Design of an online Web compiler

Raül Loscertales Vilar

Resumen—El objetivo de este trabajo es explorar el apasionante mundo del desarrollo web, con el propósito de crear un compilador web en línea. La investigación se centra en el diseño e implementación de un servicio web que permita a los usuarios escribir, compilar y ejecutar código en diversos lenguajes de programación, evitando así la necesidad de instalar múltiples aplicaciones, extensiones, compiladores, máquinas virtuales para cada lenguaje. Para lograr esto, se desarrollará una interfaz intuitiva y personalizable utilizando PHP para el backend y HTML, CSS y JavaScript para el frontend. Además, se utilizarán herramientas como Docker para asegurar la ejecución segura del código y Express.js para gestionar las solicitudes HTTP en el servidor. Durante el desarrollo del proyecto, se irán añadiendo funcionalidades adicionales para ofrecer una herramienta más completa y útil para los usuarios. Además, este proyecto tiene el potencial de ser una herramienta valiosa para la educación y la práctica de la programación en un entorno accesible y controlado.

Palabras clave—Desarrollo Web, Compilador, Frontend, Backend, PHP, JavaScript, CSS, HTML, Servidor, Docker, ACE, Programación, MVC, SQL

Abstract— The aim of this work is to explore the exciting world of web development, with the purpose of creating an online web compiler. The research focuses on the design and implementation of a web service that allows users to write, compile and execute code in different programming languages, thus avoiding the need to install multiple applications, extensions, compilers, virtual machines for each language. To achieve this, an intuitive and customisable interface will be developed using PHP for the backend and HTML, CSS and JavaScript for the frontend. In addition, tools such as Docker will be used to ensure secure code execution and Express.js to handle HTTP requests on the server. During the development of the project, additional functionalities will be added to offer a more complete and useful tool for users. Furthermore, this project has the potential to be a valuable tool for the education and practice of programming in an accessible and controlled environment.

Index Terms— Web Development, Compiler, Frontend, Backend, PHP, JavaScript, CSS, HTML, Server, Docker, ACE, Programming, MVC, SQL

1 Introducción

En la actualidad, tanto estudiantes como profesionales recurren a diversas aplicaciones para llevar a cabo sus proyectos de programación. Sin embargo, la selección del software adecuado puede resultar compleja o tediosa. Los programadores menos experimentados pueden sentirse abrumados durante el proceso de instalación de estas aplicaciones, ya que es crucial elegir aquellas que se ajusten a sus necesidades específicas y al lenguaje de programación que deseen utilizar. Además, al instalar estas aplicaciones, es común encontrarse con una variedad de paquetes y funciones adicionales que deben ser seleccionadas antes de finalizar la instalación, lo que puede añadir una capa adicional de dificultad para los usuarios menos familiarizados con el proceso.

Un caso específico se presenta al utilizar, por ejemplo, VSCode donde los usuarios se enfrentan a la tarea de instalar no solo el compilador del lenguaje deseado, sino también los depuradores necesarios, junto con varias extensiones creadas por diferentes desarrolladores. Además, es imprescindible configurar correctamente los directorios en los que se localizaran las diferentes herramientas puesto que si estas no se almacenan en una ubicación conocida por VSCode o si no se direccionan correctamente desde el programa pueden surgir errores. Por ello nace la idea de este trabajo en el cual se desarrollará un servicio en línea capaz de funcionar directamente sin la necesidad de instalar ninguna aplicación independiente, ningún compilador y sin tener en cuenta ninguna dependencia. La página web tendrá su propio editor de código donde el usuario podrá programar en diferentes lenguajes y cuando este quiera podrá ejecutar su programa en la propia web donde podrá observar los resultados de este. Además, al tratarse de un servicio web esto significa que puede ser utilizado por cualquier dispositivo, siempre y cuando este tenga acceso a Internet.

Finalmente, es importante destacar que esta página podría incluso ser utilizada a nivel docente para impartir clases, permitiendo que los alumnos se familiaricen con el lenguaje que se esté estudiando desde un entorno seguro y preparado para ellos.

2 Objetivos

El objetivo principal de este trabajo es desarrollar una página web que sea capaz de compilar código, en diferentes lenguajes, mostrando los resultados de manera dinámica. Esta funcionalidad constituirá el núcleo esencial de la página web.

Una vez alcanzado el objetivo principal, se explorarán diversas mejoras y funcionalidades adicionales con el fin de ofrecer una experiencia más completa a los usuarios. El primero de estos objetivos secundarios sería la reorganización de la aplicación con el objetivo de obtener una interfaz más amigable para los usuarios. Otra de las características que se implementarían sería la capacidad de registrar usuarios,

lo que les permitirá mantener sesiones activas y almacenar sus archivos de manera que puedan acceder rápidamente a estos. Otro objetivo secundario sería la inclusión de características que permitan la personalización del editor, proporcionando diversas opciones para que cada usuario pueda ajustar la página según sus preferencias.

Finalmente, se llevará a cabo un análisis de las vulnerabilidades de la aplicación, abarcando tanto la propia web como el servidor de compilación y la base de datos, con el fin de garantizar la seguridad de la aplicación para los usuarios.

A continuación, se presentan de forma esquemática los objetivos de este trabajo. Los objetivos principales son aquellos que necesariamente deben cumplirse, mientras que los objetivos secundarios buscan complementar la página web. Están ordenados por prioridad, siguiendo el orden establecido aquí:

Objetivos principales:
- Diseño básico de la web.
- Programación backend y frontend.
  - Investigar y evaluar el uso de ACE como editor de código.
- Crear un servidor de prueba capaz de compilar código a través de peticiones.
- Combinar la funcionalidad del servidor con la página web para mostrar los resultados de ejecutar el código dinámicamente.

Objetivos secundarios:
- Añadir opciones de personalización al editor de código.
- Añadir la posibilidad de importar archivos.
- Añadir la opción de exportar archivos.
- Realizar un análisis sobre la necesidad de implementar un sistema de inicio de sesión para usuarios.
  - Configurar la base de datos SQL.
  - Desarrollar el backend del formulario de inicio de sesión.
  - Diseñar el frontend para el proceso de inicio de sesión.
- Ampliar la cantidad de lenguajes disponibles para compilar y ejecutar.
  - Realizar la programación necesaria en el servidor.
  - Desarrollar tanto el frontend como el backend para permitir la selección del lenguaje.
- Realizar un análisis de vulnerabilidades y posibles soluciones a implementar.
- Mejorar la interfaz de usuario.

3 METODOLOGÍA

El desarrollo de este proyecto se puede dividir en diferentes fases, durante la primera se plantearán los objetivos más relevantes que se podrán implementar y la metodología que se llevará a cabo en el proyecto, debe notarse que estos objetivos son estudiados a priori motivo por el cual cabe la posibilidad de que durante el desarrollo alguno de estos pueda no resultar necesario. Durante esta primera fase también se analizarán las herramientas que se van a utilizar para desarrollar la página web.

Una vez estudiado el recorrido del proyecto se entrará en la siguiente etapa, la etapa de desarrollo, la cual abarcará la mayor parte del proyecto. Para llevar a cabo esta fase se va a utilizar una metodología de desarrollo iterativo e incremental, esta permitirá llevar a cabo el desarrollo en iteraciones, donde en cada iteración se llevarán a cabo una serie de pasos [1] . Cada una de estas iteraciones se puede considerar como un pequeño proyecto el cual se debe llevar a cabo siguiendo los siguientes pasos: investigación, desarrollo, prueba y entrega. Donde todos estos procesos se llevan a cabo dentro de cada uno de los objetivos.

Gracias a esta metodología se pueden llevar a cabo los diferentes objetivos propuestos de manera que en cada iteración se completen y se testeen adecuadamente. Esto permite una mayor flexibilidad en cuanto a la aparición de nuevos objetivos, ya que cabe la posibilidad de que durante el desarrollo de la aplicación surjan nuevas ideas que puedan ser el foco de la siguiente iteración. Además, a partir de las iteraciones se obtienen partes de la funcionalidad final de la web con lo que después de cada una de ellas se estará obteniendo un resultado completo que formará parte del producto final que se desea obtener.

Finalmente, cuando se haya finalizado la etapa de desarrollo quedará la última etapa durante la cual se acabará de analizar y revisar el producto final, se redactará un informe documentando los resultados obtenidos y se cerrará el proyecto.

4 ESTADO DEL ARTE

Para llevar a cabo este proyecto se emplearán diferentes herramientas específicas para cada una de las funciones necesarias para completar la página web.

En primera instancia para llevar a cabo la programación del backend se utilizará PHP, este es uno de los lenguajes más populares con una extensa comunidad. PHP también facilita la integración con bases de datos SQL, proporcionando una gestión eficaz de datos, y su flexibilidad permite una interacción fluida con HTML, lo que simplifica la creación de aplicaciones web [2].

En el frontend, se emplearán HTML, CSS y JavaScript (JS), los cuales son considerados los pilares del desarrollo web. HTML se utiliza para estructurar el contenido de las páginas web, CSS define el estilo y la presentación visual, y JavaScript añade interactividad y dinamismo a las páginas. JavaScript, en particular, cuenta con una amplia gama de librerías y frameworks que facilitan y agilizan el desarrollo [3] [4] [5].

Para el almacenamiento de datos de los usuarios, se optará por una base de datos SQL. Esta elección no solo complementa bien la integración con PHP, sino que también ofrece alta eficiencia y flexibilidad en la gestión de datos asegurando una respuesta rápida.

Para llevar a cabo la programación, se empleará el patrón

modelo-vista-controlador (MVC). Este patrón se centra en la separación entre la lógica de la aplicación (el controlador) de la interfaz de usuario (la vista) y de la lógica de datos (el modelo). Esto facilita tanto el mantenimiento, como la escalabilidad del código, también facilita el desarrollo dinámico permitiendo la carga dinámica de contenido de manera simple sin la necesidad de recargar la página lo que mejora la experiencia del usuario [6].

En cuanto al editor de código que se utilizará, este será ACE (AJAX.org Cloud9 Editor), una herramienta poderosa y versátil [7]. ACE es un editor de código basado en navegador, escrito en JavaScript que iguala e incluso amplia las características y el rendimiento de otros editores independientes actuales como Eclipse.

Una vez que el usuario haya programado su código en el editor, será necesario compilarlo o interpretarlo según el lenguaje y mostrar los resultados en la propia web. Para ello, se utilizará un servidor externo donde estarán instalados los diferentes intérpretes. Este servidor compilará y ejecutará el código, devolviendo el resultado a la página web para su visualización. En este servidor se utilizará Express.js, ya que este framework de Node.js permite manejar las solicitudes HTTP de forma rápida y flexible, facilitando a su vez la generación de las respuestas necesarias [8].

Otra herramienta que se utilizará en el servidor de compilación es Docker. Este método de contenedorización resultará útil para manejar ciertas vulnerabilidades, ya que proporciona un entorno aislado y seguro para la ejecución de código. Docker permite crear contenedores que encapsulan las aplicaciones y sus dependencias, asegurando que se ejecuten de manera consistente en cualquier entorno [9].

Otra herramienta que merece mención, la cual se empleará durante la sección de análisis de vulnerabilidades es el ofuscador de código. Esta es una poderosa herramienta que permite transformar un cierto código en otro ilegible y muy difícil de interpretar. Esta herramienta es útil para ocultar o entorpecer el estudio del flujo de la web desde el propio navegador, incrementando así la seguridad de la aplicación.

La combinación de estas tecnologías permite desarrollar un producto que cumple con los estándares actuales del desarrollo web, asegurando funcionalidad, eficiencia, y una experiencia de usuario optimizada.

5 Planificación

Respecto a la planificación, esta se ha llevado a cabo siguiendo los objetivos mencionados previamente. El resultado es el diagrama de Gantt presente en el apéndice, Figura 1.

Este diagrama de Gantt ilustra la planificación de las distintas tareas necesarias para llevar a cabo este trabajo, abarcando desde los objetivos principales y secundarios hasta la fase de documentación. En gris están marcados los diferentes grupos de tareas. Los objetivos principales se llevarán a cabo desde el 11 de marzo hasta el 4 de abril, después se iniciará la segunda fase en la que se desarrollarán los objetivos secundarios donde se implementarán la mayor cantidad de mejoras que se pueda en el tiempo del que se dispone, hasta el 26 de mayo. Dentro de las tareas correspondientes a los objetivos secundarios, se ha asignado un período adicional de siete días marcado en verde, destinado a hacer frente a cualquier imprevisto que pueda surgir durante la ejecución del proyecto.

Finalmente se entrará en la fase de documentación durante la cual se preparará tanto el informe final como la presentación del trabajo, esta fase terminará el 27 de junio.

6 Desarrollo

A continuación, se explica cómo se ha desarrollado el proyecto, analizando si este se ha seguido acorde a las tareas planteadas originalmente y explicando el flujo que se ha seguido para llevarlas a cabo junto con las implicaciones de cada una de estas. Esta sección se divide en dos partes: la primera se centra en las tareas principales, mientras que la segunda aborda una serie de tareas que, aunque no son imprescindibles, agregan valor al proyecto.

6.1 Desarrollo de los objetivos principales

Para empezar este proyecto se plantearon una serie de objetivos principales necesarios para garantizar la funcionalidad básica de la página web.

6.1.1 Diseño de la Web y evaluación de ACE

Donde el primero de estos ha sido establecer un diseño básico inicial de la web, con el objetivo de utilizarse como página de prueba donde se ha explorado la viabilidad de utilizar ACE (Ajax.org Cloud9 Editor) como editor de código para la programación desde la web. Para ello primero se han obtenido las librerías y dependencias necesarias a partir de su repositorio oficial, una vez instalado todo, se ha podido evaluar que efectivamente este editor de código tiene mucho potencial como para utilizarse acorde a las necesidades que se puedan presentar en la web. Se presentan numerosas ventajas en distintos aspectos: como editor, ofrece diversas funcionalidades que facilitan la programación como puede ser la posibilidad de utilizar un autocompletado de código, también se marcan los errores de sintaxis que se producen entre otras muchas funciones, desde el punto de vista de personalización este también brinda múltiples opciones para adaptarse a las preferencias de cada usuario y finalmente, en cuanto a su integración en la web, se presenta como una herramienta muy efectiva ya que esta se basa en llamadas asíncronas, lo que posibilita una actualización dinámica de los contenidos del editor asegurando así que este se pueda utilizar de forma intuitiva sin tener que refrescar la web cada vez que se produce un cambio.

Tras comprender el funcionamiento de ACE, se ha llevado a cabo una reestructuración de la web, abarcando desde la disposición de los elementos hasta la selección de la paleta de colores. Para llevar a cabo esta reorganización se ha programado tanto el frontend como el backend. En otras palabras, se ha desarrollado tanto el estilo necesario como el

HTML que servirá como página de inicio de la web. Durante este proceso, se ha contemplado tanto la disposición como la estética de los distintos botones, definiendo las funciones que deberán desempeñar. Estas funcionalidades se implementarán en fases posteriores del proyecto. En la Figura 2 se muestra la página principal de la web.

Figura 2 - Visualización de la página principal de la web.

6.1.2 Servidor de compilación

Una vez desarrollado el esqueleto básico de la página web, que incluye el editor de código, el área de visualización del resultado, y una serie de botones, ahora se debe desarrollar el servidor encargado de compilar y ejecutar el código. El objetivo inicial ha sido desarrollar un servidor simple que sea capaz de compilar código, ejecutarlo y mandar el resultado a la página web para que esta se lo muestre al usuario. Para ello se ha programado el servidor desde un entorno Linux donde previamente se ha instalado el compilador de código GCC (GNU Compiler Collection), el cual permite la compilación de código C, más adelante se instalarán herramientas que permitan tratar otros lenguajes de programación. En cuanto al servidor, este está programado en JS donde se emplea el framework de Express utilizado para manejar solicitudes HTTP, al recibir una petición en función de la solicitud POST que se haga se utilizarán diferentes compiladores, actualmente únicamente existe la opción de utilizar GCC. Antes de compilar el código primero se copia el contenido de la petición, es decir el código, en un archivo temporal el cual luego se compila con GCC y se ejecuta el ejecutable resultante, finalmente el resultado de esta ejecución se envía como respuesta al cliente, esta respuesta puede ser un mensaje de error en caso de que algo falle durante la compilación o ejecución. Para manejar múltiples peticiones al servidor se utilizan funciones asíncronas y promesas de manera que el servidor no se quede bloqueado. Una vez programado se han realizado una serie de pruebas a partir de peticiones generadas en local para comprobar que el funcionamiento del servidor fuese correcto.

Una vez validado el funcionamiento del servidor se han unido las funcionalidades de la página web con las del servidor de manera que se ha programado uno de los botones de la web, el de “Ejecutar” con el cual se crea una petición HTTP en la que se añade el código que se desea compilar y ejecutar, extraído del editor de código de ACE. Una vez en el servidor este código se procesa, se compila, se ejecuta y se devuelve la respuesta a la web, como se ha explicado previamente, aunque se han tenido que añadir unas cabeceras específicas en el mensaje de respuesta del servidor, esta cabecera es la de CORS (Cross-Origin Resource Sharing) la cual es necesaria para que se permita la comunicación entre diferentes dominios evitando así que el navegador bloquee las respuestas del servidor como medida de seguridad. Finalmente, la web al recibir la respuesta del servidor trata el mensaje y lo presenta en la sección designada para mostrar los resultados del código, la cual también se basa en ACE por lo que se actualiza dinámicamente. El resultado de una ejecución en la web se puede observar en la Figura 3 del apéndice.

6.2 Desarrollo de los objetivos secundarios

Una vez completada la funcionalidad básica de la web, se avanza ahora a la siguiente fase, en la cual se añadirán una serie de funcionalidades necesarias para enriquecer la experiencia del usuario y mejorar la utilidad de la plataforma. Entre estas nuevas funcionalidades se incluirán opciones de personalización para el editor, la posibilidad de importar y exportar archivos, la posibilidad de registrarse e iniciar sesión, entre otras. Estas adiciones están diseñadas para proporcionar a los usuarios una experiencia más completa y adaptada a sus necesidades.

6.2.1 Opciones de Personalización

Se ha empezado por agregar varias opciones de personalización al editor para permitir que cada usuario adapte su experiencia según sus preferencias individuales. Esto incluye la capacidad de ajustar el tamaño de fuente del código y la posibilidad de seleccionar un tema para mejorar la legibilidad y la comodidad visual. Además, se ha incorporado la opción de cambiar el modo de lenguaje de programación, lo que permite al editor identificar y señalar correctamente los errores de sintaxis. Una función adicional que se ha añadido es la posibilidad de activar o desactivar el autocompletado, lo que puede aumentar la eficiencia del usuario al programar, ya que esta opción puede ayudar a prevenir errores tipográficos. Todas estas opciones de personalización están disponibles al hacer clic en el botón “Ajustes”, y al hacer clic nuevamente en el mismo botón, estas opciones se ocultan para mantener una interfaz más limpia. En la Figura 4 muestra este menú desplegado.

Figura 4 - Visualización del menú de ajustes.

6.2.2 Funcionalidad adicional

Otra función importante que se ha agregado es la capacidad de descargar el archivo actual. Al hacer clic en el botón “Descargar Archivo”, se te solicitará seleccionar la ubicación donde deseas guardar el archivo. El nombre del

archivo se asignará automáticamente en función del nombre asociado en la web. Además, la extensión del archivo se determinará automáticamente según el modo en el que se encuentre el editor en ese momento. Esta función proporciona una manera conveniente para que los usuarios guarden su código a nivel local de manera rápida y eficiente.

Además, se han añadido dos nuevas funcionalidades con los botones "Nuevo Archivo" y "Abrir Archivo". Estos botones permiten a los usuarios crear y abrir archivos en la web de manera conveniente.

Para ello se ha optado por un diseño de pestañas similar al de un navegador web para gestionar estos archivos. Cada pestaña tendrá un nombre único y un botón en forma de cruz que permite cerrar el archivo, eliminando así la pestaña correspondiente. Las pestañas son seleccionables, y al seleccionar una de ellas, el editor de código se ajustará para mostrar el contenido que se encontraba en esa pestaña y el modo que se haya seleccionado para la misma anteriormente. Al seleccionar una pestaña el color de esta variará de manera que el usuario pueda saber en qué pestaña se encuentra en cada momento.

El botón "Nuevo Archivo" crea una nueva pestaña vacía para que el usuario pueda comenzar a trabajar en un nuevo archivo. Se requerirá que el usuario ingrese un nombre para el archivo, el cual no puede contener caracteres especiales ni coincidir con el nombre de otra pestaña. Además, la plataforma limita el número de pestañas activas a cinco para una mejor organización.

Por otro lado, el botón "Abrir Archivo" permite al usuario seleccionar un archivo existente para abrirlo en una nueva pestaña. El archivo seleccionado debe cumplir con ciertos requisitos: no debe superar los 50 kB de tamaño y debe tener una extensión válida. Actualmente, solo se aceptan extensiones como .c, .cpp, .js, .cs, .py, que se corresponden a los lenguajes de programación que se tratarán en la web. El editor de código ajustará automáticamente el lenguaje de programación según la extensión del archivo seleccionado.

6.2.3 Inicio de Sesión

A continuación, se ha añadido otra funcionalidad fundamental para la plataforma, esta es la capacidad de iniciar sesión, lo que ha requerido la programación de una nueva página de registro de usuarios. En esta página, se ha agregado un formulario donde el usuario debe ingresar un nombre de usuario, un correo electrónico y una contraseña la cual debe escribirse dos veces para verificarla. El nombre de usuario no debe contener números ni caracteres especiales, en el caso del correo al especificar que es un campo en el que habrá un correo electrónico el propio formulario ya verifica automáticamente que este tenga las propiedades necesarias, finalmente los campos de contraseña y verificación de contraseña se han configurado como campos de tipo contraseña para que los caracteres ingresados no sean visibles y se muestren como asteriscos.

Después de ingresar todos los datos requeridos, al presionar el botón de registro, se activará una función JavaScript para verificar si las dos contraseñas ingresadas son iguales. Si no coinciden, se informará al usuario para que realice las correcciones necesarias. Si las contraseñas coinciden, se procederá a una verificación adicional en el servidor. Aquí, se volverá a validar que el nombre de usuario no contenga caracteres especiales ni números, se verificará nuevamente el formato del correo electrónico y se confirmará que las contraseñas coincidan. Para garantizar la seguridad de los datos, se preparará la conexión con la base de datos para evitar la posibilidad de inyecciones SQL. Además, antes de almacenar la contraseña en la base de datos, se aplicará un proceso de hash para protegerla tanto de posibles intercepciones como para protegerlo de alguien que obtuviese acceso a la base de datos. Finalmente, los datos del usuario se almacenarán de manera segura en la base de datos. En la Figura 5 se muestra la página de registro.

Figura 5 - Pagina de registro de usuarios.

La base de datos se ha creado utilizando PHPMyAdmin y MySQL. Esta base de datos incluye una tabla que almacenará el nombre de usuario, el correo electrónico, el hash de la contraseña y un identificador único para cada usuario.

Una vez que los usuarios se hayan registrado, podrán iniciar sesión a través de otro formulario ubicado en la página principal. Este formulario solicita dos campos: correo electrónico y contraseña. Además, se proporciona un enlace a la página de registro para aquellos usuarios que aún no se hayan creado una cuenta. Los campos de correo electrónico y contraseña están configurados con los tipos de entrada necesarios para asegurar que se ingrese un correo electrónico válido y que la contraseña se muestre de manera oculta.

Al enviar el formulario, se ejecuta una función asíncrona que recopila los datos y realiza una solicitud al servidor para su procesamiento, mediante un fetch. En el servidor, se verifica la validez del correo electrónico y se previene cualquier intento de SQL Injection, invalidando cualquier intento de inicio de sesión donde el correo o la contraseña contengan caracteres especiales relacionados con SQL.

Una vez validados, se realiza una consulta a la base de datos solicitando la información relacionada con el correo electrónico proporcionado. Antes de realizar esta consulta, los datos se transforman para evitar interpretaciones como código. Si el correo proporcionado no existe en la base de datos, se notifica al usuario que ha habido un error en el

inicio de sesión. Del mismo modo, si el correo existe, pero la contraseña es incorrecta, se informa al usuario de que ha habido un error en el inicio de sesión, en cualquier caso, no se concreta el motivo del error para evitar revelar información sobre los correos registrados en la base de datos.

Si todas las validaciones son exitosas, se inicializa una variable de sesión que almacena el nombre de usuario extraído de la base de datos, el correo proporcionado y un indicador booleano que señala que se ha iniciado sesión. Además, al iniciar sesión se actualiza el campo de inicio de sesión para darle la bienvenida al usuario, utilizando su nombre el cual se extrae de la variable de sesión y se le muestran dos enlaces, el primero de ellos permite al usuario acceder a la página de su cuenta y el segundo le permite cerrar sesión y eliminar la sesión activa, esto se muestra en la Figura 6.

Figura 6 - Visualización del inicio de sesión de un usuario, en este caso su nombre es Pedro.

6.2.3.1 *Gestión de Sesiones*

A continuación, se ha programado la funcionalidad para un nuevo botón que se ha añadido, este es el de "Guardar Sesión". En cuanto al botón, este permite al usuario almacenar los archivos que este esté desarrollando en la web en ese momento de manera que cuando el usuario vuelva más adelante pueda retomar su sesión anterior con mayor facilidad. Para ofrecer esta funcionalidad se ha tenido que actualizar la base de datos y se ha programado la función asociada a este botón. En cuanto a la base de datos, se ha creado una nueva tabla la cual tiene tres parámetros, el primero es un id para identificar cada entrada inequívocamente, el segundo es una clave foránea de manera que cada entrada en esta tabla debe estar relacionada con algún usuario de la tabla de usuarios y finalmente el último parámetro será donde se almacenen los datos de la variable que contiene el código de los como máximo cinco archivos que el usuario quiera almacenar. Por otro lado, la función asociada al botón empieza realizando una llamada, a través de un fetch, que transmite los datos almacenados en todos los archivos abiertos en ese momento por el usuario. El fetch redirige al controlador, el cual, tras verificar que el usuario tenga la sesión iniciada, pasa esta información al modelo. En este último, el acceso a la base de datos se realiza mediante una transacción que se divide en dos partes. En la primera, se elimina la sesión anterior, para lo cual es necesario comprobar si existe alguna, mientras que, en la segunda parte, se guarda la sesión actual, verificando primero que el número de archivos sea inferior a cinco.

Una vez implementada la capacidad de almacenar la sesión, se tuvo que diseñar la manera de cargar estos datos. La idea inicial fue que, al iniciar sesión, se sincronizaran los archivos almacenados en la sesión anterior automáticamente. Sin embargo, esta solución presentaba varios problemas, el primero de ellos era que, si la sesión permanecía activa y el usuario recargaba la página, los datos no se volverán a cargar ya que estos únicamente se inicializarían cuando el usuario inicia sesión, no si este ya tiene la sesión iniciada. Ante este problema una primera solución fue realizar una petición a la base de datos para cargar los archivos de la sesión anterior cada vez que el usuario recargase la página. No obstante, esta solución tampoco resultaba aceptable, ya que podría generar muchas peticiones a la base de datos si la página se recargaba repetidamente, lo cual no es deseable.

Otra idea fue utilizar el almacenamiento en caché del navegador para almacenar los datos una vez cargados por primera vez, de manera que se redujera el número de peticiones a la base de datos. Sin embargo, esta solución también presentaba inconvenientes. Era necesario editar la caché del usuario y buscar los datos, y si estos no estaban disponibles en la caché, se presentaría el mismo problema de antes, donde se debería acceder a la base de datos y se podrían producir múltiples peticiones.

Debido a los inconvenientes de estas soluciones finalmente, se ha optado por añadir un botón de "Cargar Sesión", el cual permite al usuario cargar los archivos de la sesión anterior al presionarlo. Esta solución ofrece al usuario más libertad y control sobre los datos que se almacenan y se muestran, permitiéndole decidir si vale la pena cargar su sesión anterior o si en ese momento no necesita los archivos, puede simplemente utilizar la página sin tener que tratar los datos almacenados previamente. En cuanto a la funcionalidad de este botón, este realiza una petición mediante un fetch en la cual el controlador analizará si el usuario ha iniciado sesión, en caso afirmativo se contacta con el modelo el cual realiza una petición a la base de datos para comprobar si existe una sesión previa. En caso negativo, no se carga nada y se informa al usuario de que no hay ninguna sesión guardada. En caso afirmativo, se carga la sesión anterior, eliminando todas las pestañas actuales, creando los elementos necesarios con los nombres de los archivos almacenados y actualizando la variable de archivos en la que se guardarán los datos de todos los archivos cargados.

6.2.4 Ampliación de lenguajes soportados

El siguiente objetivo implementado es dar soporte a un número mayor de lenguajes, no únicamente C. Se han añadido los siguientes lenguajes: Python, C++, JavaScript y C#. Para ello, se ha modificado la petición que se realiza al servidor, de modo que ahora, en función del lenguaje seleccionado en el desplegable de ajustes, se ajusta el mensaje de la petición para solicitar un tipo de ejecución u otro.

Desde el punto de vista del servidor se han instalado los compiladores y ejecutores necesarios para cada lenguaje. Para Python, se ha instalado Python 3; para C++, se ha utilizado g++ incluido en GCC; para JavaScript, se ha instalado Node.js; y finalmente, para C#, se ha instalado Mono.

Tras realizar las instalaciones necesarias se han programado todas las nuevas entradas en el servidor para tratar cada uno de los lenguajes. Todos ellos siguen el mismo procedimiento utilizado por las peticiones en código C, de manera que primero se crea un archivo temporal en el que se copia el código recibido, y luego, en función del lenguaje, se compila o ejecuta dicho código. Finalmente, el resultado de la ejecución se envía al usuario. En la Figura 7 se muestra como el usuario puede alternar entre lenguajes de programación mediante el desplegable de "modo" dentro de la sección de "ajustes".

Figura 7 - Se muestran los lenguajes de programación disponibles.

6.2.5 Refactorización y finalización de tareas

Llegados a este punto antes de seguir con la planificación se ha tenido que realizar un proceso de limpieza puesto que a lo largo del desarrollo de todas las tareas previamente mencionadas han ido quedando una serie de "todo's" los cuales se han ido acumulando hasta ahora. Además, dentro de los archivos del proyecto hay una gran cantidad de archivos y funciones que han quedado obsoletas y se han eliminado durante esta fase. En cuanto a los "todo's" a realizar, son los siguientes.

Dentro de los archivos en los que se lleva a cabo el registro de nuevos usuarios se han eliminado múltiples comentarios y prints innecesarios, transformándolos en logs que se presentan al usuario en caso de error. También se ha añadido una comprobación adicional para evitar que un usuario se registre con un correo electrónico ya existente.

También se ha actualizado el logo de la página web, ya que previamente no estaba centrado correctamente.

Se han revisado todos los patrones regex para asegurar que las restricciones sean consistentes en las diferentes partes de la aplicación.

Otra mejora añadida ha sido la adición de una advertencia al cerrar un archivo, informando al usuario de que los datos se perderán. Anteriormente, estos datos se perdían instantáneamente sin previo aviso.

Por último, cuando un usuario inicia sesión, tiene la posibilidad de cerrar sesión (funcionalidad programada en una fase anterior del proyecto), y de consultar y modificar sus datos personales. Estas dos últimas funcionalidades, que no se habían desarrollado aún, se han combinado en una misma página. En esta página, al usuario se le presentan diversas opciones organizadas en cinco paneles diferentes.

El primero de ellos es el de Información Personal, este panel muestra los detalles personales del usuario, como el nombre de usuario y el correo electrónico registrado. Estos datos son extraídos de la variable de sesión.

El segundo panel es el de Cambio de Información Personal, en este panel el usuario puede actualizar su nombre de usuario y correo electrónico. Este panel incluye campos de entrada para los nuevos datos y botones para guardar los cambios o cancelar la operación. Cuando el usuario introduce los datos y pulsa el botón de guardar cambios, se realiza primero un análisis para verificar si los datos son diferentes a los actuales. En caso afirmativo, se procede con una verificación para asegurar que no se produzca ninguna clase de inyección SQL y que cada campo respete las condiciones que le corresponde, de acuerdo con las restricciones mencionadas previamente. Donde los nombres solo permiten letras y no más de 10 caracteres y el correo electrónico debe seguir un formato válido, además se comprueba que el nuevo email no coincida con el de algún otro usuario. Una vez realizados los cambios, se actualiza tanto la base de datos como la variable de sesión de manera que el usuario puede observar los cambios instantáneamente.

El tercer panel es el de Cambio de Contraseña, en este el usuario puede cambiar su contraseña. Este panel contiene campos para la contraseña actual, la nueva contraseña y la confirmación de la nueva contraseña, junto con botones para guardar los cambios o cancelar la operación. Al insertar los datos en todos los campos, primero se verifica que la nueva contraseña sea igual a su confirmación de contraseña, para evitar que el usuario pueda cometer algún error al escribir su nueva contraseña, también se comprueba que la nueva contraseña no sea igual a la anterior comparando el hash de la nueva contraseña con la guardada en la base de datos. A continuación, se realizan las pruebas pertinentes para asegurar que la nueva contraseña no tenga menos de cinco caracteres y que solo contenga letras y números. Finalmente, antes de aplicar los cambios en la base de datos, se comprueba que la contraseña actual sea correcta. Si todas las verificaciones son satisfactorias, se realizan los cambios y se actualiza la contraseña del usuario.

El cuarto panel es el de Eliminación de Sesión, en este se le permite al usuario eliminar los datos almacenados en su sesión. Para llevar a cabo esta operación, primero se comprueba si existe alguna sesión almacenada. En caso negativo, se informa al usuario de que no hay ninguna sesión existente. Si existe una sesión guardada, esta se eliminará.

Finalmente, el quinto panel es el de Eliminación de Cuenta, en este el usuario puede eliminar su cuenta de manera permanente, para ello se proporciona un botón para realizar esta acción, y se notifica al usuario que será irreversible. Para llevar a cabo esta operación, primero se eliminan los datos que el usuario pueda haber almacenado. Una vez eliminados estos datos, si existían, se procede a eliminar el usuario de la base de datos. Tras finalizar la operación, se redirige al usuario a la página de inicio. Se puede observar esta página en la Figura 8 del apéndice.

6.2.6 Análisis de vulnerabilidades

A continuación, se ha avanzado a la siguiente tarea marcada por la planificación, esta es la de realizar un análisis de las vulnerabilidades de la aplicación. Para ello se han

dividido las vulnerabilidades en tres categorías diferentes en función de que parte de la aplicación afectan: el servidor de compilación, la base de datos y la propia página web en el navegador.

Dentro de la categoría del servidor de compilación se pueden destacar cinco vulnerabilidades principales a tratar.

La primera de ellas es la de **DDoS**, donde con esta vulnerabilidad se hace referencia a que si se hacen múltiples peticiones de larga duración se podría bloquear el servidor. Para tratar esta vulnerabilidad se va a limitar la duración máxima que una petición puede estarse procesando. Para ello se añadirá una función middleware en la que se concederá un tiempo máximo a cada petición entrante de manera que pasado este tiempo se detendrá el proceso y se notificará al usuario que se ha excedido el tiempo máximo permitido. Actualmente se ha optado por un tiempo máximo de cinco segundos.

La siguiente vulnerabilidad es **ejecución remota de código (RCE)**, esta vulnerabilidad es especialmente crítica para esta aplicación, ya que la aplicación permite directamente la ejecución de código. Esto facilita que un atacante pueda llevar a cabo un ataque de este tipo de manera muy sencilla. Debido a la diversidad de vulnerabilidades que existen dentro de esta categoría, se ha optado por una estrategia para minimizar el impacto y el alcance de estos ataques. Para ello se utilizará containerización de manera que se puedan aislar las ejecuciones de código del usuario del entorno donde se encuentra el servidor. Para llevar a cabo esta tarea se pueden emplear diversas técnicas y o herramientas, pero en este caso se ha optado por utilizar Docker debido al aislamiento y eficiencia que este ofrece, además se podría llegar a utilizar Docker Swarm en una implementación final para orquestar los diferentes contenedores. El procedimiento será el siguiente, se creará un contenedor Docker temporal para cada petición que el usuario haga. Antes de poder implementar esto, se han creado imágenes Docker para cada uno de los lenguajes soportados, de modo que cada imagen contiene lo necesario para ejecutar y compilar el código en un lenguaje. Cuando un usuario realiza una petición, se crea una carpeta específica y aislada para ese usuario. En esa carpeta se almacenan los archivos necesarios para ejecutar y compilar el código, junto con un script que ejecutará todas las operaciones necesarias según el lenguaje seleccionado. En cuanto a la creación del contenedor Docker, este utilizará una imagen específica según el compilador que el usuario necesite. Se le asignarán una serie de parámetros ajustables, entre ellos está la limitación de memoria a 128 MB por contenedor y el número de núcleos a 0.5. Adicionalmente, se añadirá la opción "--rm", la cual indicará al Docker que debe cerrarse automáticamente una vez termine de ejecutar el script. También se especificará el volumen que el Docker utilizará, este volumen es la carpeta de la petición que el Docker este tratando, aislándolo así del resto de la información. Esta función se combinará con la mencionada previamente de manera que el limitador de tiempo se aplique al contenedor Docker. Así, si el Docker no ha finalizado después de un tiempo establecido, se enviará un comando para eliminarlo y se informará al usuario que se ha sobrepasado el tiempo límite para la ejecución.

Otra vulnerabilidad explotable era la de **File Path Manipulation**, esta vulnerabilidad podría ser explotada si un atacante realizara una petición de manera manual, en lugar de a través de la página web. Esto sería posible si el campo de uniqueID, el cual se utiliza para nombrar los diferentes archivos que se compilan, fuera alterado, permitiendo así el acceso a ubicaciones a las que no se debería tener acceso. Para prevenir este tipo de ataques, se implementará una función que actúe como middleware antes de proceder con la operación principal. Este middleware analizará el uniqueID, el cual debe seguir el formato de UUID versión 5. Gracias a la propia biblioteca de UUID, se puede verificar directamente si un string es un UUID válido y si es de versión 5 o no. Si alguna de estas comprobaciones falla, la petición no se dirigirá al flujo principal, sino que se detendrá en este punto.

También se debe controlar la **ejecución de comandos**, esta vulnerabilidad radica en que en el servidor se utilizan comandos para compilar código, para ejecutarlo y para crear Dockers, lo que podría permitir la inserción de comandos maliciosos. Para prevenir la ejecución de comandos con privilegios elevados (como sudo), se creará un usuario no privilegiado en el cual se ejecutará la aplicación. Esto limitará el alcance de un posible ataque, ya que el usuario no tendrá acceso a recursos o acciones que podrían comprometer la seguridad del sistema.

Finalmente, la última vulnerabilidad de esta categoría sería que se produjese un ataque **DoS** debido al tamaño del archivo a procesar. Si los archivos a procesar son demasiado grandes todo el servidor podría colapsar con una única petición. Para mitigar esta vulnerabilidad, se ha establecido un límite máximo de tamaño de archivo a 50 kB tanto dentro de la aplicación web como en el servidor. Se han implementado dos comprobaciones en el servidor: una para verificar que el JSON sea inferior al límite mencionado y por si acaso otra dentro de la función de middleware para garantizar que el código extraído del JSON no ocupe más de 50 kB.

En cuanto a las vulnerabilidades relacionadas con la base de datos, estas han sido abordadas de manera continua a lo largo del proyecto. Se ha llevado a cabo un exhaustivo análisis y comprobación de la integridad de las consultas, asegurándose siempre de limpiar y evitar cualquier tipo de dato que pudiera ser característico de un ataque. Además, se ha implementado un enfoque seguro en el manejo de las consultas, donde estas no se realizan directamente, sino que primero se preparan y luego se adjuntan los valores asegurando así que estos se interpreten como strings y no como comandos.

Finalmente, para desalentar la inspección del flujo de la web en busca de vulnerabilidades, se implementará una herramienta de ofuscación. Esta herramienta permitirá transformar todos los archivos con scripts de manera que su lectura resulte altamente compleja, dificultando así la comprensión y el análisis por parte de posibles atacantes. Esta técnica de ofuscación ayudará a aumentar la

seguridad al agregar una capa adicional de protección, dificultando la identificación de posibles puntos débiles en el sistema. Para realizar esta tarea se ha utilizado JavaScript Obfuscator [10].

6.2.7 Últimos ajustes de la UI

Como ultima tarea del desarrollo de la aplicación se han implementado algunas mejoras en la interfaz de usuario para optimizar la experiencia del usuario al interactuar con la aplicación. Una de estas mejoras incluye la adición de confirmaciones adicionales al eliminar la cuenta del usuario. Esta medida adicional proporciona una capa de seguridad adicional, asegurando que las eliminaciones de cuentas se realicen de manera intencional, evitando así posibles errores. Además, en la página de "Mi Cuenta" se ha añadido un botón para hacer más intuitiva la manera de volver a la página principal. También se ha implementado una notificación visual al ejecutar código, junto con un cambio en el color del botón "Ejecutar", para informar al usuario que la operación está en curso, este efecto se puede observar en la Figura 9. Finalmente se ha realizado una revisión en el diseño y presentación del apartado mostrado al iniciar sesión con el objetivo de mejorar su apariencia y facilitar la comprensión de la información por parte de los usuarios.

Figura 9 - Visualización de la notificación visual al ejecutar código.

7 Conclusión

En este Trabajo de Fin de Grado, he demostrado mi capacidad para desarrollar desde cero una aplicación web real que requiere de múltiples módulos para operar. A lo largo del proyecto, he incrementado significativamente mi conocimiento y habilidades en el desarrollo web, manejando tanto tecnologías de frontend, como de backend. Se ha conseguido crear una aplicación web completa, la cual permite a los usuarios seleccionar entre diferentes leguajes de programación, ejecutar su código en un editor online y visualizar los resultados en un panel de salida. Se ha implementado una interfaz de usuario intuitiva y funcional que facilita la interacción del usuario con la aplicación.

Algunos de los puntos que no se han tratado durante este trabajo y que por tanto pueden ser parte de extensiones de este proyecto en el futuro son los siguientes. El primero de todos sería el **despliegue de la aplicación** en un servidor externo con las capacidades suficientes como para soportar una serie de requisitos mínimos, lo cual es un paso indispensable para la puesta en producción de cualquier aplicación web. Otro sería la **accesibilidad** de la aplicación desde diferentes dispositivos puesto que la experiencia de usuario puede no ser consistente. Otro aspecto a tratar podría ser el realizar un **análisis exhaustivo de las vulnerabilidades**, este campo es vasto y podría ser el foco de un proyecto completo dedicado a analizar y prevenir vulnerabilidades en aplicaciones web que permiten ejecutar código. Otro punto que se podría tratar sería la **optimización del rendimiento y la escalabilidad** de la aplicación, sobre todo en cuanto al tiempo de respuesta de los servidores, el cual puede llegar a ser algo más elevado de lo deseado. También se podría añadir **soporte multilingüe** de manera que en función de la ubicación del usuario la web se muestre en un idioma u otro. Una última extensión que se podría añadir en cuanto a la funcionalidad de la web sería añadir la opción de **depurar el código**, de manera que esta herramienta permita a los usuarios analizar su código línea por línea, facilitando la identificación y corrección de errores.

En conclusión, aunque el trabajo cumple con los objetivos iniciales y el resultado final es muy satisfactorio, existen diversas áreas en las que se podría ampliar y mejorar aún más esta aplicación, abriendo la puerta a futuras extensiones y mejoras continuas, como es habitual en el desarrollo de aplicaciones web.

Bibliografía

[1] ProyectosAgiles.org, «Desarrollo iterativo e incremental,» [En línea]. Available: https://proyectosagiles.org/desarrollo-iterativo-incremental/. [Último acceso: 1 3 2024].

[2] The PHP Group, «PHP - Manual,» [En línea]. Available: https://www.php.net/manual/en/intro-whatis.php. [Último acceso: 25 2 2024].

[3] Mozilla, «CSS,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics. [Último acceso: 16 5 2024].

[4] Mozilla, «HTML,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics. [Último acceso: 21 5 2024].

[5] Mozilla, «JavaScript,» [En línea]. Available: https://developer.mozilla.org/en-liUS/docs/Learn/Getting_started_with_the_web/JavaScript_basics. [Último acceso: 20 5 2024].

[6] Mozilla, «MVC,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Glossary/MVC. [Último acceso: 25 3 2024].

[7] Cloud9 IDE y Mozilla, «Ace - The High Performance Code Editor for the Web,» [En línea]. Available: https://ace.c9.io/. [Último acceso: 27 2 2024].

[8] StrongLoop/IBM, «Express - Node.js web application framework,» [En línea]. Available: https://expressjs.com/. [Último acceso: 29 2 2024].

[9] Docker Inc., «Docker,» [En línea]. Available: https://docs.docker.com/get-started/overview/ . [Último acceso: 18 5 2024].

[10] Richscripts Inc., «Javascript Obfuscator - Protects JavaScript code from stealing,» [En línea]. Available: https://javascriptobfuscator.com/. [Último acceso: 17 5 2024].

APÉNDICE

FIGURA 1. PLANIFICACIÓN INICIAL DEL PROYECTO

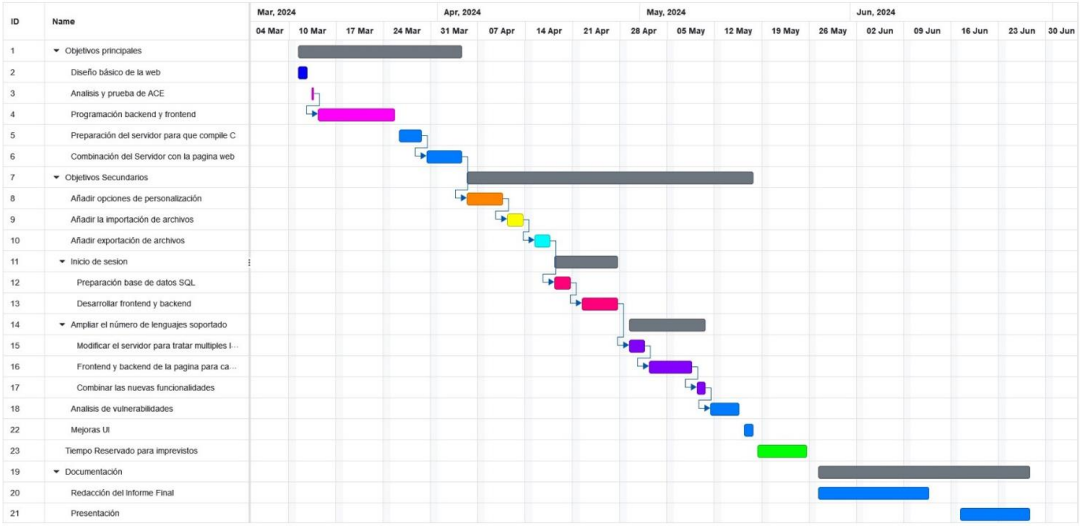

FIGURA 3. VISUALIZACIÓN DE LA APLICACIÓN TRAS EJECUTAR EL CÓDIGO C PROGRAMADO EN EL PANEL DE LA IZQUIERDA.

FIGURA 8. *MUESTRA DE LA PÁGINA DE "MI CUENTA", DONDE SE PUEDEN APRECIAR LOS CINCO PANELES DIFERENTES.*