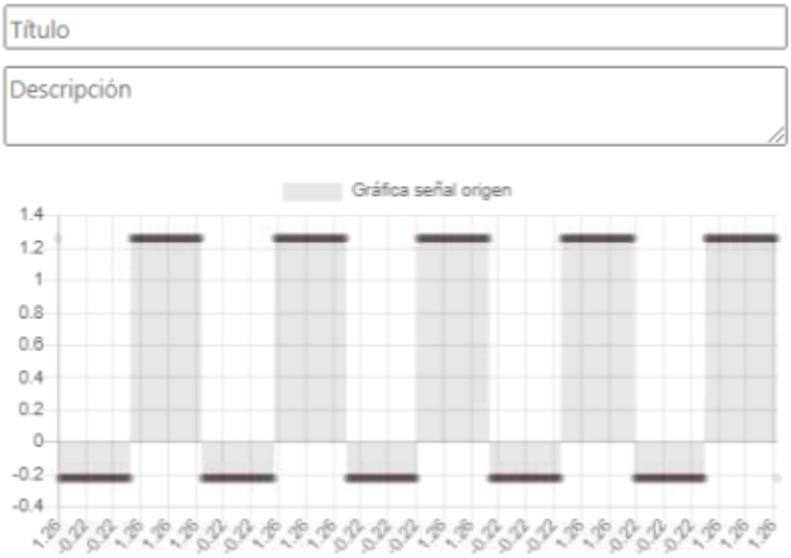

Figura 4-4. Resultado señal origen

Una vez obtenida la señal origen, podrás configurar en el formulario voltaje, tiempo, frecuencia y otros parámetros para obtener una señal diferente a la señal origen y configurada por ti mismo:

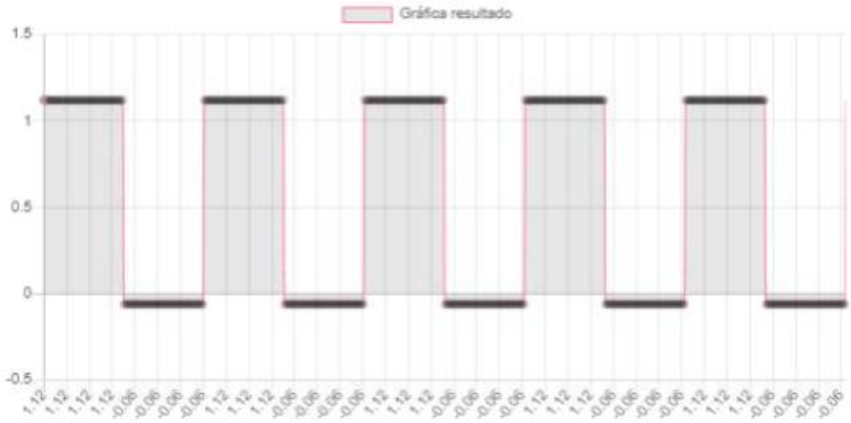

Figura 4-5. Señal generada tras configuración con formulario

Después de configurar la señal al gusto del usuario, podrías indicar el número de muestras y generar la FFT pulsando el botón “Generar FFT”.

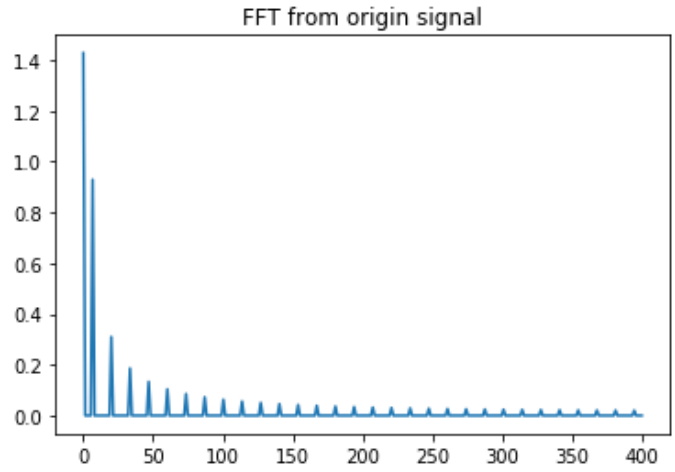

Figura 4-6. FFT

El resultado en la web comprende dos opciones, la FFT o la PSD (potencia):

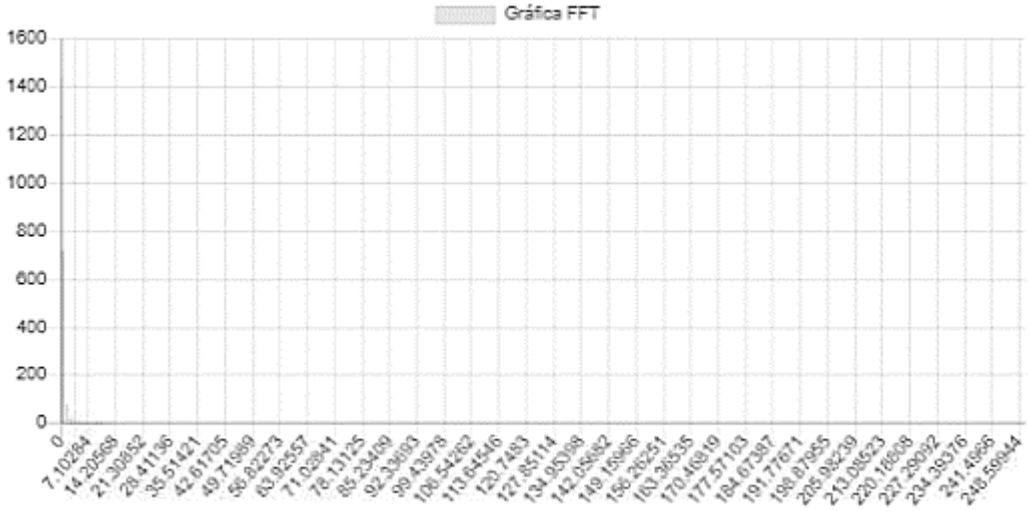

Figura 4-7. Psd

4.1.3 Funciones matemáticas implementadas

Además de la función Math.round() de angular usada para redondear valores del eje x de las gráficas, se ha implementado el cálculo de la Transformada Rápida de Fourier.

4.2 Resultado global

En resumen y de forma global, el resultado final consiste en un entorno web dividido en front-end basado en Angular, backend basado en phyhon y un osciloscopio remoto que nos ofrece la señal para trabajar con ella desde el escritorio remoto.

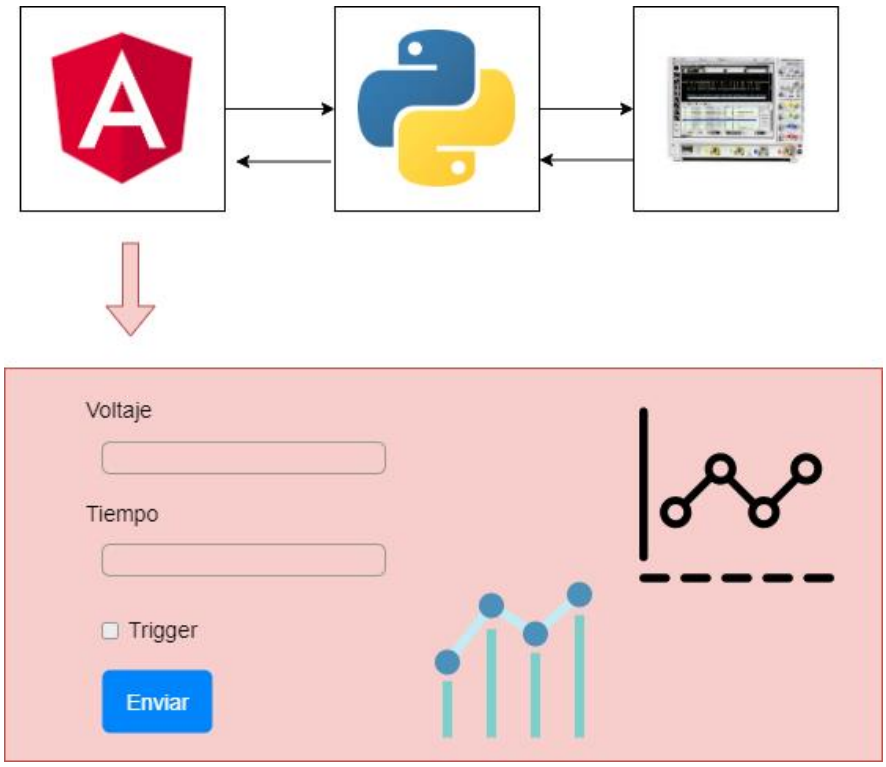

Figura 4-8. Resultado global

5 CONCLUSIONES

Después de saber cuándo debemos aprovechar una oportunidad, lo más importante es saber cuándo debemos renunciar a una ventaja.

Benjamin Disraeli, 1804

En este apartado vamos a sacar aquellas conclusiones y aprendizajes obtenidos durante el desarrollo de este proyecto.

En un primer momento, nos propusimos automatizar de la forma más dinámica posible la configuración de un circuito sigma-delta generado por nosotros mismos en el laboratorio. Debido a circunstancias actuales, no pudimos finalizar ese desarrollo de laboratorio, por lo que hicimos un cambio de especificaciones.

En lugar de una placa, se consumiría la propia señal de un osciloscopio conectado en remoto a un ordenador 24h, a modo de servidor. Dicha señal se visualizaría como se muestra en la imagen inferior.

Figura 5-1. Osciloscopio

¿Cuál era el objetivo entonces? Programar una web lo más sencilla posible y apta para cualquier usuario, de forma que éste pudiese configurar la propia señal que se recibe en el osciloscopio, calcular sus parámetros,

generar su FFT, etc.

Para ello se hizo un análisis de alguno de los frameworks de frontend más utilizados actualmente, decidiendo finalmente que el más adecuado era Angular. En cuanto a backend, se analizó que uno de los lenguajes más usados en el mundo de la instrumentación y que ofrecía la posibilidad de usar comandos pyvisa y demás era Python. Además, se eligió una de las bases de datos más usadas en backends programados con Python: SQLite.

Una vez elegida la estructura, se analizaron las mejores librerías para representar gráficos en Angular y el tipo de estructura que queríamos tener en la web. Además, se decidieron las distintas funcionalidades que íbamos a aportar al usuario.

Una vez hecho el análisis conjunto, se comenzó a programar ayudándonos de herramientas para hacer más fácil ese trabajo en equipo (swagger), creando finalmente una web que permitiera dos configuraciones principales:

Generar parámetros a partir de una señal, visualizar la señal del osciloscopio, cambiar sus valores generar su FFT.

Además, se pensó que al estar orientado principalmente a profesores y alumnos de electrónica, podía ser muy útil ofrecer la opción de guardar e imprimir tus gráficas junto a bloques de textos a modo de memoria o documento informativo.

Por último, destacar que una de las grandes ventajas de esta plataforma es su facilidad de uso. No es necesario que un administrador o gran conocedor de la aplicación sea el que la gestione. Cualquier persona capaz de completar un sencillo formulario, podrá hacer uso de esta aplicación.

Aunque nosotros hayamos llegado hasta este punto, vemos un gran margen de ampliación de nuestro trabajo, creyendo así que implementar otras configuraciones, programar la aplicación para móvil, configurar el generador de señal además del osciloscopio y usar circuitos reales serían grandes líneas futuras de trabajo.

5.1 Mejoras de futuro

Es ente apartado, vamos a tratar algunas de las aplicaciones que podría tener nuestro proyecto en el futuro, qué mejoras se podrían realizar y qué desarrollos se podrían incluir para ampliarlo.

5.1.1 Aplicaciones de futuro

A nuestro alrededor, existen más plataformas de automatización con objetivo de caracterizar o manejar dispositivos electrónicos de lo que en realidad pensamos.

Por ejemplo, está muy me moda actualmente usar aparatos de limpieza robóticos controlados mediante aplicaciones móviles. Esto no deja de ser un aparato conformado por miles de componentes electrónicos que caracterizas y controlas desde una interfaz web o móvil, es decir, una plataforma de automatización de medidas por sensor para control de dispositivos.

Figura 5-2. Automatización de robots mediante aplicaciones

Otra aplicación muy útil ya existente es TRIGGERcmd, una plataforma de automatización que permite controlar al 100% un dispositivo electrónico, como un móvil u ordenador mediante una serie de órdenes a un *echo dot*. Este echo dot, de manera interna lanza cualquier script o comando y realiza la acción deseada: relizar una medida de un dispositivo remoto que tengamos en nuestro PC, hacer una configuración, un cálculo, etc.

Es decir, TRIGGERcmd es un servicio en la nube que le permite ejecutar comandos de forma segura y remota en nuestros dispositivos.

Dichos comandos pueden instalar actualizaciones, abrir un garaje, ejecutar un script o cualquier otra acción. Para activarlos, solo necesitas Alexa, Google Home o cualquier otro dispositivo inteligente de identificación de voz.

Figura 5-3. Triggercmd

Estos proyectos, junto a otros muchos existentes en la actualidad, tienen una gran relación con nuestro Trabajo Fin de Grado desarrollado.

Además de lo ya existente, me gustaría hacer un breve repaso sobre qué aplicaciones funcionales puede tener nuestro TFG si expandimos el rango, es decir, si pensamos a lo grande.

Hoy en día existen numerosos experimentos que se encuentran en entornos donde no puede existir intervención humana o que necesitan supervisiones tan extensas en el tiempo que la única opción es hacerlas online.

Un claro ejemplo de este tipo de dispositivos son los aceleradores de partículas o cualquier dispositivo electrónico de radiación.

Figura 5-4. Acelerador de partículas

Estos dispositivos utilizan campos electromagnéticos para acelerar partículas cargadas a altas velocidades y hacerlas colisionar con otras. Esto permite generar más partículas que son estudiadas ya que ofrecen gran cantidad de información sobre las partículas que desaparecen en ese proceso.

Los aceleradores de partículas se parecen, en parte, a los rayos sobre la atmósfera terrestre, lo cual produce una lluvia de partículas inestables. Sin embargo, los aceleradores consiguen un entorno mucho más controlado para estudiar estas partículas generadas, y su proceso de desintegración.

Ese estudio de partículas, tanto inestables como estables, tiene un gran futuro en ámbitos de

medicina, exploración espacial, tecnología electrónica, etc.

Sin embargo, son muchos los peligros que, se rumorea, pueden concebir los aceleradores de partículas más potentes. Desde agujeros negros hasta destrucción de la Tierra.

No solo por su peligro, sino también por la delicadeza que requiere, es necesario un control remoto y vigilancia de 24h a este tipo de dispositivos. Una de las formas más cómodas y sencillas de hacerlo es mediante plataformas de automatización de pruebas, test, capturas fotográficas, temperatura, etc

Es decir, cualquier tipo de variable o sensor que funcione con una plataforma que permita su configuración remota desde una web, sin necesidar de estar presente en la misma sala que el dispositivo.

Por tanto, una de las aplicaciones más fuertes de una plataforma web para caracterización y configuración de dispositivos electrónicos podría ser este mismo ejemplo citado.

5.1.2 Mejoras técnicas

Alguna de las actuaciones que se pueden desarrollar en el futuro para mejorar la plataforma son:

- Generar aplicación Android y iOs además de la web: hoy en día son miles los usuarios que prefieren usar su teléfono móvil para cualquier tipo de gestión, ya que les permite no tener que llevar encima un ordenador.

  Aunque nuestra web está pensada para que alumnos y profesores que trabajan en un laboratorio enciendan su ordenador y realicen configuraciones sobre señales necesarias para sus distintos proyectos, puede ser una buena idea ampliar este proyecto con una versión móvil de dicha web (tanto Android como iOs).

  Angular funciona bastante bien para versiones responsive, por tanto, podría usarse para móvil. Pero si se quisiera implementar una aplicación móvil como tal, recomiendo el uso de *Ionic*.

- Crear un sistema de Administración de la web para que ciertos usuarios con el rol ADMIN puedan modificar la web a su antojo. Por ejemplo, incluir más pestañas en el menú, duplicar y crear formularios, borrar o incluir inputs dentro del formulario, modificar el pie y la cabecera de la web, etc

- Aumentar las pruebas y funcionalidades o configuraciones ofrecidas: sería cuestión de aumentar las pestañas del menú y generar un componente nuevo por cada funcionalidad ofrecida. Siempre se podrán duplicar y reutilizar los componentes ya existentes como las gráficas o formularios.

- Realizar la configuración del generador de señal además del osciloscopio.

- Implementar un circuito real con Altium (era nuestra idea inicial), imprimirlo y realizar estas configuraciones sobre dicho circuito real.

Esperamos y deseamos que nuestros futuros compañeros se interesen por este trabajo y quieran ampliarlo con otros Proyectos de Fin de Grado.

REFERENCIAS

[1] « Convertidores A/D de sobremuestreo usando técnicas de modulación Sigma-Delta» *Artículo,* cap. 1.

[2] KeySight Technologies, *Programmer's Guide*.

[3] Imagen Acelerador de Partículas , espaciociencia.com

[4] José Manuel Nieves, «Podría ser que se formara un agujero negro ». Periódico ABC ciencia, 2019.

WEBGRAFÍA

https://www.toptal.com/javascript/como-elegir-el-mejor-framework-de-front-end

https://existek.com/blog/top-front-end-frameworks-2020/

https://medium.com/ngesyfirebase/lo-nuevo-que-se-viene-en-angular-8-y-9-cc8571023c13

https://enmilocalfunciona.io/mejoras-y-cambios-con-angular-8/

https://www.imaginaformacion.com/tutorial/angular-8-novedades/

https://medium.com/ngesyfirebase/lo-nuevo-que-se-viene-en-angular-8-y-9-cc8571023c13

https://www.campusmvp.es/recursos/post/las-5-principales-ventajas-de-usar-angular-para-crear-aplicaciones-web.aspx

https://www.campusmvp.es/recursos/post/por-que-los-desarrolladores-y-las-empresas-eligen-angular.aspx

https://pyvisa.readthedocs.io/en/latest/

https://www.sqlite.org/index.html

https://app.swaggerhub.com/

https://visualstudio.microsoft.com/es/

https://angular.io/guide/ngmodules

https://getbootstrap.com/

https://www.monografias.com/trabajos/sigmadelta/sigmadelta.shtml

https://www.hiberus.com/crecemos-contigo/diarios-de-un-junior-css-vs-sass/

https://angular.io/api/common/http/HttpClientModule

https://angular.io/guide/reactive-forms

https://angular.io/api/platform-browser/BrowserModule

https://fireship.io/lessons/sharing-data-between-angular-components-four-methods/

https://www.chartjs.org/docs/latest/charts/

https://es.wikipedia.org/wiki/Transformada_r%C3%A1pida_de_Fourier

https://fireship.io/lessons/sharing-data-between-angular-components-four-methods/

https://www.digitalocean.com/community/tutorials/angular-component-inheritance

https://espaciociencia.com/acelerador-de-particulas/

https://docs.djangoproject.com/en/3.0/

https://app.diagrams.net/

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

https://www.triggercmd.com/es/

https://es.wikipedia.org/wiki/Acelerador_de_part%C3%ADculas