Propuesta de enseñanza virtual del Lenguaje de Marcas de Hipertexto HTML mediante un Entrenador desarrollado a la medida de la propuesta

Elfar Didier Morantes Sánchez

Universitat Oberta de Catalunya

Universidad Autónoma de Bucaramanga

Maestría E-learning

Facultad Ciencias Sociales Humanidades y Artes

Bogotá

2023

Propuesta de enseñanza virtual del Lenguaje de Marcas de Hipertexto HTML mediante un Entrenador desarrollado a la medida de la propuesta

Elfar Didier Morantes Sánchez

Trabajo de grado presentado para optar el título de Magíster en E-Learning

Jorge Andrick Parra Valencia

(Director)

Universidad Oberta de Catalunya

Universidad Autónoma de Bucaramanga

Maestría E-learning

Facultad Ciencias Sociales Humanidades y Artes

Bogotá

2023

Nota de Aceptación:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Firma del presidente del jurado

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Firma del jurado

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Firma del jurado

Bogotá 2023

Dedicatoria:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Agradecimientos:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Contenido

[Resumen: 8](#_Toc137130325)

[Abstract: 9](#_Toc137130326)

[Introducción: 10](#_Toc137130327)

[Delimitación del Problema: 11](#_Toc137130328)

[Planteamiento del Problema: 11](#_Toc137130329)

[Justificación: 12](#_Toc137130330)

[Propuesta de Desarrollo de un Entrenador/Simulador 13](#_Toc137130331)

[Estado del Arte: 17](#_Toc137130332)

[Nacimiento del HTTP y Estandarización del HTML 17](#_Toc137130333)

[Consolidación del HTML 18](#_Toc137130334)

[Complementos del HTML 19](#_Toc137130335)

[Desarrollo Web: 20](#_Toc137130336)

[Socio Constructivismo 21](#_Toc137130337)

[Aprendizaje Experiencial 22](#_Toc137130338)

[Aprendizaje Basado en Problemas ABP 23](#_Toc137130339)

[Aprendizaje Cooperativo 24](#_Toc137130340)

[Descripción de la Experiencia: 25](#_Toc137130341)

[Investigación de modelos, interfaces, y lenguajes de programación aptos para desarrollar el simulador para el aprendizaje del lenguaje de marcas HTML 27](#_Toc137130342)

[Lenguaje de programación a utilizar 29](#_Toc137130343)

[Investigación de la forma en que serán presentados los conceptos en el simulador (figuras, formas, colores, animación, ejemplos, situaciones, organización de conceptos) 30](#_Toc137130344)

[Modelo 30](#_Toc137130345)

[Formato de presentación del sistema drag and drop para el aprendizaje del lenguaje de marcas HTML 31](#_Toc137130346)

[Interfaces 31](#_Toc137130347)

[Investigación de las teorías de aprendizaje subyacentes, modelos o enfoques pedagógicos, y métodos pedagógicos a utilizar junto con el simulador 33](#_Toc137130348)

[Formato de presentación del sistema de edición-visualización simultáneo para el aprendizaje del lenguaje de marcas HTML 35](#_Toc137130349)

[Prototipo Obtener código HTML 35](#_Toc137130350)

[Prototipo HTML editable en tiempo real 37](#_Toc137130351)

[Prototipo de Sistema Drag and Drop para el Simulador 38](#_Toc137130352)

[Integración de HTML editable en tiempo real y el Prototipo de Sistema Drag and Drop para el Simulador 41](#_Toc137130353)

[Desarrollo del simulador de acuerdo con el modelo, lenguaje de programación, interfaces y formato de presentación seleccionados 46](#_Toc137130354)

[Interfaz Gráfica 46](#_Toc137130355)

[Sección de Componentes 47](#_Toc137130356)

[Sección de Arrastrar y Sección de Código 47](#_Toc137130357)

[Funcionalidad de Arrastrar y Soltar 48](#_Toc137130358)

[Edición de Código 49](#_Toc137130359)

[Eliminación de Componentes 50](#_Toc137130360)

[Código fuente completo 50](#_Toc137130361)

[Implementación del Simulador en página web para probar flexibilidad y portabilidad 50](#_Toc137130362)

[Despliegue del Desarrollo Web en GitHub Pages 51](#_Toc137130363)

[Despliegue del Desarrollo Web en didiermorantes.com 51](#_Toc137130364)

[Diseño de actividades relacionadas con el simulador y las teorías, modelos y enfoques pedagógicos seleccionados 52](#_Toc137130365)

[Actividad 1 52](#_Toc137130366)

[Actividad 2 52](#_Toc137130367)

[Actividad 3 52](#_Toc137130368)

[Actividad 4 53](#_Toc137130369)

[Actividad 5 53](#_Toc137130370)

[Actividad 6 53](#_Toc137130371)

[Actividad 7 53](#_Toc137130372)

[Actividad 8 54](#_Toc137130373)

[Actividad 9 54](#_Toc137130374)

[Actividad 10 54](#_Toc137130375)

[Actividad 11 55](#_Toc137130376)

[Actividad 12 55](#_Toc137130377)

[Actividad 13 55](#_Toc137130378)

[Actividad 14 55](#_Toc137130379)

[Actividad 15 56](#_Toc137130380)

[Pruebas de desempeño del simulador implementado como herramienta de aprendizaje 56](#_Toc137130381)

[Resultados Alcanzados 57](#_Toc137130382)

[Conclusiones y Recomendaciones 58](#_Toc137130383)

[Referencias bibliográficas 59](#_Toc137130384)

Tabla de Ilustraciones

[Ilustración 1. Prueba de código para visualización de HTML 36](#_Toc137130385)

[Ilustración 2. Resultado de presionar el botón “Solo Texto” 36](#_Toc137130386)

[Ilustración 3. Resultado de presionar el botón “HTML” 36](#_Toc137130387)

[Ilustración 4. Prueba de código para visualización de HTML en tiempo real 37](#_Toc137130388)

[Ilustración 5. Resultado de escribir código HTML en el contenedor editable 38](#_Toc137130389)

[Ilustración 6. Prueba de código de simulador en su versión minimalista y preliminar 40](#_Toc137130390)

[Ilustración 7. Resultado de probar el prototipo del simulador 41](#_Toc137130391)

[Ilustración 8. Resultado de probar el prototipo del simulador integrado con el módulo de HTML editable 43](#_Toc137130392)

[Ilustración 9. Varios componentes arrastrados al área de trabajo 44](#_Toc137130393)

[Ilustración 10. Componentes retirados del área de trabajo. 44](#_Toc137130394)

[Ilustración 11. Escritura de instrucciones en el área de código y visualización en el área de trabajo 45](#_Toc137130395)

[Ilustración 12. Componente no actualizado en el área de código. 46](#_Toc137130396)

[Ilustración 13 . Nueva Interfaz Gráfica 47](#_Toc137130397)

[Ilustración 14. Sección de Componentes 47](#_Toc137130398)

[Ilustración 15. Sección de Arrastrar Componentes y Sección de Código 48](#_Toc137130399)

[Ilustración 16. Funcionalidad de Arrastrar y Soltar 49](#_Toc137130400)

[Ilustración 17. Edición de Código 50](#_Toc137130401)

[Ilustración 18. Eliminación de Componentes individualtes 51](#_Toc137130402)

[Ilustración 19. Eliminación de componentes al seleccionar layout 51](#_Toc137130403)

[Ilustración 20. Repositorio Github con el archivo javascript que posee la funcionalidad de cambiarElemento 57](#_Toc137130404)

Tabla de Código

[Código 1. Página web simple con una caja de texto y dos botones 35](#_Toc137130405)

[Código 2. Página web simple con contenedor editable de HTML 37](#_Toc137130406)

[Código 3. Versión preliminar y minimalista del simulador 40](#_Toc137130407)

[Código 4 Integración del módulo de HTML editable y el prototipo del simulador 43](#_Toc137130408)

# Resumen:

¿Cómo facilitar el proceso de enseñanza-aprendizaje de los conceptos básicos de la programación en lenguaje de marcas HTML y superar el escollo para asimilar dichos conceptos mediante el uso de una base pedagógica en combinación con un entrenador? La propuesta de este trabajo de grado es, **por un lado, desarrollar un entrenador** que le permita al estudiante, en especial al estudiante a distancia virtual, precisar y definir los conceptos abstractos de la programación en lenguaje de marcas HTML, de manera que tenga unas bases sólidas para continuar su aprendizaje de conceptos más avanzados de forma autónoma, utilizando las facilidades, ayudas y características que le ofrece el entrenador mediante el aprendizaje activo y colaborativo. **Por otro lado, proponer la utilización de un modelo pedagógico** **que sirva de soporte y andamiaje para que el entrenador se pueda utilizar efectivamente** en la construcción de conocimientos y habilidades relacionadas con el desarrollo de páginas web. El método consiste en desarrollar un sistema para generar componentes del tipo drag and drop –arrastre y suelte-, de manera que el estudiante inicialmente no tendrá que saber cómo construir algún elemento del lenguaje de marcas HTML, sino que sencillamente lo arrastrará de un menú que contiene todos los elementos esenciales para construir formularios, y de la utilización de un modelo pedagógico que soporte y coadyuve al proceso de enseñanza-aprendizaje de desarrollo de páginas web utilizando como herramienta el entrenador desarrollado. La base del modelo pedagógico será la teoría del aprendizaje Socio Constructivista, utilizando como enfoque o modelo pedagógico el aprendizaje experiencial de la Escuela Activa o Escuela Nueva, e implementando el aprendizaje basado en problemas ABP y el aprendizaje cooperativo como métodos pedagógicos que se centran en el estudiante y promueven el aprendizaje y la socialización entre los estudiantes.

**Palabras Claves:** Entrenador, Programación, Html, Formularios, Aprendizaje a Distancia, Socio Constructivismo, Aprendizaje Experiencial, Aprendizaje Basado en Problemas, Aprendizaje Cooperativo

**Línea de investigación:** Pensamiento sistémico y educación

Abstract:  
How to facilitate the teaching-learning process of the basic concepts of programming in HTML markup language and overcome the obstacle to assimilate these concepts by using a pedagogical base in combination with a trainer? The proposal of this degree work is, on the one hand, to develop a trainer that allows the student, especially the virtual distance student, to specify and define the abstract concepts of programming in HTML markup language, so that they have a solid foundation to continue their learning of more advanced concepts autonomously, using the facilities, aids and features that the trainer offers you through active and collaborative learning. On the other hand, propose the use of a pedagogical model that serves as support and scaffolding so that the trainer can be used effectively in the construction of knowledge and skills related to the development of web pages. The method consists of developing a system to generate components of the drag and drop type, so that the student initially will not have to know how to build any element of the html markup language, but will simply drag it from a menu that It contains all the essential elements to build forms, and the use of a pedagogical model that supports and contributes to the teaching-learning process of developing web pages using the developed trainer as a tool. The basis of the pedagogical model will be the Socio-Constructivist learning theory, using the experiential learning of the Active School or New School as an approach or pedagogical model, and implementing PBL problem-based learning and cooperative learning as pedagogical methods that focus on learning and the student and promote learning and socialization among students.

**Keywords:** Trainer, Programming, Html, Forms, Distance Learning, Socio-Constructivism, experiential learning, problem-based learning PBL, cooperative learning

**Research line:** Systemic thinking and education

# Introducción:

La enseñanza de los lenguajes de programación en las carreras técnicas, tecnológicas y profesionales relacionadas con los Sistemas, la Electrónica y las Computadoras es esencial para desarrollar competencias básicas en los aspirantes a estas titulaciones. Sin embargo, la enseñanza de estos lenguajes no es una actividad que se realice de manera orgánica ni didáctica, y muchas veces se reduce a remitirse a los libros y a practicar con el código. Como lo indican algunos investigadores del campo de la enseñanza de la programación “*Ver la solución de un problema en un libro, o ver al profesor construyéndola en el tablero, no garantiza la adecuada generación de habilidades en el estudiante. Y aunque en un porcentaje significativo de los casos este enfoque funciona adecuadamente (así se ha enseñado programación durante los últimos 20 años), el problema es que su éxito depende de factores que no controlamos. (Villalobos, Casallas, Marcos 2005).”* Posiblemente estas falencias se deban a que existen muy pocas herramientas que sean intuitivas para el usuario y le permitan apropiarse cómodamente de los conceptos asociados a los lenguajes de programación. Indudablemente, si se quiere aprender a hacer pan, se tendrá que interactuar con la masa, untarse las manos, amasar, y meter al horno dicha masa para obtener resultados. Así mismo, si se quiere aprender a programar, se tiene que interactuar con el código, manipularlo, depurarlo, y modificarlo una y otra vez para obtener resultados. Como lo indica Jiménez-Toledo, Collazos, y Revelo-Sánchez en sus comentarios acerca de la deficiente fundamentación en los procesos de aprendizaje y los inconvenientes sobre la enseñanza de la programación: “*Dichos problemas se deben a múltiples circunstancias que surgen por el desconocimiento de conceptos fundamentales de programación, la falta de habilidades para modelar y construir un programa computacional, e incluso la poca disciplina al momento de la construcción cognitiva requerida para enfrentar los fundamentos de la programación (Jiménez-Toledo, Collazos, Revelo-Sánchez, 2019).”.* Para quienes se han dedicado al campo del desarrollo de software y la programación, es palpable la falta de habilidades del cuerpo docente para transmitir el conocimiento, y la consecuente falta de fundamentación en el estudiante, quien finalmente es el que tiene que tratar de resolver sus falencias y falta de habilidades por sus propios medios, y con sus propios recursos, recurriendo a sitios web especializados o videos que puedan suplir sus necesidades. No obstante, surge una pregunta: ¿Habrá una forma diferente y más entretenida de enseñar los lenguajes de programación? ¿Existirá una forma más “amigable” de acercar los conceptos de la programación que sea diferente a tratar de imaginar los conceptos abstractos presentados por un libro o por un profesor? Como lo mencionan Villalobos, Casallas, Marcos 2005 con respecto a presentar conceptos en clase o demostrar la forma de utilizar alguna estructura de programación mediante la solución de un ejercicio,  *“Dicha manera de enseñar a programar se basa en la esperanza de que el estudiante sea capaz de detectar patrones en los problemas planteados y los logre asociar con las técnicas que usa el profesor cuando desarrolla un ejemplo en el tablero y que luego, el estudiante sea capaz de hacer la generalización adecuada en su cabeza, para poder aplicar esa asociación patrón-técnica en la resolución de otros problemas (Villalobos, Casallas, Marcos 2005).”* La enseñanza de la programación actualmente se trata de un modelo en el que el docente intenta solucionar un ejercicio (en el mejor de los casos un problema) con la ilusión de que sus alumnos comprendan la mecánica detrás de la solución del problema, y traten de replicarlo en un ejercicio similar.

# Delimitación del Problema:

Evidentemente existe un vasto acervo científico y documental sobre los lenguajes de programación, y existen tantos lenguajes de programación casi como dialectos en el planeta y, por tanto, es necesario delimitar este dilema a un solo lenguaje de programación, de manera que sea viable analizar la posibilidad de construir una herramienta, simulador o entrenador que facilite la adquisición de los conocimientos relacionados con dicho lenguaje de programación. Para este caso particular, se seleccionará el lenguaje de marcas de hipertexto HTML por su gran utilidad y versatilidad en el desarrollo de páginas de Internet. En el siglo XXI la mayoría de la información se encuentra disponible en la red, y por tanto, es pertinente seleccionar el lenguaje HTML como objeto de estudio para construir un simulador o entrenador que permita facilitar el aprendizaje de las nociones asociadas a este lenguaje, y allanar el proceso de enseñanza-aprendizaje relacionado.

# Planteamiento del Problema:

Quien haya tenido experiencia o relación desarrollando software en lenguajes de marcas como HTML habrá notado que este lenguaje está conformado por una gran cantidad de conceptos abstractos tales como etiquetas, botones, cajas de texto, identificadores, eventos, entre otros conceptos fundamentales de este lenguaje. Para algunos estudiantes es difícil comprender estos conceptos que, inicialmente, no tienen muchos referentes palpables, y otros estudiantes tardan un tiempo considerable en comprender la esencia de estas ideas.

El asunto se torna más interesante cuando el desarrollo de estas competencias de programación se realiza a través de la modalidad a distancia virtual. Es el aprendiz quien por sí mismo debe construir sus definiciones y conceptos a partir del material de estudio proporcionado. Evidentemente tendrá la colaboración del tutor a través de foros y otros espacios destinados para la comunicación, pero es claro que el material proporcionado al estudiante debe ser de una calidad tal que le permita al estudiante desarrollar sus facultades intelectuales y construir conocimiento. Además, otro apuro a considerar en la programación en lenguaje de marcas HTML es el proceso de editar el código, y luego visualizarlo. Normalmente este es un proceso que consta de dos partes separadas en el tiempo, pues en primer lugar se debe escribir el código con un editor de texto plano, guardar los cambios con una extensión de archivo .html, y finalmente visualizar los productos con otra herramienta. Es decir, primero se debe escribir el código con una herramienta, y luego visualizar los frutos en otra. Esto implica que el estudiante debe realizar varios pasos, normalmente secuenciales y con diferentes herramientas, antes de poder percibir algún cambio en el proyecto web que esté desarrollando. La dificultad se incrementa si quien realiza estos pasos es una persona con poco conocimiento o contacto con la tecnología, o simplemente es una persona que no tiene familiaridad con las computadoras como por ejemplo un adulto mayor. En este punto, y habiendo identificado los distintos dilemas, se podría formular la siguiente pregunta: ¿Será posible construir una herramienta que le permita al estudiante apropiarse de estos conceptos abstractos, de manera que se vuelvan concretos, precisos y palpables? ¿Será factible desarrollar un instrumento que facilite el proceso de edición y visualización de código html, de manera que el estudiante experimente simultáneamente la escritura y la percepción de la página web?

# Justificación:

Si se define someramente un simulador como una herramienta que permite reproducir sensaciones, experiencias o comportamientos, entonces, se podría pensar en la herramienta que se quiere desarrollar para el estudiante como un simulador. En esta investigación se podrá usar indistintamente la palabra simulador y entrenador, pues el simulador permitirá reproducir experiencias similares a las reales (programando una página web), y el entrenador le permitirá seguir aprendiendo y seguir entrenándose en los campos relacionados con el desarrollo de páginas web. Pero, ¿Será posible que un simulador/entrenador le permita al estudiante acercarse más a los conceptos básicos del lenguaje de marcas HTML? ¿Podrá el simulador/entrenador reproducir la experiencia de programar, pero haciéndolo de una manera más sencilla, y quizás, hasta intuitiva y divertida?

Como lo indica Villalobos 2009, “*Como parte importante de las estrategias que se desarrollan frente al problema de enseñar a programar, se encuentran las herramientas de soporte. Principalmente herramientas de visualización y animación de algoritmos. Los instructores y profesores han encontrado gran soporte en estas herramientas cuando se mide el impacto que tienen en las sesiones de clase y laboratorios. Estas herramientas involucran a los estudiantes en diversos niveles de compromiso con el aprendizaje. (Villalobos 2009)”*. Son muchos los casos que ilustran la utilidad de los simuladores y su habilidad para reproducir ciertos comportamientos o experiencias. Por ejemplo, muchos pilotos de carreras entrenan con simuladores de autos para mejorar sus tiempos e incrementar sus habilidades. Los pilotos de aviones y helicópteros entrenan en simuladores de vuelo para reproducir ciertas experiencias críticas y de esa manera, estar preparados para eventuales situaciones. Si los simuladores han tenido éxito en estos campos, entonces se podría pensar que un simulador de programación le permitirá al estudiante reproducir la experiencia de programar en un lenguaje de marcas HTML de una manera más amigable, haciendo los conceptos más relevantes, concretos y aplicables, coadyuvando al desarrollo de las competencias orientadas a la programación, no solo como una herramienta complementaria a la formación impartida por el tutor y la información que reside en los libros, sino como un elemento de aprendizaje autocontenido que le permite al estudiante por sí mismo apropiarse de los conocimientos.

Redondeando todas estas ideas, se puede afirmar que se ha identificado una problemática: ¿Cómo facilitar el proceso de enseñanza-aprendizaje de los conceptos básicos de la programación en lenguaje de marcas HTML y superar el escollo para asimilar dichos conceptos mediante el uso de un simulador?

# Propuesta de Desarrollo de un Entrenador/Simulador

La propuesta de este proyecto es, por un lado, desarrollar un simulador que le permita al estudiante, en especial al estudiante a distancia virtual, precisar y definir los conceptos abstractos de la programación en lenguaje de marcas HTML, haciendo énfasis en aquellos conceptos y nociones relacionados con los formularios web – herramienta primordial para el intercambio de información entre un usuario y un servicio-, de manera que tenga unas bases sólidas para continuar su aprendizaje de conceptos más avanzados de manera autónoma, utilizando las facilidades, ayudas y características que le ofrece el simulador. La forma en que este simulador le simplificará al estudiante el proceso de apropiarse de los conceptos será porque utilizará un sistema para generar componentes del tipo drag and drop –arrastre y suelte-, de manera que el estudiante inicialmente no tiene que saber cómo construir un botón o una caja de texto en el lenguaje de marcas html, sino que sencillamente lo arrastrará de un menú que contiene todos los elementos esenciales para construir formularios. Además, en tiempo real podrá observar el código html asociado a dicho elemento, y podrá cambiar sus propiedades y verificar su comportamiento simultáneamente, sin necesidad de guardar el código, y refrescar el navegador para percibir los cambios. Inclusive el estudiante podrá ir más allá del sistema de “arrastre y suelte” y podrá seguir escribiendo código html adicional, de manera que pueda añadir nuevas características y nuevas funcionalidades a su proyecto web. Finalmente, el estudiante podrá copiar el código generado a partir de arrastrar y soltar componentes, y podrá utilizarlo en entornos de producción de la vida real.

**Propuesta pedagógica para usar el entrenador/simulador**

Por otro lado, se pretende articular la propuesta de enseñanza del lenguaje de marcas de hipertexto HTML con el entrenador/simulador desarrollado, sobre un modelo pedagógico que soporte y coadyuve al proceso de enseñanza-aprendizaje de desarrollo de páginas web. Se utilizará como teoría de aprendizaje subyacente el Socio Constructivismo propuesto por Lev Siminovach Vigostky, en el que el docente toma un papel de guía o facilitador en vez de tener un papel de proveedor de contenido, y donde los estudiantes toman un rol activo en la construcción de sus propias representaciones de la realidad y propio proceso de aprendizaje, interactuando con otros y trabajando en equipo. El socio constructivismo juega un papel catalizador en el uso del entrenador/simulador desarrollado, pues permite que los estudiantes construyan sus propios conceptos entre ellos, con la ayuda de una herramienta –el entrenador/simulador- que les permitirá conocer y comprender los principios detrás de los elementos constitutivos de las páginas web. Se utilizará como enfoque o modelo pedagógico el aprendizaje experiencial de la Escuela Activa o Escuela Nueva, que se centra en los intereses espontáneos del estudiante, y fortalece su actividad, libertad y autonomía. Es así que el aprendizaje experiencial sirve como apoyo al socio constructivismo con cinco características clave: “*la experiencia es la base o estímulo para el aprendizaje, los alumnos construyen su propia experiencia activamente, el aprendizaje es un proceso holístico, el aprendizaje se construye social y culturalmente, el aprendizaje se ve influenciado por el contexto socio-emocional en el que se realiza. (Boud, Cohen y Walker 1993)”.* Por tanto, el socio constructivismo y aprendizaje experiencial funcionan como columnas vertebrales de la propuesta pedagógica, puesto que se trata de lograr un aprendizaje con las bases constructivistas y construccionistas planteadas por J.Piaget y S.Papert, que indican que *“el verdadero aprendizaje va mucho más allá del simple hecho de recibir información o de adherirse a las ideas o valores de otras personas, es expresar nuestras ideas al mundo o encontrar nuestra propia voz e intercambiar nuestras ideas con otras personas”. (González, C. 2018).* Se quiere lograr que los estudiantes generen su propio conocimiento cambiando el paradigma de docente como fuente universal de conocimiento, o los libros como recursos exclusivos de información. Se pretende que los estudiantes generen su propio conocimiento a través de la práctica, la exploración y la experimentación. Como lo afirma Gonzalez 2018 “*la construcción de un nuevo aprendizaje es más eficiente cuando los estudiantes se comprometen en la elaboración, por sus propios medios, de un objeto tangible con alguna representación significativa para éstos. Es lo que llama Aprender haciendo… (Gonzalez 2018)”.* De hecho, es lo que se ha procurado plasmar en este trabajo de grado. Se ha querido construir un artefacto significativo (el entrenador/simulador) que permita aplicar todo el conocimiento obtenido durante la maestría en e-learning, de manera que se pueda utilizar en un contexto real de estudiantes, y que permita aplicar distintos conceptos pedagógicos como teorías de aprendizaje, modelos pedagógicos, y métodos pedagógicos.

Finalmente, se utilizará el aprendizaje basado en problemas ABP como método pedagógico que se centra en el estudiante, y en el que el estudiante adquiere habilidades, conocimientos y aptitudes a través de situaciones o problemas de la vida real. También se utilizará el aprendizaje cooperativo como un método pedagógico que promueve el aprendizaje y la socialización entre los estudiantes, en el que el alumnado trabaja conjuntamente para alcanzar metas comunes, maximizando su propio aprendizaje y el de los demás miembros.

La intención es que al estudiante se le facilite el proceso enseñanza-aprendizaje de los conceptos relacionados con los formularios HTML, utilizando una única herramienta que le permita arrastrar/escribir código para componentes, verificar su funcionamiento, y realizar cambios en tiempo real, simplificando los procedimientos relacionados con la creación de los mismos, y aplicando a la vez un modelo pedagógico adecuado para la utilización de la herramienta, de manera que se puedan explotar las ventajas socio constructivistas del aprendizaje cooperativo centrado en el estudiante. Como lo indica Villalobos 2009 “*En la búsqueda de propuestas actuales frente al tema de innovar la enseñanza de la programación, muchos autores concuerdan con la necesidad de integrar la generación de habilidades más que la transmisión de conocimientos planos, especialmente en el nivel de educación superior. (Villalobos 2009)”.* Por tanto, se espera que el estudiante pueda desarrollar habilidades de creación de páginas web con el uso del simulador, más que transmitir el concepto de etiqueta, o el concepto de página web.Evidentemente, desarrollar un simulador que abarque todos los conceptos del lenguaje de marcas HTML es pretencioso y podría convertirse en una tarea compleja de realizar. Por tal razón, el simulador solamente incluirá conceptos relacionados con los formularios web, pues son la herramienta por antonomasia para la comunicación entre un ser humano y una página web, de manera que se pueda verificar la apropiación de los conceptos relacionados con la construcción de elementos HTML de una página web. Así mismo, abarcar en un proceso de enseñanza todas las teorías del aprendizaje, modelos pedagógicos, enfoques y métodos pedagógicos no sería conveniente, práctico ni útil, por lo que se propone integrar únicamente como teoría de aprendizaje subyacente el socio constructivismo apoyado en el aprendizaje experiencial como enfoque o modelo pedagógico y los métodos pedagógicos del aprendizaje basado en problemas y el aprendizaje colaborativo para desarrollar conocimiento utilizando como herramienta el simulador desarrollado.

# Estado del Arte:

## Nacimiento del HTTP y Estandarización del HTML

En 1989, Tim Berners-Lee (conocido como el padre de la World Wide Web actualmente) se encontraba trabajando en el departamento de servicios informáticos en el CERN (Conseil Européen pour la Recherche Nucléaire, Organización Europea para la Investigación Nuclear ) cuando tuvo la idea de permitir el acceso a la información de diferentes colaboradores ubicados en distintas partes del mundo, pero no para poder descargar la información, sino para que pudieran enlazar la información en los mismos archivos (Raggett; Lam; Alexander y Kmiec,1998). La idea concebida era que mientras se estaba leyendo un documento, se podría acceder a cualquier parte de otro documento. Es decir, el texto permitiría mucho más que su lectura, y serviría como una estructura organizativa que enlazaría distintos documentos. Las capacidades del texto serían hiper-desarrolladas, y por tal razón empezaría a consolidarse el nombre de hipertexto.

La palabra hipertexto no era nueva para ese entonces, sino que era un concepto académico que existía desde 1940. De hecho, la primera persona en usar esta expresión fue Ted Nelson, en su artículo “No more teacher´s dirty looks” (Nelson,1970), para referirse a escritos no secuenciales que coordinaran la presentación de cualquier tipo de información, texto, imagen o audio. Un entorno en el que el usuario pudiera interactuar con la información de varias maneras. Este concepto de hipertexto se mantiene hasta el día de hoy, como base para todas las páginas web, donde se presenta una información que está enlazada o vinculada con otra.

Después de trabajar en esta idea de documentos vinculados, Berners-Lee creó un protocolo, el Hyper Text Transfer Protocol (HTTP de aquí en adelante), que era un protocolo muy simple con el cual se podía implementar un formato de texto con marcas hiperdesarrollado (HTML de aquí en adelante), en cualquier máquina, independientemente del sistema operativo que utilizase (Davison y Chen, 1995). Este protocolo HTTP permitía la independencia de la máquina en la que se estuviese trabajando (PC,Macintosh, Unix, entre otros), y la independencia del formato de texto (SGML, Interleaf, LaTex, Word, Troff, entre otros) para implementar el formato de texto HTML y además, serviría como estándar para transferir hipertexto en las máquinas existentes de la época. Desde entonces, HTTP se consolidó como el protocolo estándar utilizado por las transacciones en internet y como el sistema mediante el cual se envían las peticiones para acceder a una página Web y mediante el cual dicha página responde. Este protocolo envía el hipertexto HTML desde el servidor donde esté alojada la página web, para que el cliente (navegador web que solicita la información) lo disponga en la pantalla del cibernauta. Habiendo solucionado el inconveniente de transmitir hipertexto a distintas máquinas sin importar su arquitectura o su tecnología de red mediante el protocolo HTTP, el lenguaje de marcas de hipertexto HTML se consolidó como el estándar para crear contenido vinculado que pudiese ser compartido, visualizado y consumido en la World Wide Web. Este lenguaje de texto con marcas HTML se basa principalmente en un sistema de etiquetas que indica al navegador (software utilizado para visualizar/consumir contenido de internet) dónde está el inicio de un documento, dónde está el cuerpo de un documento, dónde y cuándo colocar una imagen, dónde y cuándo colorear un texto, dónde está el final del documento, entre otras instrucciones.

## Consolidación del HTML

El lenguaje HTML posee distintas características que hicieron que fuese fácilmente aceptado por la comunidad y la industria para estructurar las páginas web, sin embargo, un factor trascendental para facilitar dicha aceptación fue sin duda que su base y cimientos estaban en otro lenguaje de hipertexto, como era “Standard Generalized Mark-up Language” (de aquí en adelante SGML). SGML había sido inventado en 1979 por Charles Goldfarb y era un lenguaje de hipertexto que funcionaba en cualquier máquina reconocido internacionalmente. La idea central detrás de SGM era que el lenguaje fuese independiente del “formateador” (el navegador u otro software) que dispusiera la información en la pantalla. Este lenguaje introducía el concepto de separar la estructura del contenido, de su presentación (Cailliau, 1995). De esa forma, HTML se volvió popular al estar basado en un lenguaje versátil ya existente. También incidió en la popularización del lenguaje HTML la acción de su creador de divulgar su producto y fomentar la discusión abiertamente alrededor de toda la World Wide Web. Para 1992, unos cuantos académicos e informáticos habían mostrado interés por HTML y por su mejora, entre ellos Dave Raggett, de los laboratorios de Hewlett-Packard. Después de unas cuantas discusiones electrónicas Raggett visitó a Berners-Lee en 1992 en el CERN y discutieron sobre cómo convertir HTML en un “producto de masas”, procurando definir las características que un usuario pudiera ver como útiles e importantes. En el transcurso de 1992 muchos entusiastas del novedoso lenguaje de hipertexto se reunirían en torno al grupo de discusión recientemente creado, con el nombre de WWW. Entre ellos estarían Dave Raggett, Tim Berners-Lee , Dan Connolly entre otros muchos más, y con el tiempo, este grupo de discusión daría origen al Consorcio WWW, en inglés World Wide Web Consortium (W3C), que es un consorcio internacional que genera recomendaciones y estándares que aseguran el crecimiento de la World Wide Web a largo plazo. Gracias al esfuerzo de investigadores como Dan Connolly, en 1993 se recopilan la mayoría de las etiquetas que eran usadas por los navegadores que surgen entre 1993 y 1994, de manera que se pudiese estandarizar aún más el lenguaje HTML que estaba teniendo auge por esos días. Connolly recoge la mayoría de las sugerencias de entusiastas de HTML y las integra en un documento que hace circular por la red, preparando un borrador de lo que más tarde se conocería como HTML2.

Durante 1995, nuevos tipos de etiquetas de HTML surgen y crean bastantes discrepancias sobre su utilización, y así, en el transcurso de ese año, surge la primera versión de HTML3, publicada como un borrador por Dave Raggett, quien había estado trabajando en mejorar las características y etiquetas de HTML. También en 1995 aparece un nuevo elemento en el campo de los navegadores web. Microsoft lanza la primera versión de Internet Explorer, con la intención de competir con los navegadores populares de la época (Netscape) y desarrollar sus propias características de HTML. La entrada de Microsoft populariza aún más el uso del lenguaje HTML, y se convertiría más tarde en una de las organizaciones que haría parte del grupo de estandarización del lenguaje, junto con IBM, Novell y Netscape.

A finales de 1996 el HTML Working Group empezará a trabajar en la nueva versión de HTML, llamada “Cougar”, que será la base del HTML 4. Esta versión incluirá innovaciones para personas con discapacidades, soporte internacional para diferentes lenguajes (no solo aquellos con caracteres latinos), hojas de estilo en cascada, extensiones, scripting y otros.

En 2004, se crea un grupo denominado “WHATWG” con miembros de Apple, la fundación Mozilla, y Opera Software para empezar a desarrollar el estándar HTML5. En 2006 el W3C decide detener su trabajo de XHTML y empieza a trabajar con el “WHATWG” para evolucionar HTML como una tecnología. En 2008 sale a la luz el primer borrador de HTML5, escrito por Ian Hickson, indicando que HTML5 seguiría evolucionando como tecnología continuamente y que nunca estaría completamente terminada

## Complementos del HTML

En 1995 Bert Bos, Hakon Lie, Dave Raggett, Chris Lilley y otros miembros del W3C, se reunían en Versalles, para discutir el desarrollo de un lenguaje especial para escribir estilos en los documentos HTML: las hojas en cascada o CSS (“Cascading Style Sheets”, de aquí en adelante CSS). Este lenguaje CSS describiría el diseño visual de los documentos web, y cómo los elementos HTML deberían ser mostrados y desplegados por el navegador. CSS se diseñó principalmente para marcar la separación del contenido del documento y la forma de presentación de este, es decir, para separar características tales como las capas o layouts, los colores y las fuentes. Esta separación de la “forma” y del “contenido” genera mejor accesibilidad, más flexibilidad y control, y además permite que múltiples páginas (dentro de un mismo sitio Web) compartan el mismo formato y estilo, reduciendo así la complejidad y reutilizando la estructura del contenido. Otra de las principales características de las hojas de estilo en cascada CSS, es que permite especificar una prioridad para determinar qué regla se aplica y en qué secuencia, en caso de que haya algún conflicto de órdenes (de ahí proviene el término “cascada” para explicar la jerarquía de las instrucciones a ejecutar).

En 1996 Dave Raggett propone otro borrador de trabajo, basándose en un borrador inicial de Charlie Kindel y el uso de las extensiones de Netscape para JavaScript,en el cual se aborda el tema del “Scripting”. Este documento se convertiría en parte del estándar HTML y abriría las puertas para los lenguajes de scripting en HTML. Un lenguaje de scripting o script es un pequeño lenguaje de programación cuyo código se inserta dentro del documento HTML y que se ejecuta en el navegador del usuario cuando ocurre un evento, como por ejemplo al cargar la página, o cuando se pulsa sobre un enlace. Estos lenguajes de scripting permiten variar dinámicamente el contenido del documento, modificar el comportamiento normal del navegador, y realizar distintas operaciones como validar formularios, y cambiar aspectos visuales, entre otros, es decir, le proporciona más dinamismo y funcionalidad al lenguaje HTML. El primer lenguaje de script fue el JavaScript de Netscape. Nació con la versión 2.0 de este navegador de la mano de Brendan Eich y basó su sintaxis en Java (un lenguaje de programación para aplicaciones de escritorio). Su popularidad y uso extenso se debió al casi absoluto monopolio que entonces ejercía Netscape en el mercado de navegadores web de aquella época. La idea de Netscape era que este lenguaje atrajera a programadores “amateurs” y que completase al lenguaje Java original.

## Desarrollo Web

Actualmente existe una gran cantidad de lenguajes de programación para realizar desarrollo web, que define la creación de sitios web para Internet o una intranet. Este desarrollo hace uso de tecnologías de software del lado del servidor (Server) y del cliente (Browser) que involucran una combinación de procesos y sistemas de gestión de base de datos con el uso de un navegador web a fin de realizar determinadas tareas o mostrar información. Además, han surgido distintas ocupaciones relacionadas con la construcción de sitios web , como por ejemplo el desarrollador web (web developer), que es quien se preocupa por el funcionamiento del software, el diseñador web (web designer), que es quien se preocupa del aspecto final(layout) de la página y el webmaster, quien es el encargado de integrar ambas partes. En ocasiones el webmaster también se encarga de actualizar los contenidos de la página. De la mano con estas ocupaciones también han aparecido distintos Frameworks, que son un conjunto estandarizado de conceptos, prácticas, criterios, herramientas y entornos de trabajo que facilitan o simplifican el proceso de desarrollo de software y que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. Por si fuera poco, el desarrollo de un sitio web puede separar la capa de presentación de la información o capa de interacción con el usuario (Front End), y la capa de acceso a los datos (Back End), que es la capa que procesa los datos ingresados desde el Front End. Esta separación ayuda a mantener las partes del sistema separadas, simplificando así el mantenimiento, depuración, y ampliación del sitio web. Así las cosas, es claro que construir un sitio web no es una cosa tan simple, pues existen distintas tecnologías, prácticas, y frameworks involucrados, y todos ellos tiene que ver en mayor o menor medida con el lenguaje de marcas HTML.

## Socio Constructivismo

El socio constructivismo y la teoría socio cultural (TS) formulada por el psicólogo ruso Lev Siminovach Vigostky en la década del 40 señala que el individuo necesita de un contexto socio cultural para poder alcanzar un desarrollo adecuado. Indica que todo aprendizaje tiene un origen social y que los individuos necesitan de un ambiente colaborativo que les permita adquirir las capacidades para adaptarse a su entorno. Como lo indica Anama 2020 “*El Socioconstructivismo es una teoría del aprendizaje social, y postula que el sujeto se apropia de su realidad y la dota de significado, siempre y cuando su entorno le provea las herramientas necesarias. Se considera que el aprendizaje es algo colectivo, no algo aislado; esto debido a que siempre es necesario contar con otro para alcanzar el desarrollo, ya que éste cumple una función de guía u orientador en el procedimiento (Anama 2020).*” Hay varios conceptos relacionados con el socio constructivismo como *la zona de desarrollo próximo* (ZDP) considerado como un espacio imaginario entre el conocimiento que ya se posee y el conocimiento que se podría llegar a alcanzar, y que es posible atravesar esta zona de desarrollo próximo con ayuda de otro que le provea el *andamiaje* necesario, entendiéndose *andamiaje* como el proceso que se desarrolla entre el docente y el alumno para que este último alcance la *interiorización*. La *interiorización* es el resultado que obtiene el estudiante después de atravesar la zona de desarrollo próximo gracias al andamiaje recibido. Como cita Arcila 2010 las palabras de Vygotski “*el aprendiz se apodera del conocimiento que estaba tratando de alcanzar, y de esta manera puede ponerlo en práctica para cambiar su entorno (Arcila 2010)”*. Por tanto, la teoría socio constructivista sostiene que los individuos necesitan de un ambiente colaborativo que le ayude a aprender, que le permita desarrollar las capacidades para adaptarse a su entorno, y el lenguaje es el eje central de esta teoría. Según Vygotsky, el lenguaje “*es la herramienta que permitirá al sujeto internalizar el conocimiento, ya que permite la comunicación necesaria para que exista una interacción entre las mentes de los individuos y así la información pueda ser transmitida, internalizada, y apropiada exitosamente (Anama 2020)”.* En palabras de Anton, citado por Anama 2020 *“Se entiende que todo aprendizaje tiene su origen en un entorno social y que el lenguaje capacita a los humanos en el desarrollo de funciones mentales superiores tales como la memoria intencional y la atención voluntaria, la planificación , el aprendizaje y el pensamiento racional (Anama 2020)”.* Por tanto, resulta esencial involucrar el lenguaje, el resultado de utilizar el lenguaje que es la interacción, y el resultado de interactuar grupalmente que es la cultura, para lograr el desarrollo máximo de las capacidades de aprendizaje en un contexto de sociedad, pues el socio constructivismo sostiene que la creación del conocimiento es una experiencia compartida más que una experiencia individual. El lenguaje, la interacción y la cultura posibilitan que surjan nuevos caracteres y rasgos, que implican una relación simbiótica entre el individuo y el contexto. *“El ambiente y la cultura juegan un papel fundamental, siendo centrales en la adquisición y apropiación de significados, ya que cada individuo tendrá formas diferentes de pensamiento y comportamiento dependiendo de las costumbres, leyes y tradiciones del contexto sociocultural en el que se encuentra inmerso, conllevando a que los componentes de la cotidianidad puedan ser interpretados de formas opuestas si se colocan frente a dos culturas (Anama 2020)”.*

## Aprendizaje Experiencial

El aprendizaje experiencial como enfoque o modelo pedagógico indica que el conocimiento se crea a través de las experiencias. En otras palabras, señala que las personas tienen la capacidad de aprender de su propia experiencia, dentro de un marco conceptual y operativo bien estructurado y concreto. Como lo afirma Espinar 2020 “*El aprendizaje experiencial valora las diferencias de cada individuo. A partir de los conocimientos previos de los estudiantes y de la adquisición de nuevos esquemas, que se generan fluidamente como eslabones para unirse en uno desconocido que resulte en aprendizaje significativo, se conduce a la búsqueda y crecimiento del discernimiento por medio de la innovación de la enseñanza-aprendizaje (Espinar 2020)”.* En el aprendizaje experiencial es importante involucrar al estudiante y su contexto en el proceso de aprendizaje, y distinguir sus conocimientos previos, de manera que se puedan producir resultados de aprendizaje y lograr competencias. Como fue citado por Espinar 2020, Samper y Ramírez 2014 indican que “*el aprendizaje experiencial involucra al alumno de manera directa, y plasma los temas nuevos desde su mismo contexto. En este modelo, por ser significativo el proceso de enseñanza, es necesario reconocer la significación de los aprendizajes previos del educando para lograr competencias que puedan desafiar las necesidades actuales. (Espinar 2020)*”. El aprendizaje experiencial se estructura en ciclos que consideran la experiencia como algo de carácter subjetivo y que responden a diversos modelos que permiten entender cómo se desarrolla el proceso de aprendizaje. En la década del 70 el teórico educacional David A. Kolb ayudó a desarrollar la teoría del aprendizaje experiencial, basándose en los trabajos de John Dewey, Kurt Lewin, y Jean Piaget. Por ejemplo, el ciclo de aprendizaje experiencial del Kolb tiene sus bases en la idea de que el aprendizaje experiencial es inductivo, es decir que los participantes llegan a sus propias conclusiones sobre la experiencia, y que tiene cuatro etapas: Experiencia Concreta, Observación Reflexiva, Conceptualización Abstracta y Experimentación Activa. En palabras de Espinar 2020 “*Esta teoría señala las ventajas para percibir y referir el proceso de aprendizaje, fundamentado en la práctica a partir de los sentidos, que son estímulos exteriores que proporcionan experiencias que encaminan el esfuerzo y la experticia de las habilidades (Espinar 2020*)”. En palabras de Rodas Carrillo 2001 como fue citado por Samper y Ramírez 2014, el aprendizaje experiencial es “*una filosofía de educación que parte del principio de que las personas aprenden mejor cuando entran en contacto directo con sus propias experiencias y vivencias, es un aprendizaje haciendo, que reflexiona sobre el mismo hacer. La educación experiencial sucede cuando las personas se ven inmersas dentro de las actividades, reflexionan sobre lo vivido en dichas actividades, y realizan un análisis del cual obtienen conclusiones que podrán aplicar de manera autónoma (Samper y Ramírez 2014)”.*

## Aprendizaje Basado en Problemas ABP

El aprendizaje basado en problemas es un método pedagógico de enseñanza-aprendizaje centrado en el estudiante que señala que el aprendiz adquiere conocimientos y habilidades a través de situaciones de la vida real. La finalidad es formar individuos con capacidad analítica que puedan enfrentarse a problemas de la misma manera en que lo harían durante su actividad profesional. El ABP utiliza problemas como punto de partida para la adquisición de conocimientos nuevos y es el estudiante el protagonista de su propio proceso de aprendizaje. Se fundamenta en la base constructivista de que conocer y aprender implican una experiencia de construcción anterior. El principal rol del docente es el de facilitador de actividades de reflexión que ayuda a pensar críticamente los temas y un catalizador de la investigación y el descubrimiento. En la década del 70, el médico y educador Howard Barrows Barrows propuso que se les debería permitir a los estudiantes involucrarse con nueva información en el contexto de la resolución de problemas clínicos auténticos. Los postulados de Barrows establecieron las bases de lo que conocemos como el aprendizaje basado en problemas, y tal y como lo indican Barrows y Tamblyn 1980 “*en el caso del ABP se presenta primeramente el problema, se identifican las necesidades de aprendizaje, se busca la información necesaria y finalmente se regresa al problema a fin de resolverlo (Barrows y Tamblyn, 1980)”.* El ABP busca que el estudiante interiorice la solución del problema y que explore distintos campos de manera integral, pues como lo afirma Poot-Delgado 2013 “*El ABP incluye el desarrollo del pensamiento crítico en el mismo proceso de enseñanza-aprendizaje; no lo incorpora como algo adicional, sino que es parte del mismo proceso de interacción para aprender. Busca que el alumno comprenda y profundice adecuadamente en la respuesta a los problemas que se usan para aprender, abordando aspectos de orden filosófico, sociológico, psicológico, histórico, práctico y demás, haciendo todo lo anterior con un enfoque integral.( Poot-Delgado 2013)”.*

## Aprendizaje Cooperativo

El aprendizaje cooperativo es un método pedagógico que propicia la enseñanza a través de la interacción y socialización de los estudiantes. El proceso consiste en dividir el alumnado en grupos pequeños que trabajen entre sí de forma coordinada resolviendo tareas o problemas, poniéndose de acuerdo sobre las tareas que se deben realizar y decidir cómo y quién realizará las diferentes actividades. Como lo señala Domingo 2008 ***“****El Aprendizaje Cooperativo permite que los estudiantes trabajen de forma independiente y asuman responsabilidades en su propio proceso de aprendizaje. También promueve el desarrollo de la capacidad para razonar de forma crítica y facilita el desarrollo de la habilidad para escribir con claridad. (Domingo 2008)”* Es así que tanto la teoría de aprendizaje subyacente del socio constructivismo, como el modelo o enfoque pedagógico del aprendizaje experiencial, como el método pedagógico del aprendizaje basado en problemas tienen sus bases en el constructivismo. Y el aprendizaje cooperativo no es la excepción, pues como lo afirma Ovejero 1990 citado por García 2001 *“El constructivismo, al estudiar la interacción social y la construcción del conocimiento a través de los procesos interactivos, también puede considerarse como antecedente del aprendizaje cooperativo. Téngase en cuenta que los fundamentos de este van a estar directamente relacionados con la interacción social y la construcción social de la inteligencia. (García 2001)”.* Por tanto, en el aprendizaje colaborativo se requiere que todos los integrantes del grupo alcancen sus objetivos. No existe el individualismo, pues si un integrante alcanza sus propios objetivos, pero algún integrante del grupo no logra los suyos, no se considera una situación de aprendizaje colaborativo. Quizás exista el concepto de grupo, pero no es aplicable el concepto de cooperativismo. Como lo afirma García 2001: “*Una situación de aprendizaje cooperativo es aquella en la que los estudiantes pueden conseguir sus objetivos sí y solo si los demás con los que trabaja cooperativamente consiguen también los suyos (García 2001)”.* Como lo indica García 2001, existen tres procedimientos que los alumnos pueden seguir para alcanzar sus objetivos de aprendizaje y son: cooperativo, competitivo e individual. Como se mencionó anteriormente, la estructura individualista es contraria a la estructura cooperativa, que además supone una interdependencia positiva entre los individuos. “*La interdependencia positiva significa que todos los miembros de un grupo cooperativo están interesados por el máximo aprendizaje de cada uno de sus compañeros (García 2001)*”. Finalmente, se puede decir que en el aprendizaje cooperativo es más valioso el proceso por el que pasan los alumnos que el resultado o un producto, pues se pretende que los estudiantes interactúen entre sí para lograr un objetivo común, con el propósito de lograr la interdependencia positiva mencionada anteriormente. Como lo menciona Ovejero 1990 citado por García 2001 “*Todo aprendizaje cooperativo es aprendizaje en grupo, pero no todo aprendizaje en grupo es aprendizaje cooperativo. No es la mera cantidad de interacción entre los alumnos lo que provoca unos resultados beneficiosos, sino la naturaleza de tal interacción, como ya hemos reiterado numerosas veces (García 2001)”.*

# Descripción de la Experiencia:

El plan de trabajo que se ha venido desarrollando desde 2019, y que fue presentado en la versión de proyecto definitivo se presenta a continuación. Es importante mencionar que el proyecto se detuvo en la mitad del proceso, cuando se estaba empezando a construir el simulador y se habían definido algunas teorías y modelos de aprendizaje.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Actividades** | **Semanas** | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Investigación de modelos, interfaces, y lenguajes de programación aptos para desarrollar el simulador para el aprendizaje del lenguaje de marcas HTML | X | X | X | X | X | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Investigación de la forma en que serán presentados los conceptos en el simulador (figuras, formas, colores, animación, ejemplos, situaciones, organización de conceptos) |  |  |  |  |  |  | X | X | X | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Formato de presentación del sistema drag and drop para el aprendizaje del lenguaje de marcas HTML |  |  |  |  |  |  |  |  |  |  | X | X | X |  |  |  |  |  |  |  |  |  |  |  |  | |
| Investigación de las teorías de aprendizaje subyacentes, modelos o enfoques pedagógicos, y métodos pedagógicos a utilizar junto con el simulador | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |  |  |  |  |  |  |  |  |  |  | |
| Formato de presentación del sistema de edición-visualización simultáneo para el aprendizaje del lenguaje de marcas HTML |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X |  |  |  |  |  |  |  |  |  | |
| Desarrollo del simulador de acuerdo con el modelo, lenguaje de programación, interfaces y formato de presentación seleccionados |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | X | X | X | X | X | X | X | X | X | |
| Implementación del Simulador en página web para probar flexibilidad y portabilidad |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | |
| Diseño de actividades relacionadas con el simulador y las teorías, modelos y enfoques pedagógicos seleccionados |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | X | |
| Pruebas de desempeño del simulador implementado como herramienta de aprendizaje |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |

Tabla 1 Diagrama de Gantt de las Actividades a Desarrollar

Es importante aclarar que las etapas realmente tomaron mucho más tiempo de lo planteado originalmente, porque se desconocía la magnitud real de lo que se quería construir, y se partió de un supuesto que quería materializarse.

## Investigación de modelos, interfaces, y lenguajes de programación aptos para desarrollar el simulador para el aprendizaje del lenguaje de marcas HTML

Como se puede observar, la primera parte que involucraba 6 semanas se denomina “Investigación de modelos, interfaces, y lenguajes de programación aptos para desarrollar el simulador para el aprendizaje del lenguaje de marcas HTML”. En este caso, se analizaron distintos lenguajes de programación para construir el simulador/entrenador, como JAVA, Python, y C, pero se eligió finalmente Javascript por la flexibilidad que ofrecía, y la facilidad de desplegar el producto en cualquier sitio Web. A continuación se relaciona una tabla con los lenguajes de programación investigados y sus principales características, calificadas en una escala subjetiva del 1 al 10, de acuerdo con mi experiencia como desarrollador de software. Se tomaron como criterios de evaluación tres características clave para desarrollar el producto de software: dificultad para codificar el producto, dificultad para desplegar el producto y dificultad para integrar el producto con otros servicios o plataformas

|  |  |  |  |
| --- | --- | --- | --- |
| **Lenguaje** | **Dificultad para Codificar el producto** | **Dificultad para Desplegar el Producto en un ambiente de producción** | **Dificultad para integrar el producto con otros servicios o plataformas** |
| JAVA | 8  El lenguaje de programación compilado JAVA requiere el conocimiento de la Programación Orientada a Objetos para producir resultados óptimos, lo que incrementa su nivel de dificultad | 8  JAVA al ser un lenguaje compilado y orientado a aplicaciones Desktop, es necesario ubicar el resultado de la compilación en la máquina que va a ejecutar el producto, aunque tiene la ventaja de ser multiplataforma lo que le permite desplegar el mismo bytecode en distintos sistemas operativos. Entonces, si se va a desplegar el simulador en un aula de sistemas de 30 computadores, sería necesario desplegar 30 compilaciones en las 30 máquinas anteriormente mencionadas. | 10  JAVA al ser un lenguaje compilado y orientado a aplicaciones Desktop, requiere de conocimientos en XML, SOAP o SOCKETS para intercambiar información con otros sistemas, lo que aumenta la dificultad de interoperabilidad. |
| C | 6  El lenguaje de programación compilado C admite el paradigma de programación orientada a objetos, así como el de programación modular y el paradigma de programación estructurada, de manera que hay múltiples opciones para realizar el desarrollo | 9  C al ser un lenguaje compilado y orientado a aplicaciones Desktop, es necesario ubicar el resultado de la compilación en la máquina que va a ejecutar el producto al igual que JAVA, con la desventaja que no es multiplataforma, lo que implica conocer el sistema operativo de destino y que sea el mismo para todas las máquinas. Entonces, si se va a desplegar el simulador en un aula de sistemas de 30 computadores, sería necesario desplegar 30 compilaciones en las 30 máquinas anteriormente mencionadas. | 10  C al ser un lenguaje compilado y orientado a aplicaciones Desktop, requiere de conocimientos en XML, SOAP o SOCKETS para intercambiar información con otros sistemas, lo que aumenta la dificultad de interoperabilidad. |
| Python | 8  El lenguaje de programación interpretado Python admite el paradigma de programación orientada a objetos, así como el de programación modular y el paradigma de programación estructurada, de manera que hay múltiples opciones para realizar el desarrollo, sin embargo, posee la dificultad de la indentación. Si no se presta atención a la indentación, se podrían generar errores en tiempo de ejecución al alinear bloques de código erróneamente. | 7  Python al ser un lenguaje interpretado y multiplataforma, facilita un poco el despliegue del producto en un ambiente de producción. Sin embargo, hay que definir si el producto se va a construir es un proyecto desktop o un proyecto web, para determinar los pasos a seguir en el despliegue, sin embargo, su dificultad para desplegar el producto en un ambiente de producción es menor comparado con C o JAVA. | 3  Python al ser un lenguaje interpretado y con soporte a JSON facilita el intercambio de información en este formato, lo que disminuye la dificultad para integrar el producto con otros servicios o plataformas. |
| Javascript | 2  El lenguaje de programación interpretado Javascript admite el paradigma de programación orientada a objetos, así como el de programación modular y el paradigma de programación estructurada, de manera que hay múltiples opciones para realizar el desarrollo. Por otro lado, al ser un lenguaje de programación interpretado, facilita la ejecución del programa a pesar de tener fallos, pues si no se alcanza la línea de fallo, nunca se detendrá la ejecución del sistema. Eso sumado a la flexibilidad que otorga el lenguaje a la hora de escribir sentencias e interpretarlas, lo hace un firme candidato para desarrollar el simulador. | 2  El lenguaje de programación interpretado Javascript solamente necesita desplegar los archivos en el servidor web, y al ser un modelo de comunicación Cliente-Servidor, no es necesario realizar configuración adicional en las máquinas clientes. Solamente se despliegan los archivos en el servidor, y los demás máquinas clientes podrá utilizar el servicio desde cualquier parte del mundo con acceso a internet. | 2  JavaScript al ser un lenguaje interpretado y con soporte a JSON – De hecho es gracias a JavaScript que existe el nombre JSON, a saber: JavaScript Object Notation, facilita el intercambio de información en este formato, lo que disminuye la dificultad para integrar el producto con otros servicios o plataformas. |

Tabla 2. Lenguajes de Programación Considerados

Como se puede observar en la tabla 2, el lenguaje de programación que más se acopla a las necesidades de desarrollo del simulador es JavaScript. Es importante anotar que esta ponderación es meramente subjetiva y acorde con la experiencia mía como desarrollador de software en los distintos lenguajes de programación presentados. Es posible que dicha tabla sea diferente vista desde la posición de otro desarrollador de software.

## Lenguaje de programación a utilizar

En esta fase se determinó que el lenguaje de programación que se utilizará para realizar el desarrollo será JAVASCRIPT, pues proporcionará dinamismo y una fácil integración con el lenguaje de marcas HTML. Gracias a los resultados de la tabla 2 se pudo sustentar que la elección del lenguaje de programación más adecuado para la situación sería JAVASCRIPT.Además, al utilizarse JAVASCRIPT nativo, será más simple la integración de la solución con cualquier sitio web o plataforma LMS, sin necesidad de instalar lenguajes de programación adicionales o frameworks. JAVASCRIPT es fácilmente interpretado por casi todos los navegadores web, lo que redunda en que la solución podrá ser visualizada en cualquier sistema, sin importar su plataforma o sistema operativo (Windows, Linux, Macintosh, Unix, Android, etc). Es importante mencionar que casi todos los sitios web utilizan JavaScript por ser un lenguaje que ya está integrado en los navegadores (no hay que instalar nada) y este lenguaje es uno de los más solicitados en los avisos de búsqueda de empleos para programadores. A continuación se muestran algunas de las ventajas más destacadas de JavaScript:

* Es un lenguaje muy sencillo de aprender con respecto a otros lenguajes de programación web (php, python, asp)
* Es rápido (interpretado), por lo tanto tiende a ejecutar el código casi que inmediatamente.
* Es soportado por los navegadores más populares y es compatible con los dispositivos más modernos, incluyendo tablets, móviles y consolas de videojuegos
* Se puede utilizar para la validación de datos en un formulario del lado del cliente.
* Es multiplataforma, puede ser ejecutado de manera híbrida en cualquier sistema operativo ya sea de escritorio (Windows, Linux, Macintosh) o móvil (Android , IOS).
* Es el único lenguaje que permite trabajar modo FullStack en cualquier tipo de desarrollo de programación (JS, NodeJS, VueJS).

Como se puede observar, JAVASCRIPT tiene varias ventajas que lo hacen el candidato idóneo para desarrollar el simulador que facilite la enseñanza virtual del lenguaje de marcas HTML, haciendo énfasis en los formularios.

## Investigación de la forma en que serán presentados los conceptos en el simulador (figuras, formas, colores, animación, ejemplos, situaciones, organización de conceptos)

En esta segunda parte que inicialmente involucraba 4 semanas, se definió la forma en que el simulador iba a trabajar, haciendo pruebas de código, y validando la elección del lenguaje de programación utilizado. Realmente el tiempo para esta fase tomó más de cuatro semanas (casi el doble de tiempo), pues se iban probando distintas configuraciones, y distintas estrategias. Por ejemplo, en principio se determinó utilizar JQUERY para manejar los eventos y el modelo basado en funciones planteado, pero luego se optó por utilizar javascript puro para manipular los elementos del DOM con instrucciones del tipo document.getElementById()

### Modelo

En principio, y mientras se hacia la investigación, se determinó que el modelo para desarrollar el software será el patrón de arquitectura MVC, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. El MVC define la construcción de tres componentes distintos que son el Modelo, la Vista y el Controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en la noción de reutilización de código y la separación de conceptos, buscando así facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

El Modelo es la representación de la información con la cual el sistema opera, por lo tanto gestiona los accesos a dicha información tales como consultas, actualizaciones, entre otros. También implementa los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' la información que en cada momento se le solicita para que sea mostrada. Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

El Controlador responde a eventos o acciones del usuario y realiza peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos, o mostrar alguna información). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

La Vista presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por tanto requiere de dicho 'modelo' la información que debe representar como salida.

Luego cuando se empezó la construcción del Entrenador/Simulador, se evidenció que era más práctico un modelo basado en funciones. Es decir, el paradigma de programación funcional es adecuado para el desarrollo del simulador, pues se identificó que era necesario implementar muchas funciones que realizaran distintas tareas (como la funciones de callback que responden al evento de un botón ), y sería más engorroso apoyarse en una arquitectura de patrón MVC, por tanto, el modelo utilizado para el desarrollo del simulador es la programación funcional orientada a eventos. Se utilizarán los eventos dragstart para permitir arrastrar los elementos, dragover y drop para permitir soltar los elementos sobre una sección particular, y touchstart para la situación especial de visualizar el simulador/entrenador en dispositivos móviles y permitir la ubicación de los objetos tocándolos.

## Formato de presentación del sistema drag and drop para el aprendizaje del lenguaje de marcas HTML

### Interfaces

En principio, la interfaz gráfica que se utilizará consiste en una ventana dividida inicialmente en tres columnas. La primera de ellas poseerá los controles y elementos HTML que responderán al sistema drag and drop con respecto a la segunda columna. Es decir, se pretende que en la primera columna se encuentren los componentes arrastrables, y en la segunda columna se puedan soltar, de manera que en la tercera columna se pueda visualizar el código html asociado.

|  |  |  |
| --- | --- | --- |
| COMPONENTES HTML  Estos componentes permitirán ser arrastrados, de manera que sean soltados en el área de trabajo | AREA DE TRABAJO  En el área de trabajo se sueltan los componentes y se organizan visualmente de acuerdo a los requerimientos del usuario | AREA DE CÓDIGO  En el área de código se visualizará el código HTML generado, de acuerdo a la organización y componentes proporcionada en el área de trabajo. |

Tabla 3. Interfaz Gráfica Propuesta

Esta interfaz además se actualizará en tiempo real cada vez que se suelte un componente en el área de trabajo, de manera que el usuario no tenga que refrescar el navegador para visualizar los cambios. Es decir, una de las ventajas del simulador es que el estudiante vaya observando en tiempo real el resultado de manipular los componentes en el área de trabajo. Así mismo, se tiene pensado que si el estudiante tiene experiencia utilizando el lenguaje de programación, utilice el área de código para que modifique los componentes a través instrucciones, y pueda visualizar los resultados de la manipulación en el área de trabajo. En otras palabras, el área de trabajo y el área de código serán interdependientes.

Luego de comenzado el desarrollo, se determinó una opción diferente para presentar la información de la siguiente manera:

|  |  |
| --- | --- |
| COMPONENTES HTML  Estos componentes permitirán ser arrastrados, de manera que sean soltados en el área de trabajo | |
| AREA DE TRABAJO  En el área de trabajo se sueltan los componentes y se organizan visualmente de acuerdo a los requerimientos del usuario | AREA DE CÓDIGO  En el área de código se visualizará el código HTML generado, de acuerdo a la organización y componentes |

Tabla 4. Nueva Interfaz Gráfica Propuesta

En esta nueva interfaz, hay más espacio para el área de trabajo y para el área de código, en comparación con lo presentado en la tabla 3.

## Investigación de las teorías de aprendizaje subyacentes, modelos o enfoques pedagógicos, y métodos pedagógicos a utilizar junto con el simulador

Con todas las asignaturas cursadas en la maestría en E-Learning, se encuentra que el constructivismo es un candidato fuerte para sustentar la propuesta de enseñanza virtual mediante el simulador/entrenador, pues el constructivismo propende por entregar “andamiajes” a los estudiantes que les permitan construir su propio conocimiento y sus propias estrategias para resolver problemas. Sin embargo, pensando en la problemática de desarrollar software de manera colaborativa, se profundiza en la teoría de aprendizaje del socio-constructivismo, que como se mencionó en el apartado de estado del arte  *“postula que el sujeto se apropia de su realidad y la dota de significado, siempre y cuando su entorno le provea las herramientas necesarias. Se considera que el aprendizaje es algo colectivo, no algo aislado; esto debido a que siempre es necesario contar con otro para alcanzar el desarrollo, ya que éste cumple una función de guía u orientador en el procedimiento (Anama 2020)”.* Por tanto, en esta propuesta se hace hincapié en la necesidad de trabajar con el otro en el proceso de desarrollo de software, para alcanzar un conocimiento pleno y fructífero, y que el conocimiento se arraigue en el interior de cada sujeto. En palabras de Arraya citado por Anama et al. (2020) *“… la corriente constructivista (…) propone el desarrollo máximo (…) de las capacidades (…) del aprendiz. El propósito se cumple cuando se considera al aprendizaje en el contexto de una sociedad, impulsado por un colectivo (…) en la formación de las nuevas generaciones (…) los constructivistas sociales insisten en que la creación del conocimiento es más bien una experiencia compartida que individual (Anama 2020)”.* Entonces, en el socio constructivismo el docente proporciona el andamiaje al estudiante o aprendiz como una “*estructura de apoyo que el tutor experto ofrece al discípulo novato durante la interacción, con el fin de completar la tarea con éxito. El andamiaje se concreta en varias funciones de los enunciados del experto como pueden ser llamar la atención sobre los elementos importantes de la actividad o tarea, ofrecer un modelo o simplificar la tarea, controlar el grado de frustración, etc (Anama 2020)”.* Después de haber elegido el socio-constructivismo como teoría de aprendizaje por las razones anteriormente expuestas, se procede a analizar el aprendizaje experiencial como enfoque o modelo pedagógico. Con esta propuesta de enseñanza que involucra un entrenador/simulador se busca que el estudiante construya su propio conocimiento, y que se apoye en lo que ya sabe para que descubra el saber mediante su propia experiencia. Es así que la combinación de aprender de lo que ya se sabe y aprender de lo nuevo por descubrir, se convierte en una excelente fórmula para que los estudiantes asimilen el conocimiento mediante el uso de un simulador/entrenador. Como lo afirma Espinar 2020 *“El aprendizaje previo brinda una oportunidad magnífica para vincular lo teórico práctico en espacios reales; lo convierte en conocimientos sólidos, perpetuo, en situaciones verdaderas que se consolidado en un conocimiento significativo, contextualizado, transmisible y beneficioso para la vida. La enseñanza que parte de la experiencia busca la resolución mediante las cuatro fases del ciclo de aprendizaje, que se origina de una incógnita y se solucionará mediante el razonamiento entre mediador y alumno. (Espinar 2020)”.*  Por tanto, con el uso de este simulador/entrenador se quiere que los estudiantes aprendan mediante la experiencia de uso del mismo, y que pueda utilizar los conocimientos que ya posee y que va construyendo, para consolidar el conocimiento relacionado con el lenguaje de marcas HTML. En otras palabras, se quiere acrecentar el potencial de los estudiantes a través de la profundización en su experiencia, mediando cuatro elementos: experiencia real, introspección, conformación de concepciones imprecisas, y ensayo de nuevos esquemas. Se plantearán preguntas y actividades que trabajarán de la mano con el simulador para que el estudiante construya el nuevo conocimiento mediante una experiencia real, que solucione sus propias inquietudes y concepciones imprecisas, y pruebe nuevas ideas y nuevos esquemas que le permitan interiorizar el lenguaje de marcas HTML. En palabras de Espinar 2020 acerca del uso de aprendizaje experiencial : “*es un aprendizaje operativo que se manifiesta por mediación de cuatro elementos: experiencia real, que utiliza como sustento para el reconocimiento; introspección sobre la experiencia real y el reconocimiento para originar un supuesto de la nueva información; conformación de concepciones imprecisas y nociones básicas con razonamientos concretos asentados sobre conjeturas anteriores, y el ensayo de los nuevos esquemas aplicados a nuevas experiencias que permitan solucionar problemas en la vida real (Espinar 2020)”.* De la mano con el aprendizaje experiencial, y soportando al socio constructivismo podemos mencionar el cooperativismo o aprendizaje cooperativo como método pedagógico que propicia la enseñanza, que como lo afirma García 2020, es una situación de aprendizaje “*en la que los estudiantes pueden conseguir sus objetivos sí y solo si los demás con los que trabaja cooperativamente consiguen también los suyos (García 2001)”.* Finalmente, y como soporte fundamental para todo el modelo pedagógico de la propuesta, se usará el Aprendizaje Basado en Problemas ABP como método pedagógico que articulará el trabajo con el Simulador/Entrenador y los resultados de aprendizaje que espera el docente desarrollar en sus alumnos. Se plantearán problemáticas que se puedan solucionar mediante el uso del simulador/entrenador, de manera que incentive el análisis, la reflexión y el pensamiento crítico de la situación. No se trata de proporcionar una herramienta y dejar que los estudiantes la utilicen sin una guía, sino que se proporcionan una serie de preguntas y problemas que harán reflexionar al estudiante sobre su proceso de aprendizaje. Por tanto, el aprendizaje basado en problemas calza perfectamente en una situación de aprendizaje que se apoye en el uso de una herramienta o simulador. En palabras de Poot-Delgado (2013) “*el aprendizaje basado en problemas incluye el desarrollo del pensamiento crítico en el mismo proceso de enseñanza-aprendizaje; no lo incorpora como algo adicional, sino que es parte del mismo proceso de interacción para aprender. Busca que el alumno comprenda y profundice adecuadamente en la respuesta a los problemas que usan para aprender, abordando aspectos de orden filosófico, sociológico, psicológico, histórico, práctico y demás, haciendo todo lo anterior con un enfoque integral Poot-Delgado 2013)”.* De esta manera, se pretende por un lado construir un simulador/entrenador que facilite el proceso de enseñanza-aprendizaje del lenguaje de marcas HTML y por otro lado, proponer el uso de un conjunto de teorías de aprendizaje, enfoques y métodos pedagógicos que resulten efectivos en el uso del simulador/entrenador desarrollado. En el apartado denominado “Diseño de actividades relacionadas con el simulador y las teorías, modelos y enfoques pedagógicos seleccionados” de este documento, se ilustran las actividades creadas con base en la teoría de aprendizaje, enfoques y métodos seleccionados.

## Formato de presentación del sistema de edición-visualización simultáneo para el aprendizaje del lenguaje de marcas HTML

### Prototipo Obtener código HTML

En primer lugar, se investigó la forma en que se podría obtener tanto el contenido como el código de un objeto HTML presionando un botón. La idea era que se pudiese comprobar que era posible extraer el contenido HTML de cualquier objeto y que este pudiese ser visualizado en pantalla. El código presentado a continuación utiliza JAVASCRIPT, específicamente su librería JQUERY, para realizar realiza dicha tarea:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#btn1").click(function(){

alert($("#texto").text());

});

$("#btn2").click(function(){

alert($("#texto").html());

});

});

</script>

</head>

<body>

<div id="texto"><input type="text" placeholder="prueba">Este texto <b>esta en negrita</b></div>

<button id="btn1">Solo Texto</button>

<button id="btn2">HTML</button>

</body>

</html>

Código 1. Página web simple con una caja de texto y dos botones

El resultado de ejecutar dicho código en un navegador web cualquiera es el siguiente:

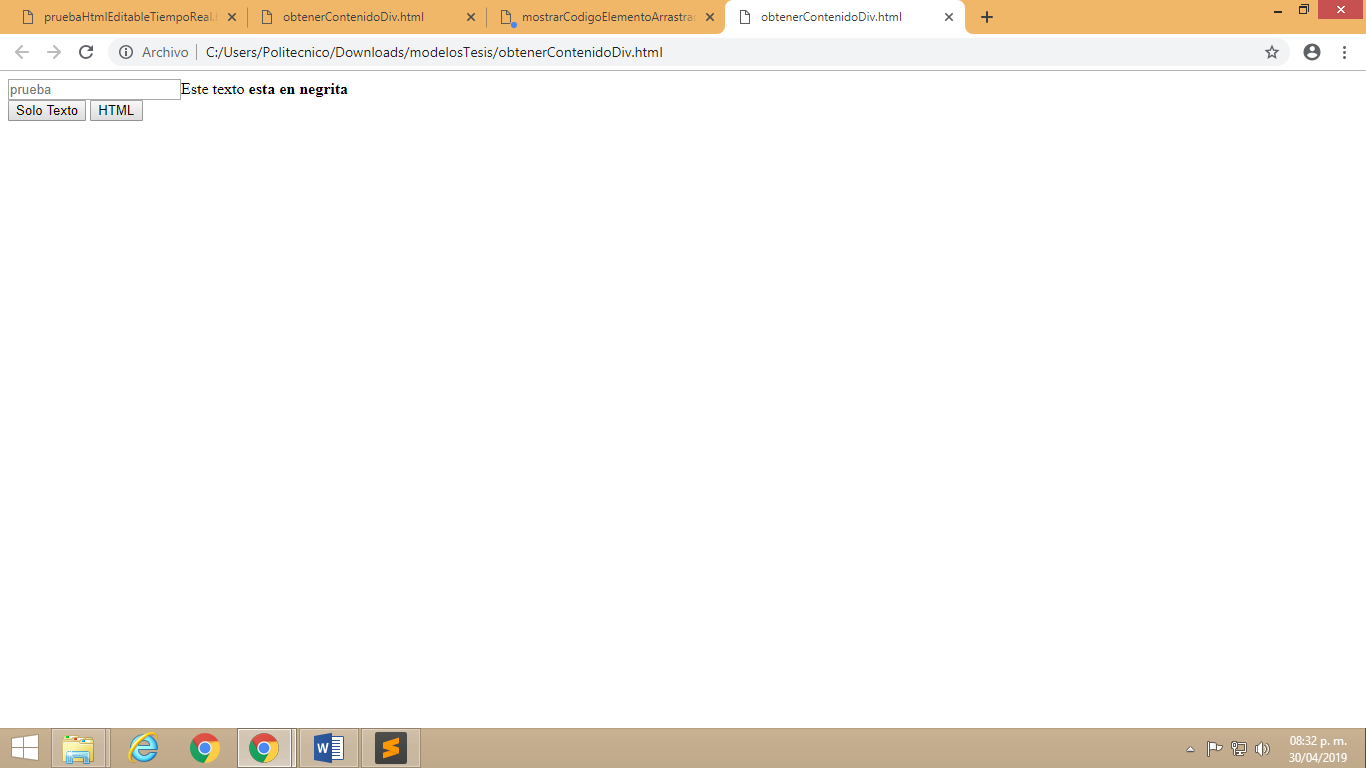


Ilustración 1. Prueba de código para visualización de HTML

Es decir, se observa una caja de texto con la palabra “prueba” , y seguidamente una frase que dice “Este texto está en negrita”. Debajo de ellos se encuentra un botón que dice “Solo Texto” y otro que dice “HTML”. Lo interesante del asunto es que la caja de texto que dice “prueba” y la frase que dice “Este texto está en negrita” se encuentran dentro de un contenedor HTML denominado “<div>”. Lo que se quiere hacer con el código enmarcado dentro de las etiquetas <script> </script> es que cuando se presiona el botón denominado “Solo Texto”, aparezca en pantalla un mensaje con el contenido del elemento <div>, en este caso la frase “Este texto está en negrita”, y que cuando se presione el botón denominado “HTML”, aparezca en pantalla un mensaje con el código HTML que se encuentre dentro del elemento <div>, en este caso, los siguientes comandos HTML : <input type="text" placeholder="prueba">Este texto <b>esta en negrita</b>.

En la siguiente imagen se puede visualizar el resultado de presiona el botón “Solo Texto”,

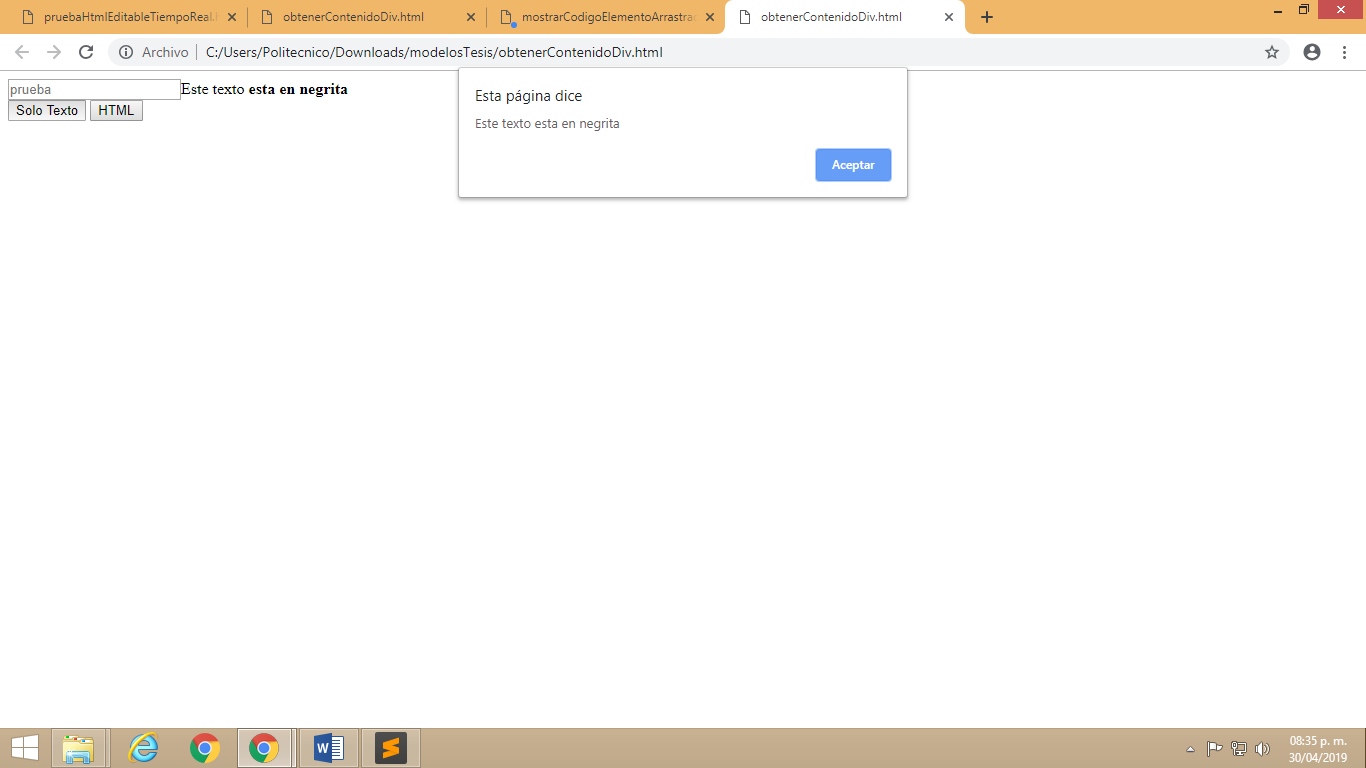


Ilustración 2. Resultado de presionar el botón “Solo Texto”

En la siguiente imagen se puede visualizar el resultado de presionar el botón “HTML”.

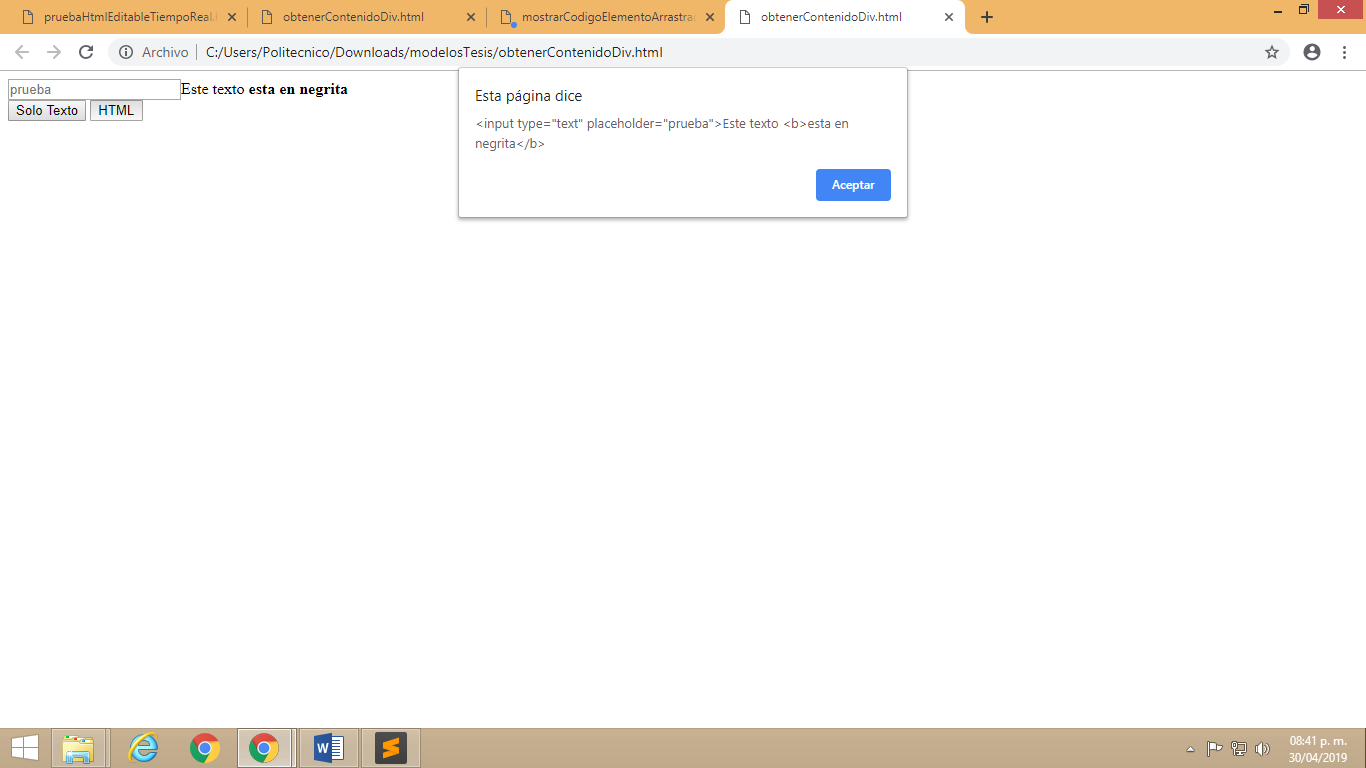


Ilustración 3. Resultado de presionar el botón “HTML”

Como se puede observar en la ilustración 3, es posible utilizar un lenguaje de programación como JAVASCRIPT mediante su librería JQUERY para obtener tanto el código como el contenido de un objeto HTML (en este caso el contenedor <div>), y por tanto, se puede hacer extensible este conocimiento para empezar a desarrollar el simulador, específicamente la sección de “Área de Trabajo” y “Área de Código” descritas en la Tabla 1.

### Prototipo HTML editable en tiempo real

Una segunda prueba que se realizó fue crear un contenedor que permitiera el ingreso de código HTML y que en tiempo real fuese ilustrando la materialización de dicho código en el navegador, ya sea para crear un botón, una caja de texto, o cualquier componente HTML. La idea era que se pudiese comprobar que era posible escribir código HTML en un contenedor, y poder visualizar en tiempo real el componente o elemento que se estuviese describiendo en dichas instrucciones. El código presentado a continuación utiliza JAVASCRIPT, específicamente su librería JQUERY, para realizar realiza dicha tarea:

<html>

<head></head>

<body>

<table>

<tr>

<td id="real"></td>

</tr>

</table>

<textarea id="txt"></textarea>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$('#txt').keyup(function(){

$('#real').html($(this).val());

});

});

</script>

</body>

</html>

Código 2. Página web simple con contenedor editable de HTML

El resultado de ejecutar dicho código en un navegador web cualquiera es el siguiente:

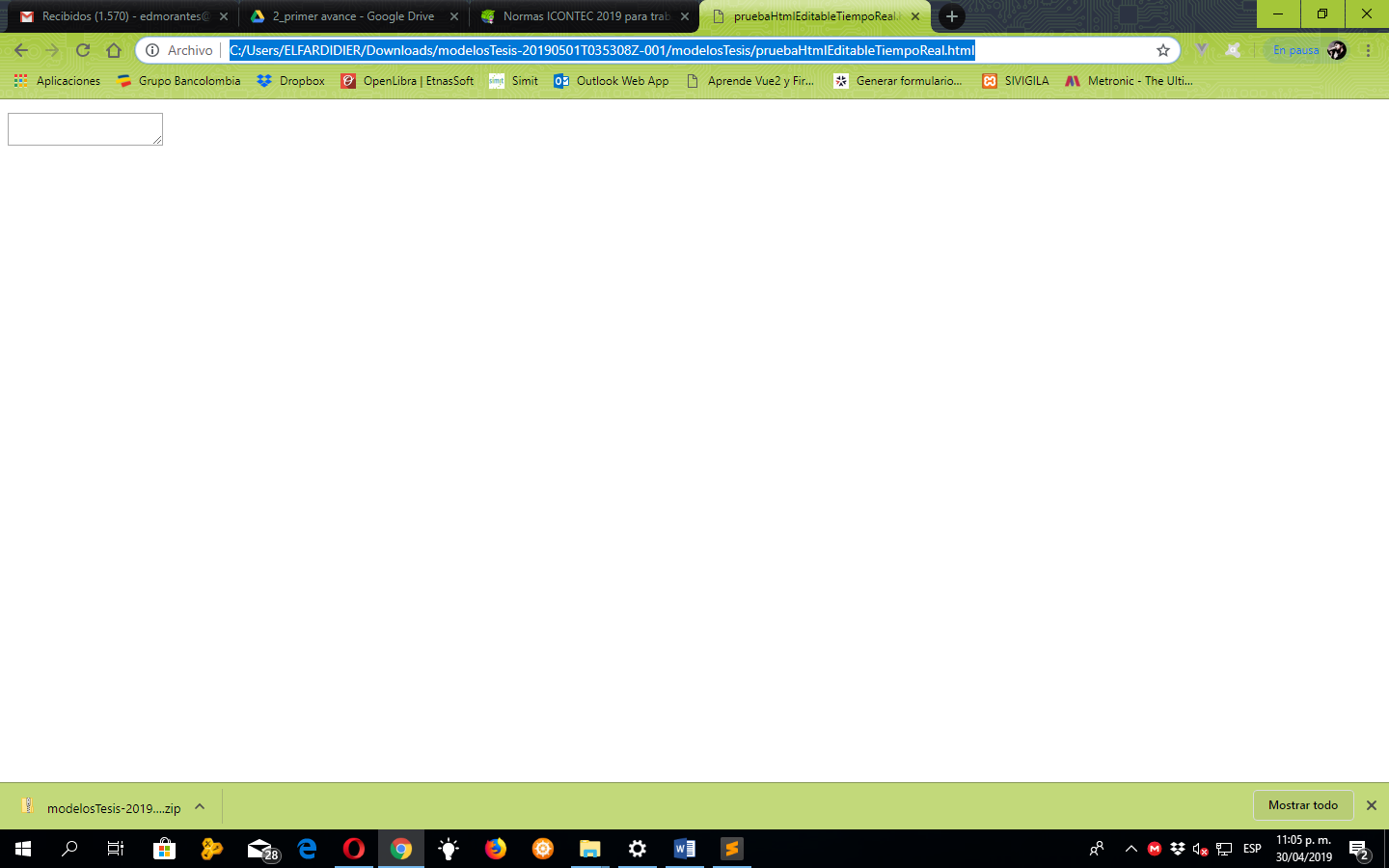


Ilustración 4. Prueba de código para visualización de HTML en tiempo real

Como se puede observar en la ilustración 4, solamente se visualiza un contenedor con la propiedad de poder escribirse en él. Evidentemente como no se ha escrito nada, no se puede visualizar su utilidad. Pero si se empiezan a escribir sentencias HTML, dicho contenedor comenzará a responder, y por tanto, se podrán visualizar los componentes HTML que se describan en el contenedor. Por ejemplo, si se escribe el código correspondiente a la creación de un botón en lenguaje HTML dentro de este contenedor, podremos ver dicho botón en nuestra web tras finalizar la escritura del último carácter correspondiente a la sentencia: <input type= “button” value= “enviar”>, como se muestra en la ilustración 5.

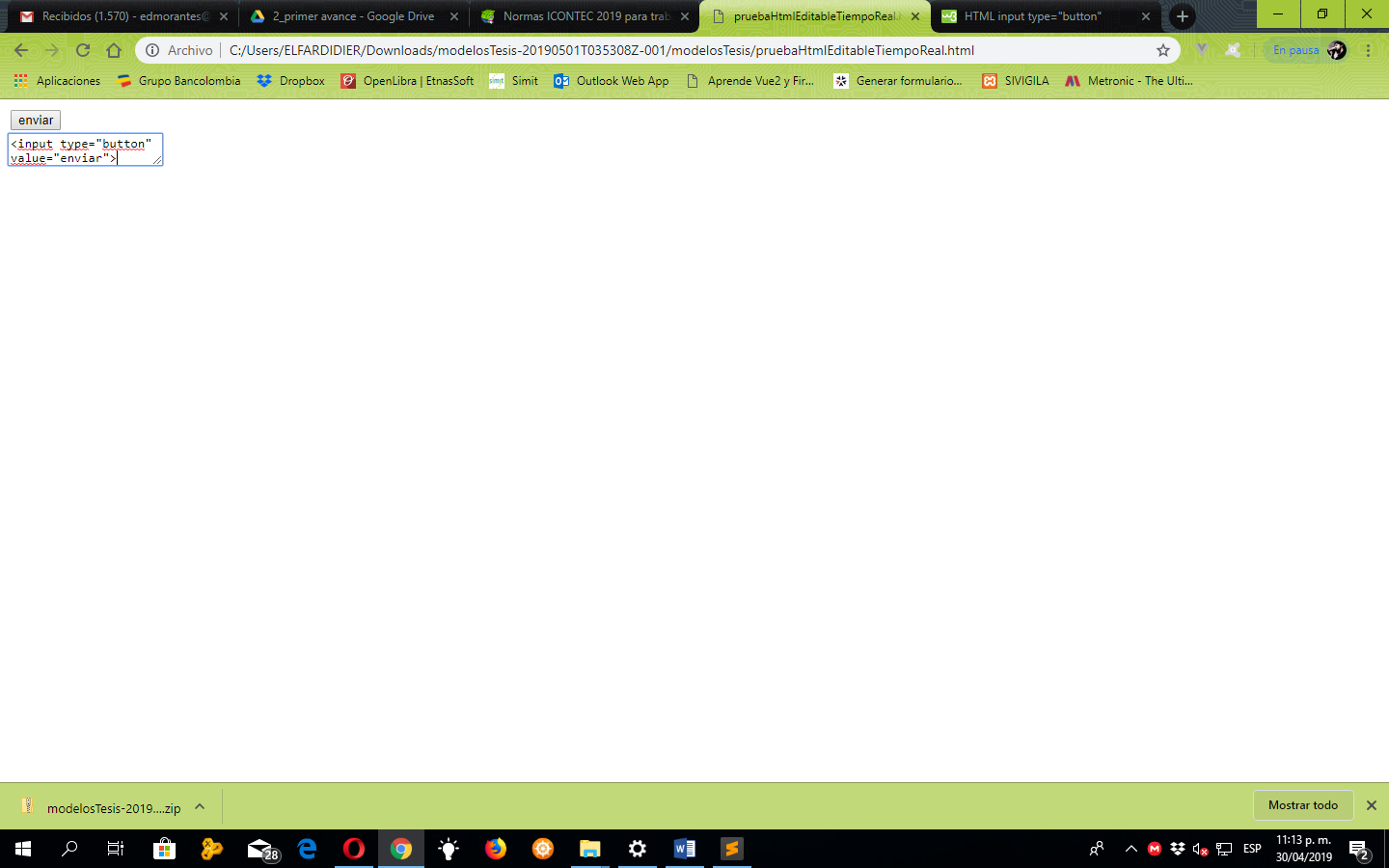


Ilustración 5. Resultado de escribir código HTML en el contenedor editable

Es de anotar que si no se escriben correctamente las sentencias HTML dentro del contenedor, no se podrá visualizar nada, ya que dicho contenedor se ciñe fuertemente a la sintaxis del lenguaje de marcas HTML. Como se puede observar en la ilustración 5, es posible utilizar un lenguaje de programación como JAVASCRIPT mediante su librería JQUERY para generar un contenedor reactivo que presente elementos HTML en la página web en la medida en que se vaya escribiendo correctamente el código, y por tanto, se puede hacer extensible este conocimiento para empezar a desarrollar el simulador, específicamente la sección de “Área de Código” descrita en la Tabla 1.

### Prototipo de Sistema Drag and Drop para el Simulador

Una tercera prueba realizada fue crear un prototipo que plasmara la idea de la Tabla 1, de tener tres secciones diferenciadas, en las cuales se pudiese comprobar de manera minimalista las funcionalidades del simulador propuesto. La idea era que se pudiese comprobar que era posible construir un sistema de secciones interdependientes que respondieran al sistema de arrastrar y soltar (Drag and Drop), de manera que desde la sección 1 se pudiesen tomar componentes y alojarlos en la sección 2, y el código de dicho componente se mostrara en la sección 3. El código presentado a continuación utiliza JAVASCRIPT, específicamente su librería JQUERY, para realizar realiza dicha tarea:

<!DOCTYPE HTML>

<html>

<head>

<style>

#div1, #div2 , #div3{

float: left;

width: 400px;

height: 450px;

margin: 10px;

padding: 10px;

border: 1px solid black;

}

</style>

<script>

function allowDrop(ev) {

ev.preventDefault();

}

function drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

function drop(ev) {

ev.preventDefault();

var data = ev.dataTransfer.getData("text");

ev.target.appendChild(document.getElementById(data));

}

</script>

</head>

<body>

<h2>Drag and Drop</h2>

<p>Drag the image back and forth between the two div elements.</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">

<img src="yopGenial.jpg" draggable="true" ondragstart="drag(event)" id="drag1" width="400" height="400">

<input type="button" value="prueba" draggable="true" ondragstart="drag(event)" id="drag2" >

<select draggable="true" ondragstart="drag(event)" id="drag22">

<option value="seiya">Seiya</option>

<option value="shiryu">Shiryu</option>

<option value="hyoga">Hyoga</option>

<option value="shun">Shun</option>

<option value="ikki">Ikki</option>

</select>

<input type="text" placeholder="prueba dos" draggable="true" ondragstart="drag(event)" id="drag3" >

<label draggable="true" ondragstart="drag(event)" id="drag4" >

<select style="pointer-events: none" onclick="console.log(event)" >

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="mercedes">Mercedes</option>

<option value="audi">Audi</option>

</select>

</label>

<label draggable="true" ondragstart="drag(event)" id="drag5" >

<input type="submit" style="pointer-events: none" onclick="console.log(event)">

</label>

</div>

<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<div id="div3">

<table>

<tr>

<td id="real"></td>

</tr>

</table>

<textarea id="txt" cols="50" rows="20"></textarea>

<div id="log">

<table>

<tr>

<td id="real"></td>

</tr>

</table>

</div>

</div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$('#div2').mouseleave(function(){

var htmlString= ($(this).html());

alert(htmlString);

var htmlParaCajaTexto=document.createTextNode(htmlString);

$( "#txt" ).append(htmlParaCajaTexto) ;

});

});

</script>

</body>

</html>

Código 3. Versión preliminar y minimalista del simulador

El resultado de ejecutar dicho código en un navegador web cualquiera es el siguiente:

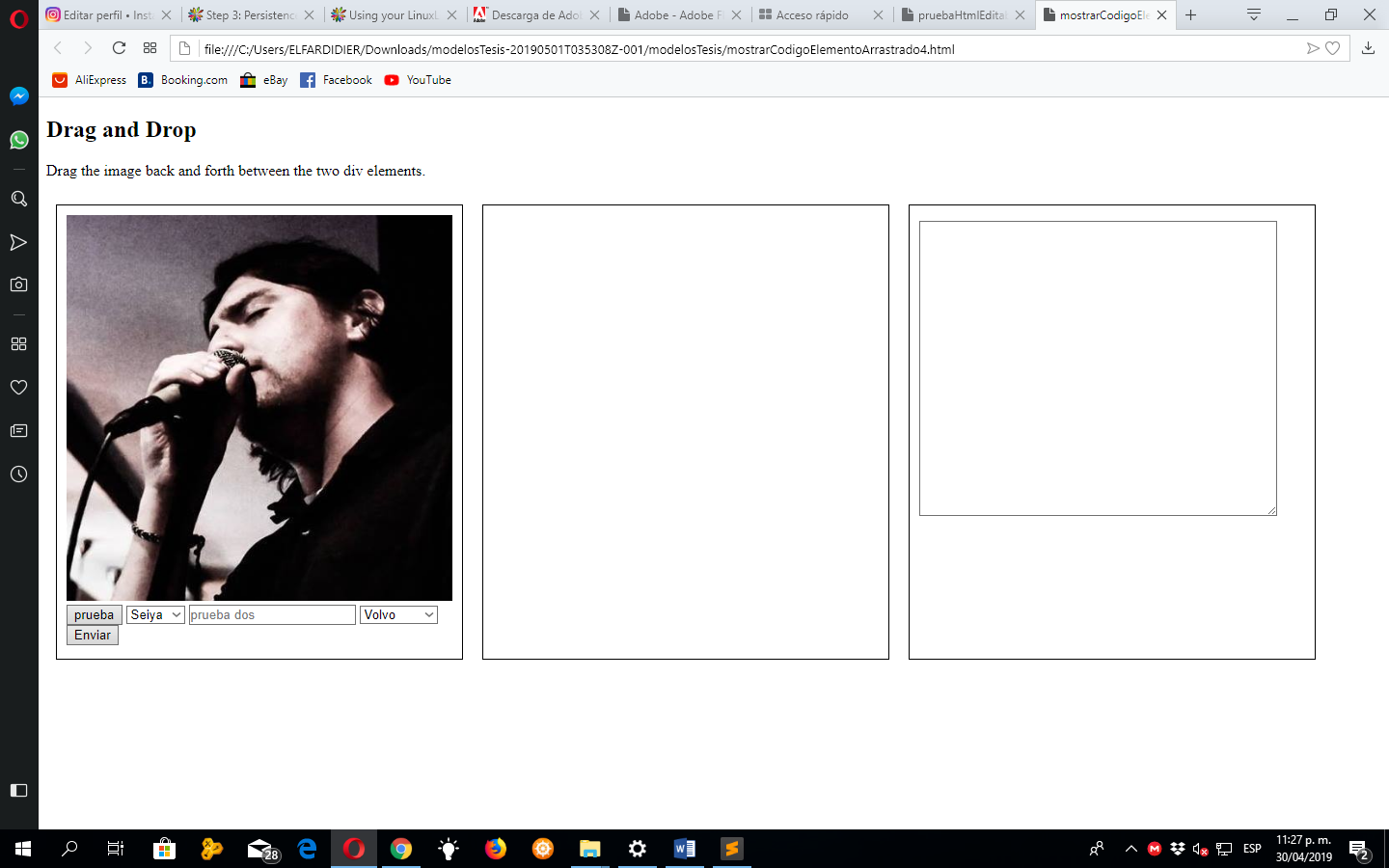


Ilustración 6. Prueba de código de simulador en su versión minimalista y preliminar

Como se puede observar en la ilustración 6, encontramos las tres secciones descritas en la Tabla 1, y en el área de componentes HTML encontramos diversos elementos, como una imagen, unos botones, y unas cajas de texto, entre otros. La prueba resulta interesante cuando se toma cualquier elemento del área de componentes HTML y se suelta en el área de trabajo (segunda sección). En la ilustración 7 se observa el resultado de realizar dicha operación.

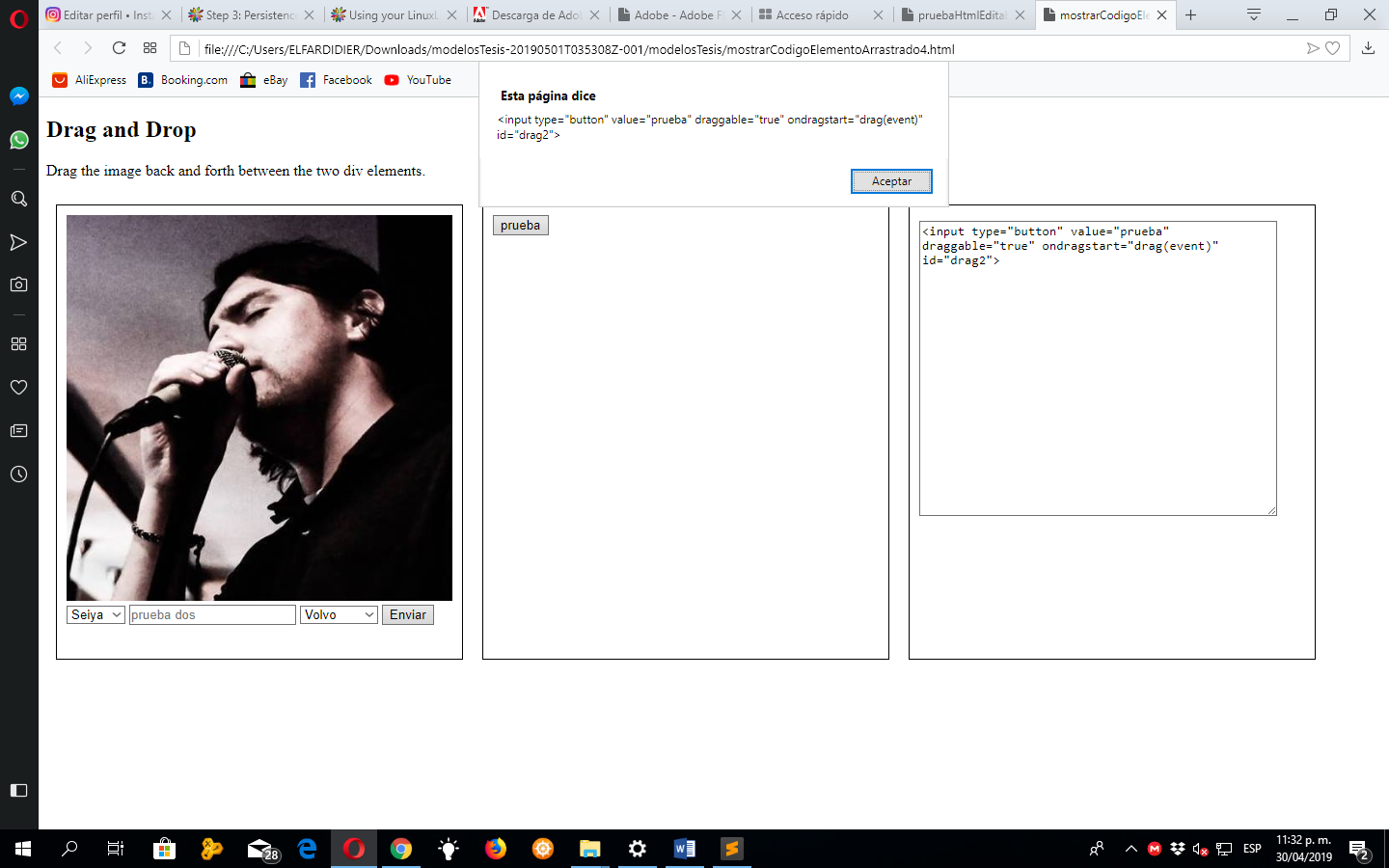


Ilustración 7. Resultado de probar el prototipo del simulador

Como se puede observar en la ilustración 7, se ha arrastrado un botón del área de componentes HTML en el área de trabajo, y ha aparecido el código de dicho componente en el área de código. De esta forma se espera que trabaje el simulador, donde el estudiante neófito en el campo del lenguaje HTML empieza a arrastrar componentes en el área de trabajo, y puede visualizar el código correspondiente a dicho componente en el área de código. Evidentemente esta es una versión preliminar del simulador, pero es una aproximación a lo que se quiere lograr. Que el estudiante comience a relacionar en tiempo real el código de cada componente que decida arrastrar al área de trabajo, y si quiere modificar dicho componente a través del código, podrá visualizar sus cambios en el área de trabajo. Se trata de construir una herramienta intuitiva que le permita al estudiante apropiarse de los conceptos HTML asociados con los formularios.

### Integración de HTML editable en tiempo real y el Prototipo de Sistema Drag and Drop para el Simulador

Una cuarta prueba realizada fue tratar de integrar el prototipo de sistema Drag and Drop para el simulador, con el módulo de HTML editable en tiempo real. La idea era que se pudiese comprobar que era posible construir la interdependencia de la sección de código (sección 3) con la sección donde se sueltan los componentes (sección 2). El código presentado a continuación utiliza JAVASCRIPT, específicamente su librería JQUERY, para realizar realiza dicha tarea:

<!DOCTYPE HTML>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<style>

#div1, #div2 , #div3{

float: left;

width: 400px;

height: 450px;

margin: 10px;

padding: 10px;

border: 1px solid black;

}

</style>

<script>

function allowDrop(ev) {

ev.preventDefault();

}

function drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

function drop(ev) {

ev.preventDefault();

var data = ev.dataTransfer.getData("text");

ev.target.appendChild(document.getElementById(data));

}

</script>

</head>

<body>

<h2>Arrastre y suelte</h2>

<p>arrastre y suelte la imagen entre el primer y segundo recuadro.</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">

<img src="yopGenial.jpg" draggable="true" ondragstart="drag(event)" id="drag1" width="400" height="400">

<input type="button" value="prueba" draggable="true" ondragstart="drag(event)" id="drag2" >

<select draggable="true" ondragstart="drag(event)" id="drag22">

<option value="seiya">Seiya</option>

<option value="shiryu">Shiryu</option>

<option value="hyoga">Hyoga</option>

<option value="shun">Shun</option>

<option value="ikki">Ikki</option>

</select>

<input type="text" placeholder="prueba dos" draggable="true" ondragstart="drag(event)" id="drag3" >

<label draggable="true" ondragstart="drag(event)" id="drag4" >

<select style="pointer-events: none" onclick="console.log(event)" >

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="mercedes">Mercedes</option>

<option value="audi">Audi</option>

</select>

</label>

<label draggable="true" ondragstart="drag(event)" id="drag5" >

<input type="submit" style="pointer-events: none" onclick="console.log(event)">

</label>

</div>

<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<div id="div3">

<!-- inicio del codigo html -->

<textarea id="txt" cols="45" rows="20"></textarea>

</div>

<script>

$(document).ready(function(){

$('#div2').mouseleave(function(){

var htmlString= ($(this).html());

alert(htmlString);

var htmlParaCajaTexto=document.createTextNode(htmlString);

$( "#txt" ).html(htmlString) ;

});

$('#txt').keyup(function(){

$('#div2').html($(this).val());

});

});

</script>

</body>

</html>

Código 4 Integración del módulo de HTML editable y el prototipo del simulador

El resultado de ejecutar dicho código en un navegador web cualquiera es el siguiente:

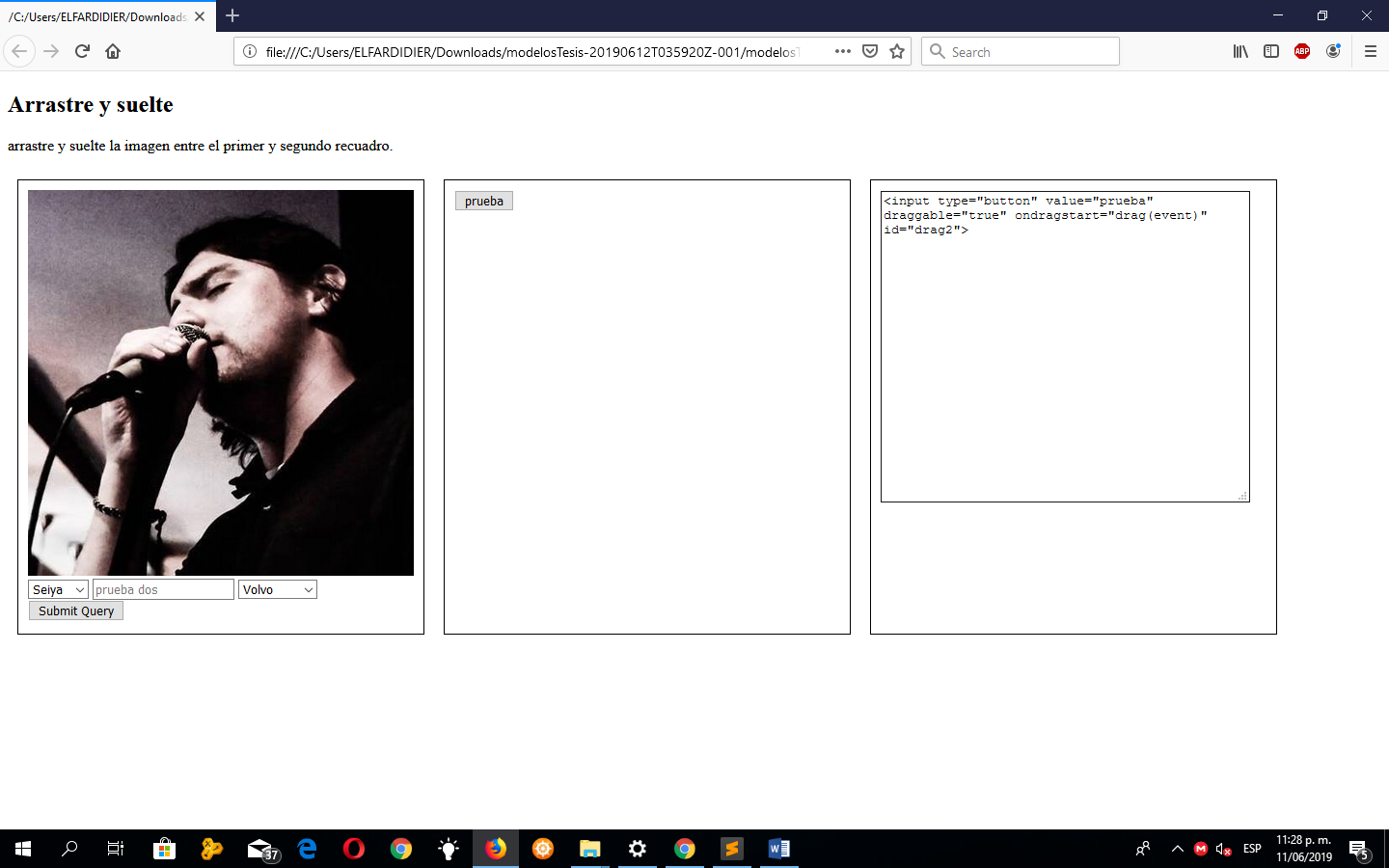


Ilustración 8. Resultado de probar el prototipo del simulador integrado con el módulo de HTML editable

Como se puede observar en la ilustración 8, al arrastrar un componente del área de componentes (sección 1) hacia el área de trabajo (sección 2), encontramos el código correspondiente en el área de código (sección 3). Si arrastramos varios componentes del área de componentes (sección1), encontraremos su código correspondiente en el área de código (sección 3), como se muestra en la ilustración 9.

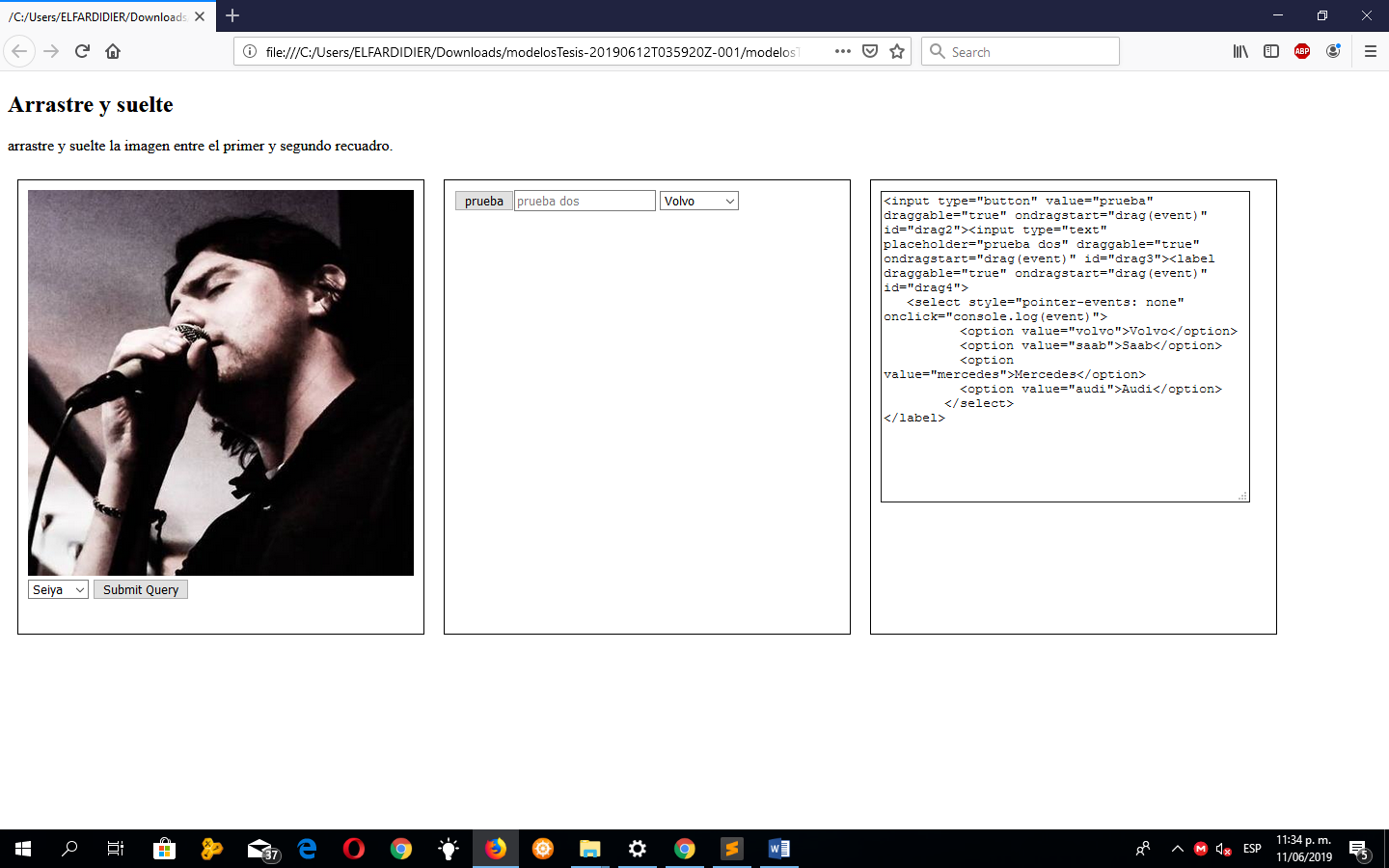


Ilustración 9. Varios componentes arrastrados al área de trabajo

Si se retiran componentes del área de trabajo (sección 2), inmediatamente se eliminarán las instrucciones del área de código (sección 3), como se observa en la ilustración 10.

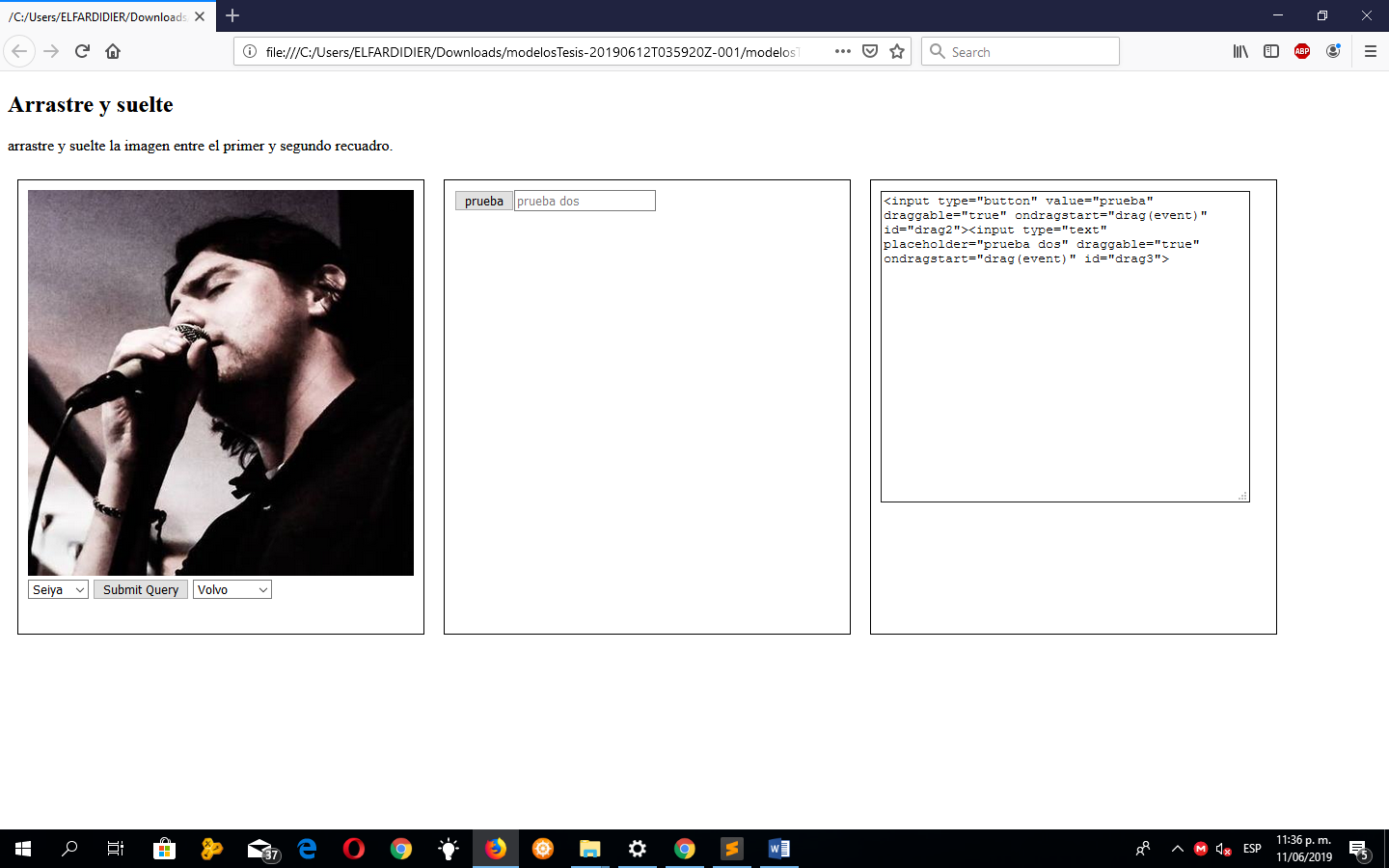


Ilustración 10. Componentes retirados del área de trabajo.

A continuación, se procede a escribir código adicional en el área de código (sección 3), y visualizar si es posible generar automáticamente contenido en el área de trabajo (sección 2), como se observa en la ilustración 11.

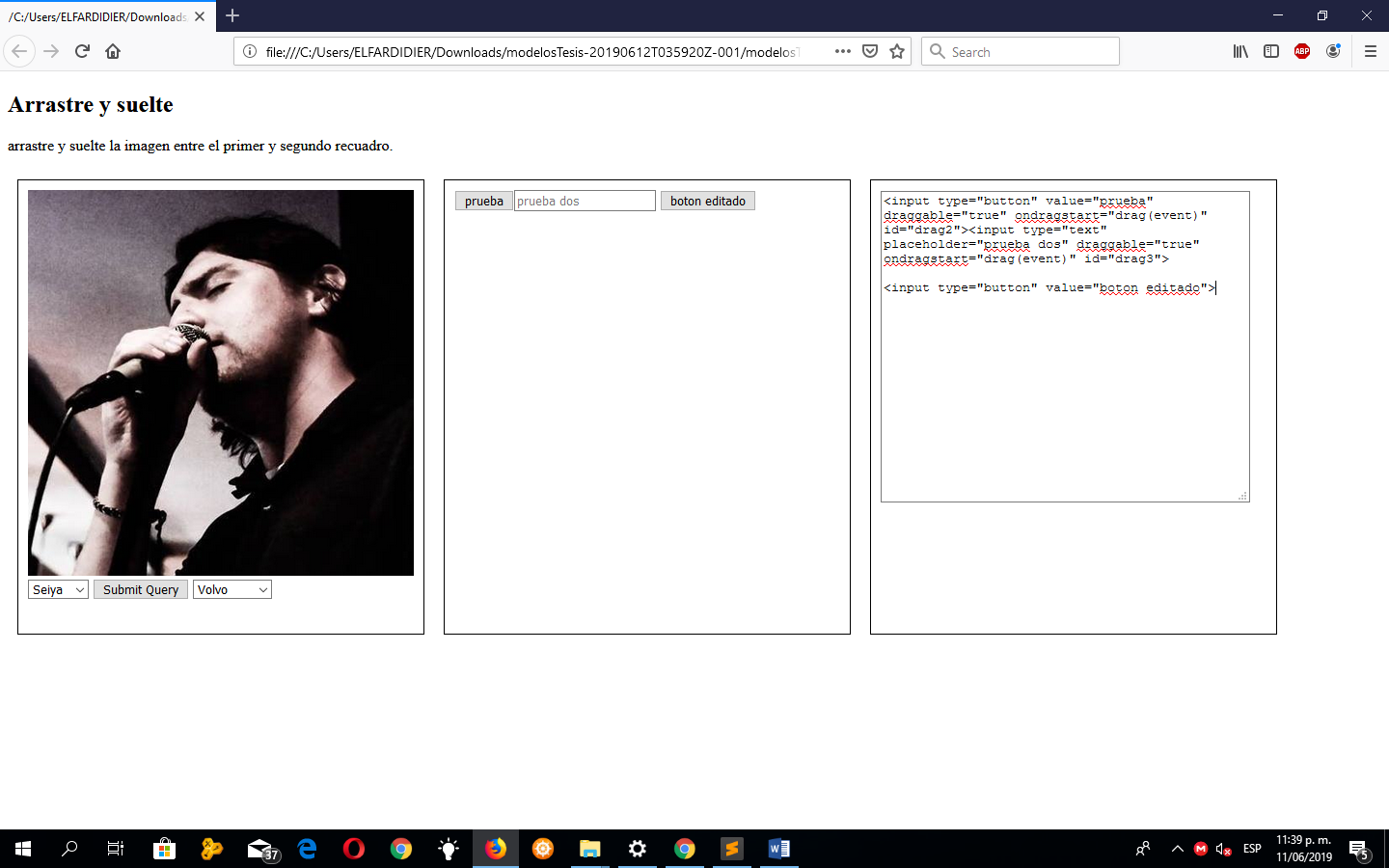


Ilustración 11. Escritura de instrucciones en el área de código y visualización en el área de trabajo

Como se puede observar en la ilustración 11, al escribir instrucciones HTML en el área de código (sección 3), inmediatamente aparece el componente en el área de trabajo (sección 2) siempre y cuando la sintaxis del código HTML sea la adecuada. Sin embargo, en este punto se encuentra que cuando se quiere arrastrar un nuevo elemento del área de componentes (sección 1) al área de trabajo (sección 2), no se actualiza la información en el área de código (sección 3), como se muestra en la ilustración 12.

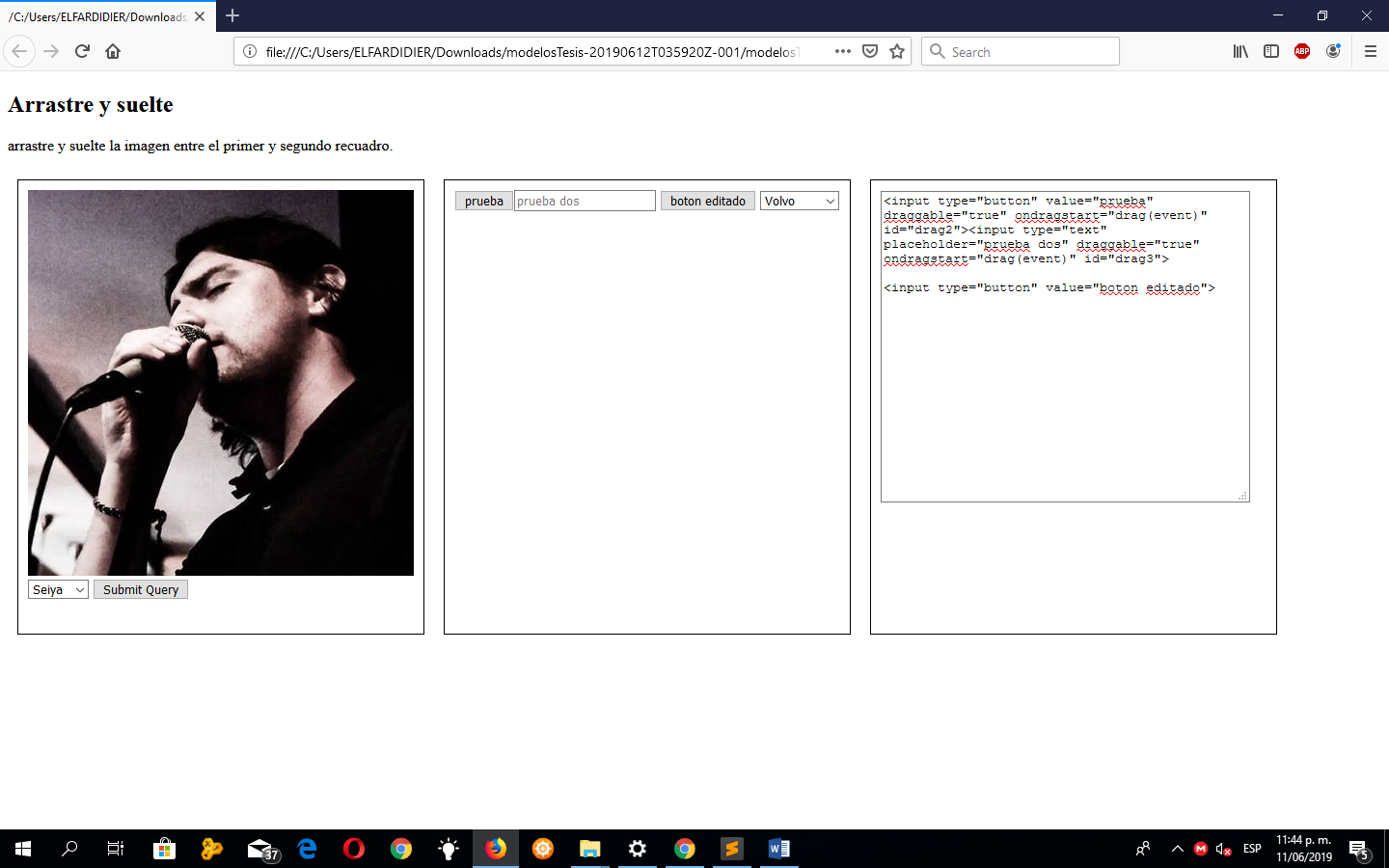


Ilustración 12. Componente no actualizado en el área de código.

Como se puede observar en la ilustración 12, a pesar de que se arrastró un nuevo componente al área de trabajo (sección 2), sus instrucciones no son presentadas en el área de código. Esto quizás se deba a un conflicto de instrucciones, o que sencillamente no es posible realizar la interdependencia de los objetos pues existirá una jerarquía de acciones y componentes que debe ser respetada. De cualquier forma, es una oportunidad para explorar e investigar más el asunto, de forma que se pueda desarrollar el proyecto tal cual se concibió en su idea original.

## Desarrollo del simulador de acuerdo con el modelo, lenguaje de programación, interfaces y formato de presentación seleccionados

De acuerdo con los prototipos anteriormente planteados, se procede a desarrollar formalmente la solución del problema, teniendo en cuenta los principios de arrastre y suelte, de visualizar en una zona el código html de un elemento arrastrado en el lienzo de trabajo, y de editar el código en tiempo real los elementos arrastrados.

### Interfaz Gráfica

El primer cambio es respecto a la interfaz gráfica. En este caso, se solicita al usuario que escoja el layout o diseño de pantalla para arrastrar los componentes. El usuario puede escoger entre un diseño de 2Columnasx12Filas, 3Columnasx12Filas y 12Columnasx12Filas, como se observa en la ilustración 13. Además, se ha usado el framework de CSS bootstrap 5 para poder diseñar la interfaz visual y reutilizarla cuando se edite código HTML en los componentes.

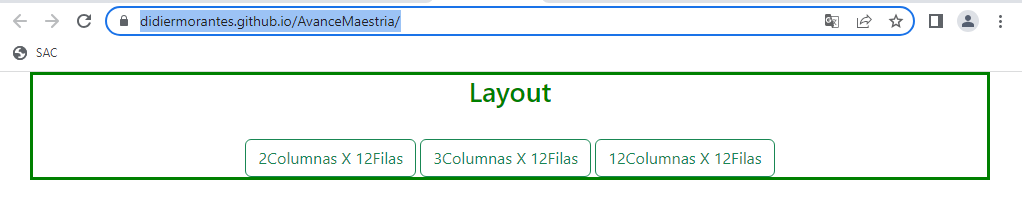


Ilustración 13 . Nueva Interfaz Gráfica

### Sección de Componentes

Cuando se selecciona el tipo de layout, aparece la sección de componentes que poseerá los controles que se pueden arrastrar en la zona de trabajo y un contador de componentes arrastrados, como se muestra en la ilustración 14. Por cada componente que se arrastre a la zona de trabajo, el contador se incrementará en uno. Cuando se seleccione el layout se estableceré automáticamente un límite de componentes a arrastrar en la zona de trabajo así:

* Layout 2Columnasx12Filas: Máximo 24 Componentes
* Layout 3Columnasx12Filas: Máximo 36 Componentes
* Layout 12Columnasx12Filas: Máximo 144 Componentes

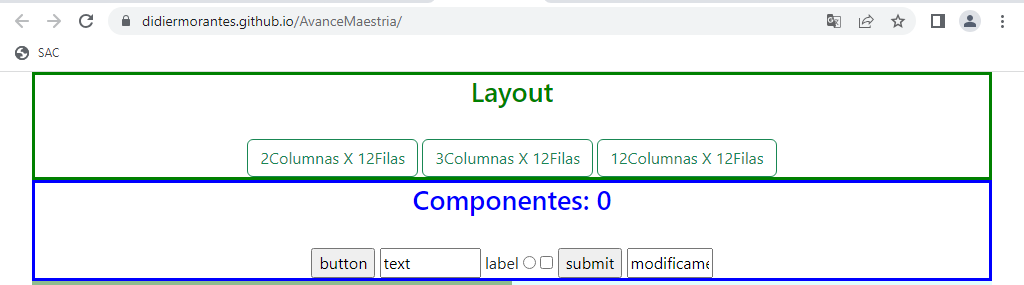


Ilustración 14. Sección de Componentes

### Sección de Arrastrar y Sección de Código

Una vez el usuario ha seleccionado el tipo de layout, se presentan dos columnas con colores. La columna de la izquierda de color verde que será la zona para arrastrar los elementos, y la columna de la derecha de color azul que será la zona donde aparecerá el código fuente referente al elemento arrastrado, como se muestra en la ilustración 15.

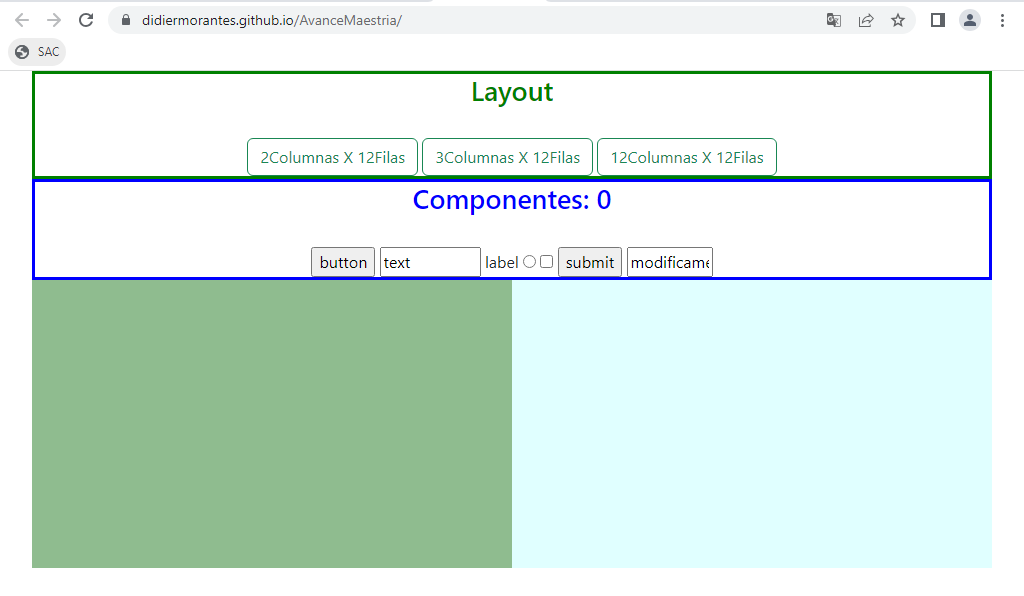


Ilustración 15. Sección de Arrastrar Componentes y Sección de Código

### Funcionalidad de Arrastrar y Soltar

Cuando se haya seleccionado el layout, es posible arrastrar componentes de la sección de componentes, y se muestra cómo se va incrementando el contador de componentes cada vez que se arrastra un elemento en la zona de arrastrar identificada de color verde. Además se observa cómo van apareciendo líneas de código en la sección de código identificada con color azul claro, como se observa en la ilustración 16.

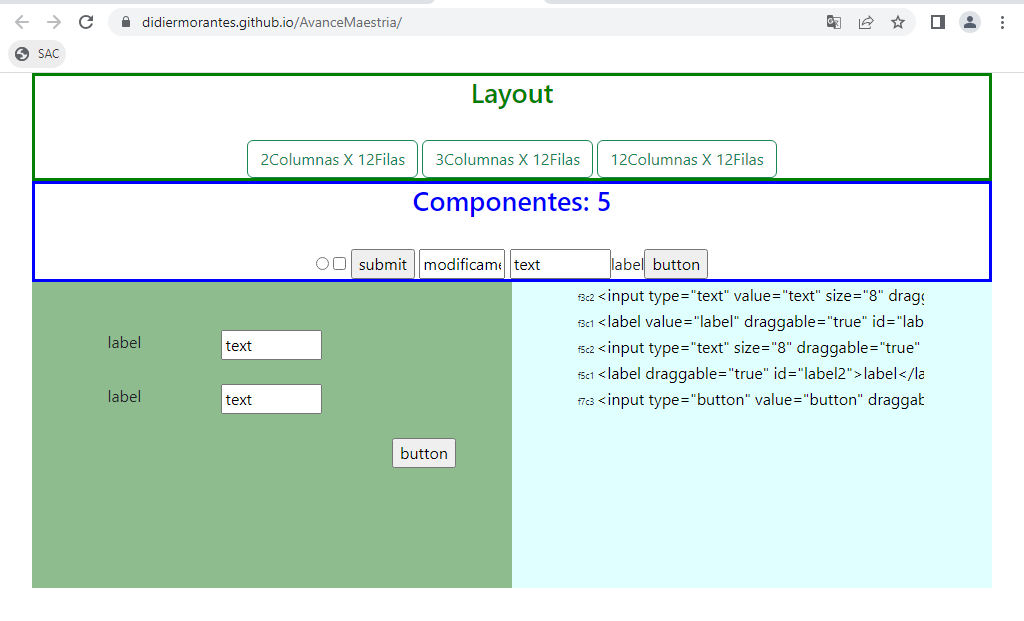


Ilustración 16. Funcionalidad de Arrastrar y Soltar

### Edición de Código

Una vez aparezca la sección de código, es posible editar el código para que en tiempo real aparezcan cambios en los elementos HTML, por ejemplo, se ha editado el botón desde la sección de código para agregarle la clase de bootstrap 5 btn btn-danger, de manera que el botón aparezca de color rojo sobre el lienzo de trabajo, como se muestra en la ilustración 17.

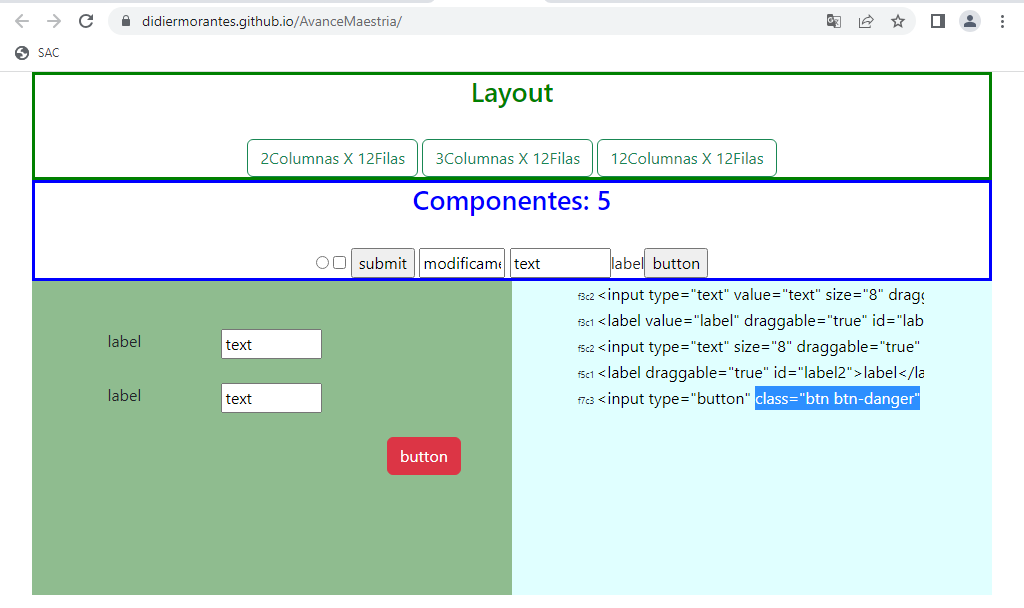


Ilustración 17. Edición de Código

### Eliminación de Componentes

Si se arrastra algún elemento de la sección de arrastrar identificada de color verde, de nuevo hacia la sección de componentes, se borrará el elemento de la sección de arrastrar, y también se borrará el código correspondiente de la zona de código, como se muestra en la ilustración 18, donde se ha arrastrado uno de los label de la zona de arrastre hacia la zona de componentes para eliminarlo.

Además, si se escoge otro layout, como por ejemplo el 12Columnasx12Filas, se borrarán todos los elementos de la zona de arrastre y de la zona de código, como se observa en la ilustración 19

### Código fuente completo

La totalidad del código fuente desarrollado se puede consultar en el repositorio público de GitHub <https://github.com/didiermorantes/AvanceMaestria>

## Implementación del Simulador en página web para probar flexibilidad y portabilidad

Es posible desplegar el entrenador/simulador en distintos sitios web. En principio, y para probar el funcionamiento, se hizo el despliegue en GitHub pages. Dicho proceso no requiere que se invierta dinero, y el único requisito es que el repositorio de código sea público y que no presente errores de JavaScript para permitir a las herramientas de automatización realizar el despliegue. Luego, se desplegó la solución en el dominio didiermorantes.com, de manera que se pudiera visualizar el comportamiento del entrenador simulador en un hosting real de almacenamiento de sitios web.

### Despliegue del Desarrollo Web en GitHub Pages

El despliegue del desarrollo web se puede consultar en el despliegue de GitHub <https://didiermorantes.github.io/AvanceMaestria/>

### Despliegue del Desarrollo Web en didiermorantes.com

<https://didiermorantes.com/AvanceMaestria>

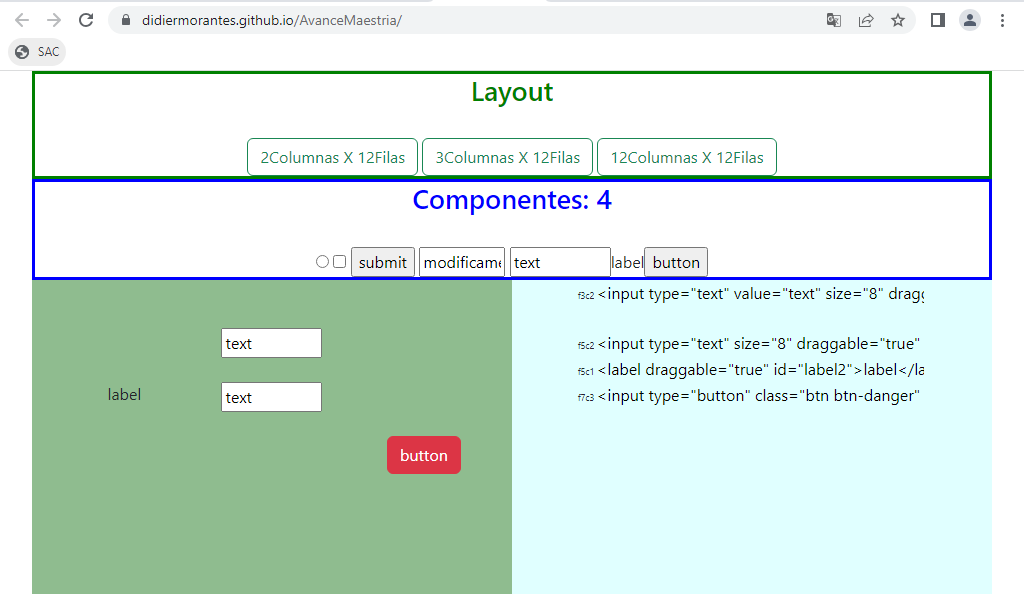


Ilustración 18. Eliminación de Componentes individualtes

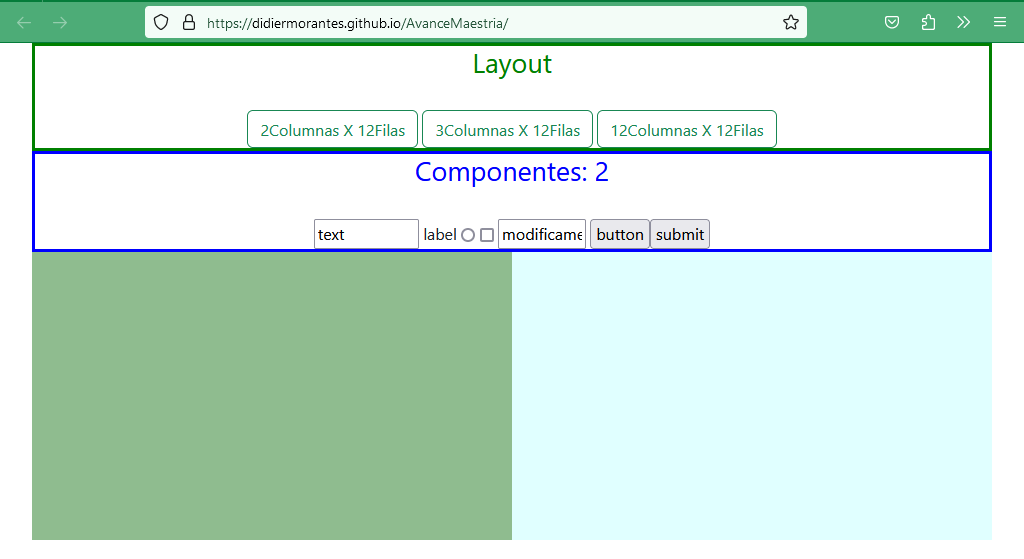


Ilustración 19. Eliminación de componentes al seleccionar layout

## Diseño de actividades relacionadas con el simulador y las teorías, modelos y enfoques pedagógicos seleccionados

Con base en las teorías, modelos y enfoques pedagógicos seleccionados se puede pensar en actividades que permitan desarrollar el conocimiento con el aprendizaje basado en problemas, el cooperativismo, el aprendizaje experiencial, y el socio constructivismo.Se le indica al estudiante que puede complementar la información con el sitio especializado en teoría web <https://www.w3schools.com/> , de manera que el conocimiento sea completo y pueda aplicar los conceptos allí ilustrados mediante el uso del entrenador/simulador.Para utilizar el socio constructivismo y el cooperativismo, se opta por realizar actividades con un mínimo de dos personas y que entre ellos mismos puedan desarrollar y construir el conocimiento. Se proporcionan problemas con objetivos y preguntas planteadas para poner en práctica el aprendizaje basado en problemas, y se pide al estudiante que modifique el código y realice distintas pruebas para poner en práctica el aprendizaje experiencial. A continuación se presentan 15 actividades diferentes diseñadas para utilizarse mancomunadamente con el simulador.

### Actividad 1

Objetivo: Reconocer las características de un botón en HTML

En grupos mínimo de dos personas, arrastrar uno o varios botones en la sección de arrastre y verifique su código fuente. Apoye su investigación mediante la lectura del contenido <https://www.w3schools.com/html/html_form_input_types.asp>

Problema: ¿Qué hay que hacer para convertir un botón convencional a un radio botón o un checkbox? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales y en código existen en los tres tipos de botones?

### Actividad 2

Objetivo: Reconocer las características de una caja de texto en HTML

En grupos mínimo de dos personas, arrastrar una o varias cajas de texto en la sección de arrastre y verifique su código fuente. Apoye su investigación mediante la lectura del contenido <https://www.w3schools.com/html/html_form_input_types.asp>

Problema: ¿Qué hay que hacer para convertir una caja de texto en una caja de texto que reciba sólo números o una caja de texto que reciba contraseñas? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales y en código existen en los tres tipos de cajas de texto?

### Actividad 3

Objetivo: Reconocer las características de un botón submit en HTML

En grupos mínimo de dos personas, arrastrar uno o varios botones submit en la sección de arrastre y verifique su código fuente. Apoye su investigación mediante la lectura del contenido <https://www.w3schools.com/html/html_form_input_types.asp>

Problema: ¿Qué hay que hacer para convertir un botón tipo submit en un botón convencional? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales, de apariencia y en código existen en los dos tipos de botones?

### Actividad 4

Objetivo: Reconocer las características de un botón submit en HTML

En grupos mínimo de dos personas, arrastrar uno o varios botones submit en la sección de arrastre y verifique su código fuente. Apoye su investigación mediante la lectura del contenido <https://www.w3schools.com/html/html_form_input_types.asp>

Problema: ¿Qué hay que hacer para convertir un botón tipo submit en un botón convencional? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales, de apariencia y en código existen en los dos tipos de botones?

### Actividad 5

Objetivo: Reconocer las características de un label

En grupos mínimo de dos personas, arrastrar dos o más label en la sección de arrastre y verifique su código fuente. Apoye su investigación mediante la lectura del contenido <https://www.w3schools.com/html/html_form_input_types.asp>

Problema: ¿Qué hay que hacer para convertir un label en una etiqueta de párrafo o en una etiqueta de encabezado h1? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales, de apariencia y en código existen en los tres tipos de etiquetas? ¿Qué relación tienen los label con las cajas de texto, los radio botones y los botones tipo checkbox?

### Actividad 6

Objetivo: Reconocer las características de un botón checkbox

En grupos mínimo de dos personas, arrastrar dos o más botones checkbox en la sección de arrastre y verifique su código fuente. Apoye su investigación mediante la lectura del contenido <https://www.w3schools.com/html/html_form_input_types.asp>

Problema: ¿Qué hay que hacer para convertir un botón tipo checkbox en un botón radio botón? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales, de apariencia y en código existen en los dos tipos de botones? ¿En qué situaciones usaría botones checkbox?

### Actividad 7

Objetivo: Reconocer las características de un radio botón

En grupos mínimo de dos personas, arrastrar dos o más radio botones en la sección de arrastre y verifique su código fuente. Apoye su investigación mediante la lectura del contenido <https://www.w3schools.com/html/html_form_input_types.asp>

Problema: ¿Qué hay que hacer para convertir un radio botón en un botón checkbox? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales, de apariencia y en código existen en los dos tipos de botones? ¿En qué situaciones usaría radio botones?

### Actividad 8

Objetivo: Reconocer las características de un encabezado h1

En grupos mínimo de dos personas, arrastrar dos o más elementos modificables en la sección de arrastre y verifique su código fuente. Modifique el código fuente del elemento modificable para convertirlo en un encabezado h1. Apoye su investigación mediante la lectura del contenido <https://www.w3schools.com/html/html_headings.asp>

Problema: ¿Qué hay que hacer para convertir una etiqueta cualquiera en una etiqueta de tipo heading? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales, de apariencia y en código existen entre un heading h1 y un heading h6? ¿En qué situaciones usaría los headings?

### Actividad 9

Objetivo: Reconocer las características de párrafo p

En grupos mínimo de dos personas, arrastrar dos o más elementos modificables en la sección de arrastre y verifique su código fuente. Modifique el código fuente del elemento modificable para convertirlo en un párrafo p. Apoye su investigación mediante la lectura del contenido <https://www.w3schools.com/html/html_paragraphs.asp>

<https://www.w3schools.com/tags/tag_span.asp>

Problema: ¿Qué hay que hacer para convertir una etiqueta cualquiera en una etiqueta de tipo párrafo? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales, de apariencia y en código existen entre un párrafo y un span? ¿En qué situaciones usaría los párrafos y en qué situaciones usaría el span?

### Actividad 10

Objetivo: Reconocer las características de las etiquetas para listas

En grupos mínimo de dos personas, arrastrar dos o más elementos modificables en la sección de arrastre y verifique su código fuente. Modifique el código fuente del elemento modificable para convertirlo en una etiqueta de tipo lista. Apoye su investigación mediante la lectura del contenido

<https://www.w3schools.com/html/html_lists.asp>

Problema: ¿Qué hay que hacer para convertir una etiqueta cualquiera en una etiqueta de tipo lista? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Qué diferencias funcionales, de apariencia y en código existen entre una lista desordenada y una lista ordenada? ¿En qué situaciones usaría las etiquetas de tipo lista?

### Actividad 11

Objetivo: Reconocer las características de los links o hipervínculos

En grupos mínimo de dos personas, arrastrar dos o más elementos modificables en la sección de arrastre y verifique su código fuente. Modifique el código fuente del elemento modificable para convertirlo en un vínculo href. Apoye su investigación mediante la lectura del contenido

<https://www.w3schools.com/html/html_links.asp>

Problema: ¿Qué hay que hacer para convertir una etiqueta cualquiera en una etiqueta de tipo vínculo? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿En qué situaciones usaría los vínculos href ?

### Actividad 12

Objetivo: Reconocer las características de las etiquetas para imágenes

En grupos mínimo de dos personas, arrastrar dos o más elementos modificables en la sección de arrastre y verifique su código fuente. Modifique el código fuente del elemento modificable para convertirlo en una etiqueta de tipo imagen. Apoye su investigación mediante la lectura del contenido

<https://www.w3schools.com/html/html_images.asp>

Problema: ¿Qué hay que hacer para convertir una etiqueta cualquiera en una etiqueta de tipo imagen? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿En qué situaciones usaría las etiquetas de tipo imagen ?

### Actividad 13

Objetivo: Reconocer las características de las etiquetas para iframes

En grupos mínimo de dos personas, arrastrar dos o más elementos modificables en la sección de arrastre y verifique su código fuente. Modifique el código fuente del elemento modificable para convertirlo en una etiqueta de tipo iframe. Apoye su investigación mediante la lectura del contenido

<https://www.w3schools.com/html/html_iframe.asp>

Problema: ¿Qué hay que hacer para convertir una etiqueta cualquiera en una etiqueta de tipo iframe? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿En qué situaciones usaría las etiquetas de tipo iframe ?

### Actividad 14

Objetivo: Reconocer las características de las etiquetas para tablas

En grupos mínimo de dos personas, arrastrar dos o más elementos modificables en la sección de arrastre y verifique su código fuente. Modifique el código fuente del elemento modificable para convertirlo en una etiqueta de tipo tabla. Apoye su investigación mediante la lectura del contenido

<https://www.w3schools.com/html/html_tables.asp>

Problema: ¿Qué hay que hacer para convertir una etiqueta cualquiera en una etiqueta de tipo tabla? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿En qué situaciones usaría las etiquetas de tipo tabla ?

### Actividad 15

Objetivo: Reconocer las características de los atributos de estilo

En grupos mínimo de dos personas, arrastrar dos o más elementos modificables en la sección de arrastre y verifique su código fuente. Modifique el código fuente del elemento modificable para convertirlo en una etiqueta de tipo párrafo o de tipo heading y modifique sus estilos internos. Apoye su investigación mediante la lectura del contenido

<https://www.w3schools.com/html/html_styles.asp>

Problema: ¿Qué hay que hacer para utilizar los atributos de estilo dentro de una etiqueta html? Modifique el código en la sección de código del entrenador/simulador y verifique resultados ¿Cuántos atributos de estilo puedo agregar? ¿Estoy limitado a una cantidad específica? ¿En qué situaciones usaría los atributos de estilo ?

## Pruebas de desempeño del simulador implementado como herramienta de aprendizaje

El software fue desarrollado con una metodología que probaba cada módulo en la medida en que se iba desarrollando, de manera que se pudiesen minimizar la cantidad de errores en etapas de funcionamiento. Yo mismo desempeñaba los roles como desarrollador, tester y QA, de manera que si encontraba una funcionalidad equivocada o un error en tiempo de ejecución, inmediatamente procedía a corregir el mal funcionamiento. En la ilustración 20 se muestra un ejemplo de la funcionalidad “cambiarElemento()” que reside en el archivo scripts.js del proyecto que fue cargado en la herramienta de versionamiento Github. Como se puede observar en la ilustración 20, se incluyen bloques try-catch que atrapan errores que puedan ocurrir en tiempo de ejecución. De esta manera probamos que el código se comporte de la manera que esperamos y en caso de ocurrir un error no impedir que el sistema se siga ejecutando.

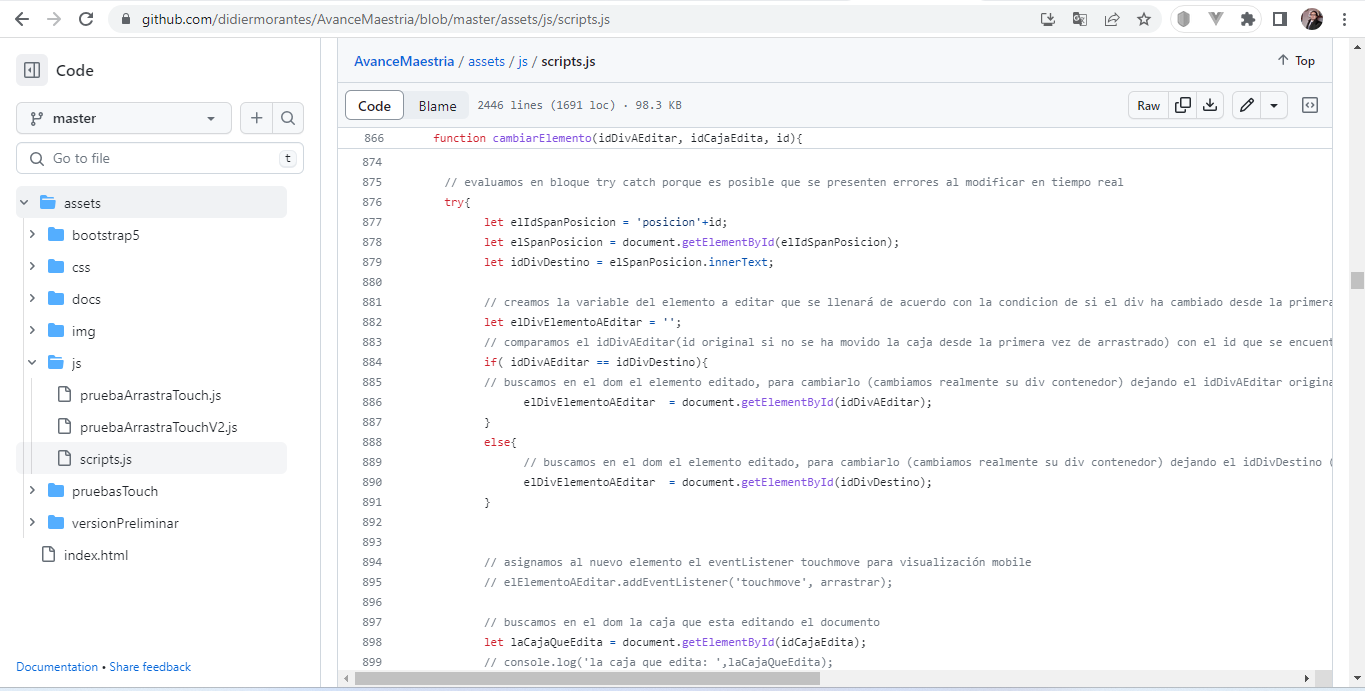


Ilustración 20. Repositorio Github con el archivo javascript que posee la funcionalidad de cambiarElemento

# Resultados Alcanzados

Como se pudo observar en la descripción de la experiencia, se obtuvo una versión preliminar del simulador, que ilustra la funcionalidad clave que se espera obtener, a saber: Que se puedan arrastrar componentes HTML de una primera sección hacia una segunda sección, y que el código de dicho componente se pueda visualizar en una tercera sección. Además, se ahondó en la investigación para hacer más funcional el simulador e incluir distintos componentes y arquitecturas que le permitan al estudiante construir formularios completos desde el simulador. De acuerdo al plan de trabajo, se han alcanzado los objetivos pues se ha realizado una Investigación de modelos, interfaces, y lenguajes de programación aptos para desarrollar el simulador para el aprendizaje del lenguaje de marcas HTML y también se ha realizado una Investigación de la forma en que serán presentados los conceptos en el simulador (figuras, formas, colores, animación, ejemplos, situaciones, organización de conceptos). Luego, dicho prototipo fue refinado y mejorado, de manera que se comporte tal y como fue planteado en los objetivos del proyecto, y cuyo código fuente completo y funcional se relaciona en el Anexo A –Código Fuente de la Solución, y cuyo repositorio Github se encuentra en el siguiente enlace:

Repositorio público del código fuente:

<https://github.com/didiermorantes/AvanceMaestria>

El despliegue en Github Pages del desarrollo web se puede consultar en: <https://didiermorantes.github.io/AvanceMaestria/>

### El Despliegue del Desarrollo Web en el dominio didiermorantes.com:

<https://didiermorantes.com/AvanceMaestria>

Es importante resaltar que el código fuente fue desarrollado entera y completamente por mí, y no existe ninguna página web o documentación que indique cómo construir el simulador, pues fue concebido y desarrollado plenamente por mí, con los conceptos de desarrollo de software que poseo, y con la experiencia que esta profesión me ha otorgado por años. Nada más en el archivo scripts.js hay aproximadamente 2500 líneas de código incluyendo espacios. Todo el código se encuentra completamente documentado, lo que sustenta y argumenta la evolución del código durante meses de trabajo.

Me hubiese gustado utilizar dicho simulador en una situación de aprendizaje real, pues me desempeño en las noches como docente e instructor de desarrollo de software en distintas universidades de la ciudad de Bogotá, pero eso requeriría otra investigación y un análisis más profundo de las actividades y problemáticas planteadas, para determinar su eficacia y efectividad en un proceso de enseñanza-aprendizaje real. Sin embargo hay muchas variables que se deberían tomar en cuenta, como por ejemplo lograr que se me permita orientar la asignatura de front-end o lenguajes de marcas, luego buscar la alternativa de formación virtual, pues una vez terminó la pandemia de Covid-19 las instituciones de educación superior retornaron a la presencialidad y muy pocas asignaturas quedaron virtuales, y por último, diseñar instrumentos de evaluación que permitan registrar la opinión y sentir de los estudiantes frente al uso del entrenador/simulador, lo que requeriría como se mencionó anteriormente, prácticamente una nueva investigación.

# Conclusiones y Recomendaciones

Se puede concluir parcialmente que es posible construir el simulador propuesto en este trabajo de grado, utilizando el lenguaje de programación JAVASCRIPT y su librería JQUERY, de manera que puede materializarse la solución planteada en la introducción de este trabajo ¿Será factible desarrollar un instrumento que facilite el proceso de edición y visualización de código html, de manera que el estudiante experimente simultáneamente la escritura y la percepción de la página web? Sin embargo, es necesario sortear algunos obstáculos, pues es posible que no se pueda seguir al pie de la letra el diseño inicial del simulador debido a limitaciones del lenguaje de programación elegido. En ese punto, es necesario replantear el diseño, para no cambiar la arquitectura desarrollada, y continuar con un prototipo que posea todas las funcionalidades, pero cambiando su aspecto visual. Por ejemplo, es posible diseñar una sección superior del simulador que posea las tres secciones trabajadas en el diseño inicial, y una sección inferior que permita realizar edición de código en tiempo real, como se ilustra en la figura 13.

|  |  |  |
| --- | --- | --- |
| COMPONENTES HTML  Estos componentes permitirán ser arrastrados, de manera que sean soltados en el área de trabajo | AREA DE TRABAJO  En el área de trabajo se sueltan los componentes y se organizan visualmente de acuerdo a los requerimientos del usuario | AREA DE CÓDIGO  En el área de código se visualizará el código HTML generado, de acuerdo a la organización y componentes proporcionada en el área de trabajo. |
| AREA DE TRABAJO EDITABLE | AREA DE CÓDIGO EDITABLE |

Tabla 2. Nueva Interfaz gráfica propuesta

COLOCAR RECOMENDACIÓN DE CAJA EXCLUSIVA CSS

COLOCAR RECOMENDACIONC CAJA EXCLUSICA JS

# Referencias bibliográficas

Anama, M., Herrera, L., Solarte, A., Torres,M. (2020) Prácticas Educativas Socio constructivistas en Hispanoamérica <https://repository.ucc.edu.co/items/d3ad1099-e9a3-4660-9757-32f8808a15a6> . [Consultado el 04/03/2023 11:10].

Arcila, P., Mendoza, Y., Jaramillo, M., & Cañón, Ó. (2010). Comprensión del significado desde Vygotsky, Bruner y Gergen. Diversitas: Perspectivas en psicología, 6(1), 37-49. <https://www.redalyc.org/pdf/679/67916261004.pdf>

Barrows, H.S. y Tamblyn, R.M. (1980). Problem-based learning: an approach to medical education. New York: Springer Publishing Company.

Berners-Lee, Tim (2000): Weaving the Web, Harper Business.

Boud, D., Cohen, R. y Walker, D. (1993) Understanding learning from experience, en Boud, D., Cohen, R. y Walker, D. (Eds.) Using Experience for Learning, SRHE y Open University Press, Buckingham, págs. 1-17.

Cailliau, Robert (1995): A Short History of the Web. Text of a speech delivered at the launching of the European branch of the W3 Consortium. http://www.netvalley.

com/archives/mirrors/robert\_cailliau\_speech.htm [Consultado el 30/04/2019].

Davison, Daniel B; Chen, Edward (1995): A brief introduction to the internet,

Computers & Geoscience, vol. 21, n.º 6, pp. 731-735

Domingo, J. (2008). El aprendizaje cooperativo. Cuadernos de trabajo social, 21, 231-246. <https://revistas.ucm.es/index.php/CUTS/article/download/CUTS0808110231A/7531> [Consultado el 04/03/2023 18:28].

Espinar, E., Vigueras, J (2020) El aprendizaje experiencial y su impacto en la educación actual. Revista Cubana de Educación Superior. <http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0257-43142020000300012> [Consultado el 04/03/2023 11:25].

García, R., Traver, J. A., & Candela, I. (2001). Aprendizaje cooperativo. Fundamentos, características y técnicas. Madrid: CCS. <https://edicionescalasancias.org/wp-content/uploads/2019/10/Cuaderno-11.pdf>

[Consultado el 02/03/2023 18:45].

González, C. (2018). La enseñanza-aprendizaje del Pensamiento Computacional en edades tempranas: una revisión del estado del arte. Pensamiento computacional, 1-37. <https://www.researchgate.net/profile/Carina-Gonzalez-Gonzalez/publication/323450498_La_ensenanza-aprendizaje_del_Pensamiento_Computacional_en_edades_tempranas_una_revision_del_estado_del_arte/links/5a969e7245851535bcdd5e66/La-ensenanza-aprendizaje-del-Pensamiento-Computacional-en-edades-tempranas-una-revision-del-estado-del-arte.pdf>

[Consultado 05/03/2023 11:06]

Jiménez-Toledo, J. A., Collazos, C., & Revelo-Sánchez, O. (2019). Consideraciones en los procesos de enseñanza-aprendizaje para un primer curso de programación de computadores: una revisión sistemática de la literatura. TecnoLógicas, 22, 83-117. <https://www.redalyc.org/journal/3442/344262226008/html/>

[Consultado 06/03/2023 09:12]

Nelson, Ted (1970): No More Teacher´s Dirty Looks, Computer Decisions

OVEJERO, A. (1990). El aprendizaje cooperativo. Una alternativa efi caz a la enseñanza tradicional. Barcelona: PPU.

Poot-Delgado, C. A. (2013). Retos del aprendizaje basado en problemas. Enseñanza e investigación en psicología, 18(2), 307-314. <https://www.redalyc.org/pdf/292/29228336007.pdf> [Consultado el 04/03/2023 14:36].

Raggett, Dave; Lam, Jenny; Alexander, Ian y Kmiec, Michael (1998): Raggett on HTML 4 Capítulo 2. Addison Wesley Longman. <http://www.w3.org/People/Raggett/book4/ch02.html> [Consultado el 04/04/2019 11:10].

Rodríguez, Zulema & Gutiérrez, Martha & BURITICÁ, OLGA. (2011). EL SOCIOCONSTRUCTIVISMO EN LA ENSEÑANZA Y EL APRENDIZAJE ESCOLAR.

Samper, A. y A. Ramírez(2014): Diseño de una propuesta pedagógica de educación para la seguridad vial estructurada bajo el modelo de aprendizaje experiencial. <https://repository.uniminuto.edu/handle/10656/2918>

[Consultado el 05/04/2023 15:54].

Tabarés G, Raul, (2014) El inicio de la Web: historia y cronología del hipertexto hasta HTML 4.0 (1990-99). <http://revistas.usal.es/index.php/artefactos/article/viewFile/12423/12757> [Consultado el 04/04/2019 10:05]

Villalobos, J. (2009). Proyecto cupi2–una solución integral al problema de enseñar y aprender a programar. Bogotá: Universidad de los Andes.

Villalobos, J.A., Casallas, R., Marcos, K. El Reto de Diseñar un Primer Curso de Programación de Computadores. XIII Congreso Iberoamericano de Educación Superior en Computación, Cali, Colombia, Octubre 2005.

**Anexo A – Código Fuente Completo de la Solución**

**Archivo index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<link rel="stylesheet" href="assets/bootstrap5/css/bootstrap.css">

<link rel="stylesheet" href="assets/css/styles.css">

<link rel="icon" type="image/x-icon" href="assets/img/favicon.ico">

<body onload="asignarEventos()">

<div class="container text-center">

<div class="row">

<div class="col-md-12">

<div class="row">

<div class="col-md-12 seccion-layout">

<span class="text-center titulo-layout"><h3>Layout</h3></span><br>

<button id="btn2x12" class="btn btn-outline-success">2Columnas X 12Filas</button>

<button id="btn3x12" class="btn btn-outline-success">3Columnas X 12Filas</button>

<button id="btn12x12" class="btn btn-outline-success">12Columnas X 12Filas</button>

</div>

</div>

<div class="row">

<div id="contenedorControles" class="col-md-12 seccion-controles">

<span class="text-center titulo-controles"><h3>Componentes: <span id="spanCantidadElementos">0</span> </h3></span><br>

<input type="button" value="button" draggable="true" id="button0" >

<input type="text" value="text" size="8" draggable="true" id="text0" >

<label value="label" draggable="true" id="label0" >label</label>

<input type="radio" value="radio" name="grupoRadio" draggable="true" id="radio0" >

<input type="checkbox" value="checkbox" name="checkbox0" draggable="true" id="checkbox0" >

<input type="submit" value="submit" draggable="true" id="submit0" >

<input type="text" value="modificame" size="6" draggable="true" id="independiente0" >

</div>

</div>

</div>

</div>

<div class="row">

<div id='lienzoPadre' class="col-md-6 fondo-lienzo">

</div><!-- fin col-md-4 -->

<div class="col-md-6 fondo-codigo">

<!-- inicio del codigo html. Div para alojar cajas de texto con código html -->

<div id="cajaDeCodigo">

</div>

</div>

</div>

</div>

<script src="assets/bootstrap5/js/bootstrap.bundle.js"></script>

<script src="assets/js/scripts.js"></script>

</body>

</html>

**Archivo styles.css**

/\* se pueden poner bordes, pero no contribuye mucho a la visualización

.contenedoresArrastraSuelta{

border-color: burlywood;

border-style: dotted;

}

\*/

.sin-bordes{

border: none;

outline: none;

background-color: lightcyan;

}

.seccion-layout{

border-style: solid;

border-color: green;

}

.seccion-controles{

border-style: solid;

border-color: blue;

}

.titulo-layout{

color:green;

}

.titulo-controles{

color:blue;

}

.fondo-lienzo{

background-color: darkseagreen;

}

.fondo-codigo{

background-color:lightcyan;

}

**Archivo scripts.js**

// arreglo global para llevar registro de botones

var arrBotones = ['button0'];

// arreglo global para llevar registro de cajas de texto

var arrTexto = ['text0'];

// arreglo global para llevar registro de radio botones

var arrRadio = ['radio0'];

// arreglo global para llevar registro de labels

var arrLabel = ['label0'];

// arreglo global para llevar registro de checkbox

var arrCheckbox = ['checkbox0'];

// arreglo global para llevar registro de submit

var arrSubmit = ['submit0'];

// arreglo global para llevar registro de independiente

var arrIndependiente = ['independiente0'];

// variable para guardar el ultimoTargetId después de haber editado un elemento

var ultimoTargetId = '';

// variable para determinar limite de elementos a arrastrar

var limiteElementos = 0;

// variable para llevar el conteo de elementos en el lienzo

var cantidadElementos = 0;

// variable para guardar los elementos que se han tocado mobile

var elementoTouch1 = '';

// variable para guardar los elementos que se han tocado mobile

var elementoTouch2 = '';

// variable para llevar los conteos de toques

var contadorTouch=0;

function arrastrar(ev) {

// identificamos el id del elemento que está siendo arrastrado

// El método DataTransfer.setData() establece los datos de arrastre de la operación de arrastre en el tipo y datos especificados

// Se agrega el id del elemento arrastrado al objeto DataTransfer

// decidimos si el evento es arrastrar o por touch

if(ev.type=='touchmove'){

// console.log(ev);

// no existe la propiedad dataTransfer en los eventos touch

// extraemos los datos directamente de target

// console.log(arrastrar ev.target.id);

}

else if(ev.type == 'dragstart'){

// El objeto DataTransfer se utiliza para contener los datos que se arrastran durante una operación de arrastrar y soltar

ev.dataTransfer.setData("text/plain", ev.target.id);

// el idAnterior que está produciendo el evento de arrastrar

let idAnterior = ev.target.id;

// console.log('arrastrar idAnterior: ', idAnterior);

// extraemos el ultimo elemento del idAnterior, pues viene en formato letraNumero

let oldIdString = idAnterior.slice(-1);

// si el número es diferente de cero, almacenamos el evento en ultimoTargetId, de lo contrario no porque fue arrastrado de la caja de controles

if(oldIdString != 0){

// almacenamos el ultimoTargetId por si editamos un elemento y queda en blanco

ultimoTargetId = ev.target.id;

}

}

// console.log(ev.target.id);

// console.log(ev.dataTransfer);

}

function permitirSoltar(ev) {

ev.preventDefault();

// console.log(ev);

}

function actualizarIdDivDestino(idAnterior, idDivDestino){

// console.log('actualizarIdDivDestino idAnterior: ', idAnterior);

// console.log('actualizarIdDivDestino idDivDestino: ', idDivDestino);

// actualizamos el span de posicion

let elIdSpanPosicion = 'posicion'+idAnterior;

let elSpanPosicion = document.getElementById(elIdSpanPosicion);

elSpanPosicion.innerText = idDivDestino;

// actualizamos la posicion de la caja de codigo

let elIdCajaDeCodigo = 'editar'+idAnterior;

let laCajaDeCodigo = document.getElementById(elIdCajaDeCodigo);

elSpanPosicion.innerText = idDivDestino;

}

function actualizarSpanCantidadElementos(cantidad){

let elSpanCantElementos = document.getElementById('spanCantidadElementos');

elSpanCantElementos.innerText = cantidad;

}

function generarNuevoId(idAnterior, idDivDestino){

// console.log('generarNuevoId: data idAnterior',idAnterior);

// extraemos el nombre del id sin contar el número

let nombreId = idAnterior.slice(0,-1);

// console.log(nombreId);

// extraemos el ultimo elemento del idAnterior, pues viene en formato letraNumero

let oldIdString = idAnterior.slice(-1);

// convertimos el ultimo elemento en número

let oldIdInt = Number(oldIdString);

// console.log('generarNuevoId oldIdInt: ', oldIdInt);

// variable para el nuevo elemento arrastrado

let elNuevoElemento = '';

// nuevoNombreId para el botón arrastrado

let nuevoNombreId = '';

// evaluamos el nombreId, pues nos indicará si es un botón, una caja de texto , o algún elemento funcional modelado

if(nombreId == 'button'){

// si el oldIdInt es diferente de cero, significa que es un botón que ya existe en el lienzo, y no debemos incrementar ningún id

// si el oldIdInt es cero, significa que es un botón que se arrastró de la sección de controles, y debemos registrar dicho boton en el arreglo global de botones

if(oldIdInt==0){

//validamos el arreglo de elementos para incrementar en uno

// se diseña la lógica para que la cantidad de elementos almacenados en el array coincida con la cantidad de elementos en pantalla

// let longitudArregloBotones = arrBotones.length;

// se diseña la lógica para extraer el último elemento del arreglo, extraer el último caracter y luego incrementarlo en uno

let ultimoElementoArreglo = arrBotones[arrBotones.length - 1 ];

// extraemos el ultimo caracter del ultimoElementoArreglo, pues viene en formato letraNumero

let oldIdUltimoElementoString = ultimoElementoArreglo.slice(-1);

// convertimos el ultimo elemento en número

let oldIdUltimoElementoInt = Number(oldIdUltimoElementoString);

let longitudArregloBotones = oldIdUltimoElementoInt+1;

// Generamos un nuevo id si se arrastra de la sección de controles, y ya hay elementos del mismo tipo en el lienzo

// le sumamos 1 al ultimo elemento (el numero) del id que haya en el arreglo global de elementos para generar un nuevo id

// con la propiedad .length del arreglo ya trae incrementado en uno la cantidad de elementos del arreglo, pues ya existe un primer elemento en el arreglo -boton0- no hay que hacer la operación matemática, solamente asignar el valor que entregue la propiedad .length

newIdInt = longitudArregloBotones;

// console.log(newIdInt);

// nuevoNombreId para el botón arrastrado, que será el nombre anterior sin número concatenado con el nuevo número generado

nuevoNombreId = nombreId+newIdInt;

// console.log(nuevoNombreId);

// Buscamos el elemento recien arrastrado

elNuevoElemento = document.getElementById(idAnterior);

// console.log('elNuevoElemento', elNuevoElemento);

// le cambiamos el id por el nuevo id generado, es decir, el número incrementado en uno para diferenciarlo del botón de base

elNuevoElemento.id = nuevoNombreId;

// console.log('elNuevoElemento', elNuevoElemento);

// registramos el nuevo nombre del elemento en el arreglo global de elementos correspondiente

arrBotones.push(nuevoNombreId);

// console.log(arrBotones);

// El orden para regenerar el elemento importa. Si regenero primero antes de cambiar el id del elemento recién arrastrado, se identificará en el DOM el elemento recién generado en la sección de controles

// si el ultimo elemento es cero, significa que es la primera vez que se arrastra de la sección de controles

// regeneramos el elemento base si es la primera vez que se arrastra

regenerarElementoBase(idAnterior);

}// fin oldInt

}// fin if boton

else if(nombreId == 'text'){

// si el oldIdInt es diferente de cero, significa que es un texto que ya existe en el lienzo, y no debemos incrementar ningún id

// si el oldIdInt es cero, significa que es un texto que se arrastró de la sección de controles, y debemos registrar dicho texto en el arreglo global de texto

if(oldIdInt==0){

//validamos el arreglo de elementos para incrementar en uno

// se diseña la lógica para que la cantidad de elementos almacenados en el array coincida con la cantidad de elementos en pantalla

// let longitudArregloTexto = arrTexto.length;

// se diseña la lógica para extraer el último elemento del arreglo, extraer el último caracter y luego incrementarlo en uno

let ultimoElementoArreglo = arrTexto[arrTexto.length - 1 ];

// extraemos el ultimo caracter del ultimoElementoArreglo, pues viene en formato letraNumero

let oldIdUltimoElementoString = ultimoElementoArreglo.slice(-1);

// convertimos el ultimo elemento en número

let oldIdUltimoElementoInt = Number(oldIdUltimoElementoString);

let longitudArregloTexto = oldIdUltimoElementoInt+1;

// Generamos un nuevo id si se arrastra de la sección de controles, y ya hay elementos del mismo tipo en el lienzo

// le sumamos 1 al ultimo elemento (el numero) del id que haya en el arreglo global de elementos para generar un nuevo id

// con la propiedad .length del arreglo ya trae incrementado en uno la cantidad de elementos del arreglo, pues ya existe un primer elemento en el arreglo -texto0- no hay que hacer la operación matemática, solamente asignar el valor que entregue la propiedad .length

newIdInt = longitudArregloTexto;

// console.log(newIdInt);

// nuevoNombreId para el texto arrastrado, que será el nombre anterior sin número concatenado con el nuevo número generado

nuevoNombreId = nombreId+newIdInt;

// console.log(nuevoNombreId);

// Buscamos el elemento recien arrastrado

elNuevoElemento = document.getElementById(idAnterior);

// console.log('elNuevoElemento', elNuevoElemento);

// le cambiamos el id por el nuevo id generado, es decir, el número incrementado en uno para diferenciarlo del texto de base

elNuevoElemento.id = nuevoNombreId;

// console.log('elNuevoElemento', elNuevoElemento);

// registramos el nuevo nombre del elemento en el arreglo global de elementos correspondiente

arrTexto.push(nuevoNombreId);

// console.log(arrTexto);

// El orden para regenerar el elemento importa. Si regenero primero antes de cambiar el id del elemento recién arrastrado, se identificará en el DOM el elemento recién generado en la sección de controles

// si el ultimo elemento es cero, significa que es la primera vez que se arrastra de la sección de controles

// regeneramos el elemento base si es la primera vez que se arrastra

regenerarElementoBase(idAnterior);

}// fin oldInt

}// fin if texto

else if(nombreId == 'radio'){

// si el oldIdInt es diferente de cero, significa que es un radio que ya existe en el lienzo, y no debemos incrementar ningún id

// si el oldIdInt es cero, significa que es un radio que se arrastró de la sección de controles, y debemos registrar dicho radio en el arreglo global de radio botones

if(oldIdInt==0){

//validamos el arreglo de elementos para incrementar en uno

// se diseña la lógica para que la cantidad de elementos almacenados en el array coincida con la cantidad de elementos en pantalla

// let longitudArregloRadio = arrRadio.length;

// se diseña la lógica para extraer el último elemento del arreglo, extraer el último caracter y luego incrementarlo en uno

let ultimoElementoArreglo = arrRadio[arrRadio.length - 1 ];

// extraemos el ultimo caracter del ultimoElementoArreglo, pues viene en formato letraNumero

let oldIdUltimoElementoString = ultimoElementoArreglo.slice(-1);

// convertimos el ultimo elemento en número

let oldIdUltimoElementoInt = Number(oldIdUltimoElementoString);

let longitudArregloRadio = oldIdUltimoElementoInt+1;

// Generamos un nuevo id si se arrastra de la sección de controles, y ya hay elementos del mismo tipo en el lienzo

// le sumamos 1 al ultimo elemento (el numero) del id que haya en el arreglo global de elementos para generar un nuevo id

// con la propiedad .length del arreglo ya trae incrementado en uno la cantidad de elementos del arreglo, pues ya existe un primer elemento en el arreglo -radio0- no hay que hacer la operación matemática, solamente asignar el valor que entregue la propiedad .length

newIdInt = longitudArregloRadio;

// console.log(newIdInt);

// nuevoNombreId para el radio arrastrado, que será el nombre anterior sin número concatenado con el nuevo número generado

nuevoNombreId = nombreId+newIdInt;

// console.log(nuevoNombreId);

// Buscamos el elemento recien arrastrado

elNuevoElemento = document.getElementById(idAnterior);

// console.log('elNuevoElemento', elNuevoElemento);

// le cambiamos el id por el nuevo id generado, es decir, el número incrementado en uno para diferenciarlo del radio de base

elNuevoElemento.id = nuevoNombreId;

// console.log('elNuevoElemento', elNuevoElemento);

// registramos el nuevo nombre del elemento en el arreglo global de elementos correspondiente

arrRadio.push(nuevoNombreId);

// console.log(arrRadio);

// El orden para regenerar el elemento importa. Si regenero primero antes de cambiar el id del elemento recién arrastrado, se identificará en el DOM el elemento recién generado en la sección de controles

// si el ultimo elemento es cero, significa que es la primera vez que se arrastra de la sección de controles

// regeneramos el elemento base si es la primera vez que se arrastra

regenerarElementoBase(idAnterior);

}// fin oldInt

}// fin if radio

else if(nombreId == 'label'){

// si el oldIdInt es diferente de cero, significa que es un label que ya existe en el lienzo, y no debemos incrementar ningún id

// si el oldIdInt es cero, significa que es un label que se arrastró de la sección de controles, y debemos registrar dicho label en el arreglo global de label

if(oldIdInt==0){

//validamos el arreglo de elementos para incrementar en uno

// se diseña la lógica para que la cantidad de elementos almacenados en el array coincida con la cantidad de elementos en pantalla

// let longitudArregloLabel = arrLabel.length;

// se diseña la lógica para extraer el último elemento del arreglo, extraer el último caracter y luego incrementarlo en uno

let ultimoElementoArreglo = arrLabel[arrLabel.length - 1 ];

// extraemos el ultimo caracter del ultimoElementoArreglo, pues viene en formato letraNumero

let oldIdUltimoElementoString = ultimoElementoArreglo.slice(-1);

// convertimos el ultimo elemento en número

let oldIdUltimoElementoInt = Number(oldIdUltimoElementoString);

let longitudArregloLabel = oldIdUltimoElementoInt+1;

// Generamos un nuevo id si se arrastra de la sección de controles, y ya hay elementos del mismo tipo en el lienzo

// le sumamos 1 al ultimo elemento (el numero) del id que haya en el arreglo global de elementos para generar un nuevo id

// con la propiedad .length del arreglo ya trae incrementado en uno la cantidad de elementos del arreglo, pues ya existe un primer elemento en el arreglo -label0- no hay que hacer la operación matemática, solamente asignar el valor que entregue la propiedad .length

newIdInt = longitudArregloLabel;

// console.log(newIdInt);

// nuevoNombreId para el label arrastrado, que será el nombre anterior sin número concatenado con el nuevo número generado

nuevoNombreId = nombreId+newIdInt;

// console.log(nuevoNombreId);

// Buscamos el elemento recien arrastrado

elNuevoElemento = document.getElementById(idAnterior);

// console.log('elNuevoElemento', elNuevoElemento);

// le cambiamos el id por el nuevo id generado, es decir, el número incrementado en uno para diferenciarlo del label de base

elNuevoElemento.id = nuevoNombreId;

// console.log('elNuevoElemento', elNuevoElemento);

// registramos el nuevo nombre del elemento en el arreglo global de elementos correspondiente

arrLabel.push(nuevoNombreId);

// console.log(arrLabel);

// El orden para regenerar el elemento importa. Si regenero primero antes de cambiar el id del elemento recién arrastrado, se identificará en el DOM el elemento recién generado en la sección de controles

// si el ultimo elemento es cero, significa que es la primera vez que se arrastra de la sección de controles

// regeneramos el elemento base si es la primera vez que se arrastra

regenerarElementoBase(idAnterior);

}// fin oldInt

}// fin if label

else if(nombreId == 'checkbox'){

// si el oldIdInt es diferente de cero, significa que es un checkbox que ya existe en el lienzo, y no debemos incrementar ningún id

// si el oldIdInt es cero, significa que es un checkbox que se arrastró de la sección de controles, y debemos registrar dicho checkbox en el arreglo global de checkbox

if(oldIdInt==0){

//validamos el arreglo de elementos para incrementar en uno

// se diseña la lógica para que la cantidad de elementos almacenados en el array coincida con la cantidad de elementos en pantalla

// let longitudArregloCheckbox = arrCheckbox.length;

// se diseña la lógica para extraer el último elemento del arreglo, extraer el último caracter y luego incrementarlo en uno

let ultimoElementoArreglo = arrCheckbox[arrCheckbox.length - 1 ];

// extraemos el ultimo caracter del ultimoElementoArreglo, pues viene en formato letraNumero

let oldIdUltimoElementoString = ultimoElementoArreglo.slice(-1);

// convertimos el ultimo elemento en número

let oldIdUltimoElementoInt = Number(oldIdUltimoElementoString);

let longitudArregloCheckbox = oldIdUltimoElementoInt+1;

// Generamos un nuevo id si se arrastra de la sección de controles, y ya hay elementos del mismo tipo en el lienzo

// le sumamos 1 al ultimo elemento (el numero) del id que haya en el arreglo global de elementos para generar un nuevo id

// con la propiedad .length del arreglo ya trae incrementado en uno la cantidad de elementos del arreglo, pues ya existe un primer elemento en el arreglo -checkbox0- no hay que hacer la operación matemática, solamente asignar el valor que entregue la propiedad .length

newIdInt = longitudArregloCheckbox;

// console.log(newIdInt);

// nuevoNombreId para el checkbox arrastrado, que será el nombre anterior sin número concatenado con el nuevo número generado

nuevoNombreId = nombreId+newIdInt;

// console.log(nuevoNombreId);

// Buscamos el elemento recien arrastrado

elNuevoElemento = document.getElementById(idAnterior);

// console.log('elNuevoElemento', elNuevoElemento);

// le cambiamos el id por el nuevo id generado, es decir, el número incrementado en uno para diferenciarlo del checkbox de base

elNuevoElemento.id = nuevoNombreId;

// console.log('elNuevoElemento', elNuevoElemento);

// registramos el nuevo nombre del elemento en el arreglo global de elementos correspondiente

arrCheckbox.push(nuevoNombreId);

// console.log(arrCheckbox);

// El orden para regenerar el elemento importa. Si regenero primero antes de cambiar el id del elemento recién arrastrado, se identificará en el DOM el elemento recién generado en la sección de controles

// si el ultimo elemento es cero, significa que es la primera vez que se arrastra de la sección de controles

// regeneramos el elemento base si es la primera vez que se arrastra

regenerarElementoBase(idAnterior);

}// fin oldInt

}// fin if checkbox

else if(nombreId == 'submit'){

// si el oldIdInt es diferente de cero, significa que es un submti que ya existe en el lienzo, y no debemos incrementar ningún id

// si el oldIdInt es cero, significa que es un submit que se arrastró de la sección de controles, y debemos registrar dicho submit en el arreglo global de submit

if(oldIdInt==0){

//validamos el arreglo de elementos para incrementar en uno

// se diseña la lógica para que la cantidad de elementos almacenados en el array coincida con la cantidad de elementos en pantalla

// let longitudArregloSubmit = arrSubmit.length;

// se diseña la lógica para extraer el último elemento del arreglo, extraer el último caracter y luego incrementarlo en uno

let ultimoElementoArreglo = arrSubmit[arrSubmit.length - 1 ];

// extraemos el ultimo caracter del ultimoElementoArreglo, pues viene en formato letraNumero

let oldIdUltimoElementoString = ultimoElementoArreglo.slice(-1);

// convertimos el ultimo elemento en número

let oldIdUltimoElementoInt = Number(oldIdUltimoElementoString);

let longitudArregloSubmit = oldIdUltimoElementoInt+1;

// Generamos un nuevo id si se arrastra de la sección de controles, y ya hay elementos del mismo tipo en el lienzo

// le sumamos 1 al ultimo elemento (el numero) del id que haya en el arreglo global de elementos para generar un nuevo id

// con la propiedad .length del arreglo ya trae incrementado en uno la cantidad de elementos del arreglo, pues ya existe un primer elemento en el arreglo -submit0- no hay que hacer la operación matemática, solamente asignar el valor que entregue la propiedad .length

newIdInt = longitudArregloSubmit;

// console.log(newIdInt);

// nuevoNombreId para el submit arrastrado, que será el nombre anterior sin número concatenado con el nuevo número generado

nuevoNombreId = nombreId+newIdInt;

// console.log(nuevoNombreId);

// Buscamos el elemento recien arrastrado

elNuevoElemento = document.getElementById(idAnterior);

// console.log('elNuevoElemento', elNuevoElemento);

// le cambiamos el id por el nuevo id generado, es decir, el número incrementado en uno para diferenciarlo del submit de base

elNuevoElemento.id = nuevoNombreId;

// console.log('elNuevoElemento', elNuevoElemento);

// registramos el nuevo nombre del elemento en el arreglo global de elementos correspondiente

arrSubmit.push(nuevoNombreId);

// console.log(arrSubmit);

// El orden para regenerar el elemento importa. Si regenero primero antes de cambiar el id del elemento recién arrastrado, se identificará en el DOM el elemento recién generado en la sección de controles

// si el ultimo elemento es cero, significa que es la primera vez que se arrastra de la sección de controles

// regeneramos el elemento base si es la primera vez que se arrastra

regenerarElementoBase(idAnterior);

}// fin oldInt

}// fin if submit

else if(nombreId == 'independiente'){

// si el oldIdInt es diferente de cero, significa que es un submti que ya existe en el lienzo, y no debemos incrementar ningún id

// si el oldIdInt es cero, significa que es un submit que se arrastró de la sección de controles, y debemos registrar dicho submit en el arreglo global de submit

if(oldIdInt==0){

//validamos el arreglo de elementos para incrementar en uno

// se diseña la lógica para que la cantidad de elementos almacenados en el array coincida con la cantidad de elementos en pantalla

// let longitudArregloSubmit = arrSubmit.length;

// se diseña la lógica para extraer el último elemento del arreglo, extraer el último caracter y luego incrementarlo en uno

let ultimoElementoArreglo = arrIndependiente[arrIndependiente.length - 1 ];

// extraemos el ultimo caracter del ultimoElementoArreglo, pues viene en formato letraNumero

let oldIdUltimoElementoString = ultimoElementoArreglo.slice(-1);

// convertimos el ultimo elemento en número

let oldIdUltimoElementoInt = Number(oldIdUltimoElementoString);

let longitudArregloSubmit = oldIdUltimoElementoInt+1;

// Generamos un nuevo id si se arrastra de la sección de controles, y ya hay elementos del mismo tipo en el lienzo

// le sumamos 1 al ultimo elemento (el numero) del id que haya en el arreglo global de elementos para generar un nuevo id

// con la propiedad .length del arreglo ya trae incrementado en uno la cantidad de elementos del arreglo, pues ya existe un primer elemento en el arreglo -submit0- no hay que hacer la operación matemática, solamente asignar el valor que entregue la propiedad .length

newIdInt = longitudArregloSubmit;

// console.log(newIdInt);

// nuevoNombreId para el submit arrastrado, que será el nombre anterior sin número concatenado con el nuevo número generado

nuevoNombreId = nombreId+newIdInt;

// console.log(nuevoNombreId);

// Buscamos el elemento recien arrastrado

elNuevoElemento = document.getElementById(idAnterior);

// console.log('elNuevoElemento', elNuevoElemento);

// le cambiamos el id por el nuevo id generado, es decir, el número incrementado en uno para diferenciarlo del submit de base

elNuevoElemento.id = nuevoNombreId;

// console.log('elNuevoElemento', elNuevoElemento);

// registramos el nuevo nombre del elemento en el arreglo global de elementos correspondiente

arrIndependiente.push(nuevoNombreId);

// console.log(arrSubmit);

// El orden para regenerar el elemento importa. Si regenero primero antes de cambiar el id del elemento recién arrastrado, se identificará en el DOM el elemento recién generado en la sección de controles

// si el ultimo elemento es cero, significa que es la primera vez que se arrastra de la sección de controles

// regeneramos el elemento base si es la primera vez que se arrastra

regenerarElementoBase(idAnterior);

}// fin oldInt

}// fin if independiente

// si no es un nuevo elemento (oldInt==0) no enviamos nada. No invocamos a enviarCodigoHTML

if(oldIdInt==0){

// invocamos al método que enviará el código HTML del nuevo elemento a la sección de código, junto con su id

enviarCodigoHTML(elNuevoElemento,nuevoNombreId, idDivDestino );

// si el oldIdInt es cero, actualizamos la referencia de ultimoTargetId actualizado con el nuevoNombreId

ultimoTargetId = nuevoNombreId;

// console.log('generarNuevoId ultimoTargetId: ',ultimoTargetId);

//incrementamos en uno la cantidad de elementos

cantidadElementos+=1;

// console.log('generarNuevoId cantidadElementos: ', cantidadElementos);

// invocamos la funcion que actualiza el span de la cantidad de elementos

actualizarSpanCantidadElementos(cantidadElementos);

}

else{

// si el ultimo numero del id no es cero, hay que actualizar la caja de código

// console.log('se movio elemento. Posiblemente haya cambiado codigo en la caja de codigo');

// la caja de codigo mantiene la referencia del primer lugar donde se movio y se comenzo a editar codigo

actualizarIdDivDestino(idAnterior, idDivDestino);

}

}// fin function

function EliminarDelArreglo(elementoAeliminar){

// console.log('Eliminar del Arreglo',elementoAeliminar);

// extraemos el nombre del id sin contar el número

let nombreId = elementoAeliminar.slice(0,-1);

// console.log(nombreId);

// extraemos el ultimo elemento del idAnterior, pues viene en formato letraNumero

let oldIdString = elementoAeliminar.slice(-1);

// convertimos el ultimo elemento en número

let oldIdInt = Number(oldIdString);

// evaluamos el nombreId, pues nos indicará si es un botón, una caja de texto , o algún elemento funcional modelado

if(nombreId == 'button'){

// identificamos el indice del elemento a eliminar

let indiceAEliminar = arrBotones.indexOf(elementoAeliminar);

// eliminamos el elemento del arreglo, con el índice identificado

arrBotones.splice(indiceAEliminar, 1);

//mostramos el arreglo después de la eliminación

// console.log(arrBotones);

}// fin if boton

else if(nombreId == 'text'){

// identificamos el indice del elemento a eliminar

let indiceAEliminar = arrTexto.indexOf(elementoAeliminar);

// eliminamos el elemento del arreglo, con el índice identificado

arrTexto.splice(indiceAEliminar, 1);

//mostramos el arreglo después de la eliminación

// console.log(arrTexto);

}// fin if texto

else if(nombreId == 'radio'){

// identificamos el indice del elemento a eliminar

let indiceAEliminar = arrRadio.indexOf(elementoAeliminar);

// eliminamos el elemento del arreglo, con el índice identificado

arrRadio.splice(indiceAEliminar, 1);

//mostramos el arreglo después de la eliminación

// console.log(arrRadio);

}// fin if radio

else if(nombreId == 'label'){

// identificamos el indice del elemento a eliminar

let indiceAEliminar = arrLabel.indexOf(elementoAeliminar);

// eliminamos el elemento del arreglo, con el índice identificado

arrLabel.splice(indiceAEliminar, 1);

//mostramos el arreglo después de la eliminación

// console.log(arrLabel);

}// fin if label

else if(nombreId == 'checkbox'){

// identificamos el indice del elemento a eliminar

let indiceAEliminar = arrCheckbox.indexOf(elementoAeliminar);

// eliminamos el elemento del arreglo, con el índice identificado

arrCheckbox.splice(indiceAEliminar, 1);

//mostramos el arreglo después de la eliminación

// console.log(arrCheckbox);

}// fin if checkbox

else if(nombreId == 'submit'){

// identificamos el indice del elemento a eliminar

let indiceAEliminar = arrSubmit.indexOf(elementoAeliminar);

// eliminamos el elemento del arreglo, con el índice identificado

arrSubmit.splice(indiceAEliminar, 1);

//mostramos el arreglo después de la eliminación

// console.log(arrSubmit);

}// fin if submit

else if(nombreId == 'independiente'){

// identificamos el indice del elemento a eliminar

let indiceAEliminar = arrIndependiente.indexOf(elementoAeliminar);

// eliminamos el elemento del arreglo, con el índice identificado

arrIndependiente.splice(indiceAEliminar, 1);

//mostramos el arreglo después de la eliminación

// console.log(arrSubmit);

}// fin if independiente

}// fin function

function EliminarDeLaCajaCodigo(elementoAeliminar){

// console.log('Eliminar del Arreglo',elementoAeliminar);

// identificamos la caja de código

// ubicamos la caja de código en el DOM

let laCajaDeCodigo = document.getElementById('cajaDeCodigo');

// construimos el id del elemento a eliminar anteponiendole la palabra editar al parámetro recibido

let idElementoAEliminar = 'editar'+ elementoAeliminar;

// identificamos el elemento a eliminar mediante el id construido

let elElementoAEliminar = document.getElementById(idElementoAEliminar);

// removemos de la caja de código el elemento hijo identificado con el id recien configurado

laCajaDeCodigo.removeChild(elElementoAEliminar);

// construimos el id del elemento a eliminar anteponiendole la palabra posicion al parámetro recibido

let idPosicionAEliminar = 'posicion'+ elementoAeliminar;

// identificamos el elemento de posicion a eliminar mediante el id construido

let elElementoPosicionAEliminar = document.getElementById(idPosicionAEliminar);

// removemos de la caja de código el elemento hijo identificado con el id recien configurado

laCajaDeCodigo.removeChild(elElementoPosicionAEliminar);

// Restamos uno de la variable global la cantidad de elementos en el lienzo

cantidadElementos-=1

// actualizar cantidad de elementos en el span. Se hace en este método porque es transversal a los eventos de arrastrar y tocar

actualizarSpanCantidadElementos(cantidadElementos);

}// fin function

function regenerarElementoBase(idAnterior){

// extraemos el nombre del id sin contar el número, de esa manera sabremos que tipo de elemento es

let nombreId = idAnterior.slice(0,-1);

// identificamos en el DOM el contenedor de controles

let elContenedorControles = document.getElementById('contenedorControles');

if(nombreId == 'button'){

// creamos un nuevo elemento en el DOM de tipo input

let elElementoBase = document.createElement('input');

// asignamos al nuevo elemento un tipo botón

elElementoBase.setAttribute('type', 'button');

// asignamos al nuevo elemento la propiedad draggable en true

elElementoBase.setAttribute('draggable', 'true');

// asignamos al nuevo elemento el eventListener dragstart

elElementoBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchstart para visualizacion mobile

elElementoBase.addEventListener('touchstart', tocar);

// le agregamos un cero al nombre del elemento base para indicar que es el primero

elElementoBase.id=nombreId+0;

// le asignamos al nuevo elemento el valor boton

elElementoBase.value='button';

// añadimos al contenedor de controles el nuevo elemento recién creado con id 0, indicando que es un elemento de base

elContenedorControles.appendChild(elElementoBase);

}

else if(nombreId == 'text'){

// creamos un nuevo elemento en el DOM de tipo input

let elElementoBase = document.createElement('input');

// asignamos al nuevo elemento un tipo texto

elElementoBase.setAttribute('type', 'text');

// asignamos al nuevo elemento un tipo texto

elElementoBase.setAttribute('size', '8');

// asignamos al nuevo elemento la propiedad draggable en true

elElementoBase.setAttribute('draggable', 'true');

// asignamos al nuevo elemento el eventListener dragstart

elElementoBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchstart para visualización mobile

elElementoBase.addEventListener('touchstart', tocar);

// le agregamos un cero al nombre del elemento base para indicar que es el primero

elElementoBase.id=nombreId+0;

// le asignamos al nuevo elemento el valor texto

elElementoBase.value='text';

// añadimos al contenedor de controles el nuevo elemento recién creado con id 0, indicando que es un elemento de base

elContenedorControles.appendChild(elElementoBase);

}

else if(nombreId == 'radio'){

// creamos un nuevo elemento en el DOM de tipo input

let elElementoBase = document.createElement('input');

// asignamos al nuevo elemento un tipo radio

elElementoBase.setAttribute('type', 'radio');

// asignamos al nuevo elemento un name que permite el agrupamiento de los radio botones

elElementoBase.setAttribute('name', 'grupoRadio');

// asignamos al nuevo elemento la propiedad draggable en true

elElementoBase.setAttribute('draggable', 'true');

// asignamos al nuevo elemento el eventListener dragstart

elElementoBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchstart para visualización mobile

elElementoBase.addEventListener('touchstart', tocar);

// le agregamos un cero al nombre del elemento base para indicar que es el primero

elElementoBase.id=nombreId+0;

// le asignamos al nuevo elemento el valor radio

elElementoBase.value='radio';

// añadimos al contenedor de controles el nuevo elemento recién creado con id 0, indicando que es un elemento de base

elContenedorControles.appendChild(elElementoBase);

}

else if(nombreId == 'label'){

// creamos un nuevo elemento en el DOM de tipo label

let elElementoBase = document.createElement('label');

// asignamos al nuevo elemento la propiedad draggable en true

elElementoBase.setAttribute('draggable', 'true');

// asignamos al nuevo elemento el eventListener dragstart

elElementoBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchstart para visualización mobile

elElementoBase.addEventListener('touchstart', tocar);

// le agregamos un cero al nombre del elemento base para indicar que es el primero

elElementoBase.id=nombreId+0;

// le asignamos al nuevo elemento el valor label

elElementoBase.value='label';

// le asignamos al nuevo elemento el texto de label

elElementoBase.innerText='label';

// añadimos al contenedor de controles el nuevo elemento recién creado con id 0, indicando que es un elemento de base

elContenedorControles.appendChild(elElementoBase);

}

else if(nombreId == 'checkbox'){

// creamos un nuevo elemento en el DOM de tipo input

let elElementoBase = document.createElement('input');

// asignamos al nuevo elemento un tipo checkbox

elElementoBase.setAttribute('type', 'checkbox');

// asignamos al nuevo elemento la propiedad draggable en true

elElementoBase.setAttribute('draggable', 'true');

// asignamos al nuevo elemento el eventListener dragstart

elElementoBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchstart para visualización mobile

elElementoBase.addEventListener('touchstart', tocar);

// le agregamos un cero al nombre del elemento base para indicar que es el primero

let newId = nombreId+0;

elElementoBase.id=newId;

// le asignamos al nuevo elemento el valor radio

elElementoBase.value='radio';

// asignamos al nuevo elemento el mismo name que el id para los checkbox

elElementoBase.setAttribute('name', newId);

// añadimos al contenedor de controles el nuevo elemento recién creado con id 0, indicando que es un elemento de base

elContenedorControles.appendChild(elElementoBase);

}

else if(nombreId == 'submit'){

// creamos un nuevo elemento en el DOM de tipo input

let elElementoBase = document.createElement('input');

// asignamos al nuevo elemento un tipo submit

elElementoBase.setAttribute('type', 'submit');

// asignamos al nuevo elemento la propiedad draggable en true

elElementoBase.setAttribute('draggable', 'true');

// asignamos al nuevo elemento el eventListener dragstart

elElementoBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchstart para visualización mobile

elElementoBase.addEventListener('touchstart', tocar)

// le agregamos un cero al nombre del elemento base para indicar que es el primero

elElementoBase.id=nombreId+0;

// le asignamos al nuevo elemento el valor submit

elElementoBase.value='submit';

// añadimos al contenedor de controles el nuevo elemento recién creado con id 0, indicando que es un elemento de base

elContenedorControles.appendChild(elElementoBase);

}

else if(nombreId == 'independiente'){

// creamos un nuevo elemento en el DOM de tipo input

let elElementoBase = document.createElement('input');

// asignamos al nuevo elemento un tipo texto

elElementoBase.setAttribute('type', 'text');

// asignamos al nuevo elemento un tipo texto

elElementoBase.setAttribute('size', '6');

// asignamos al nuevo elemento la propiedad draggable en true

elElementoBase.setAttribute('draggable', 'true');

// asignamos al nuevo elemento el eventListener dragstart

elElementoBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchstart para visualización mobile

elElementoBase.addEventListener('touchstart', tocar);

// le agregamos un cero al nombre del elemento base para indicar que es el primero

elElementoBase.id=nombreId+0;

// le asignamos al nuevo elemento el valor texto

elElementoBase.value='modificame';

// añadimos al contenedor de controles el nuevo elemento recién creado con id 0, indicando que es un elemento de base

elContenedorControles.appendChild(elElementoBase);

}

}

function resaltarCajaCodigo(){

// hacemos que el elemento que invoca la función (this) cambie su estilo

this.style.backgroundColor = 'lightblue';

}

function noResaltarCajaCodigo(){

// hacemos que el elemento que invoca la función (this) cambie su estilo

this.style.backgroundColor = 'lightcyan';

}

function cambiarElemento(idDivAEditar, idCajaEdita, id){

// console.log('cambiarElemento el id div a editar: ', idDivAEditar);

// console.log('el id caja que edita: ', idCajaEdita);

// console.log('cambiarElemento el id elemento editado: ', id);

// obtenemos el id registrado en la caja de span de posicion

// actualizamos el span de posicion

// evaluamos en bloque try catch porque es posible que se presenten errores al modificar en tiempo real

try{

let elIdSpanPosicion = 'posicion'+id;

let elSpanPosicion = document.getElementById(elIdSpanPosicion);

let idDivDestino = elSpanPosicion.innerText;

// creamos la variable del elemento a editar que se llenará de acuerdo con la condicion de si el div ha cambiado desde la primera vez o si se ha movido

let elDivElementoAEditar = '';

// comparamos el idDivAEditar(id original si no se ha movido la caja desde la primera vez de arrastrado) con el id que se encuentre en el span posicion idDivDestino

if( idDivAEditar == idDivDestino){

// buscamos en el dom el elemento editado, para cambiarlo (cambiamos realmente su div contenedor) dejando el idDivAEditar original (cuando se arrastró por primera vez)

elDivElementoAEditar = document.getElementById(idDivAEditar);

}

else{

// buscamos en el dom el elemento editado, para cambiarlo (cambiamos realmente su div contenedor) dejando el idDivDestino (cuando se arrastró multiples veces el elemento)

elDivElementoAEditar = document.getElementById(idDivDestino);

}

// asignamos al nuevo elemento el eventListener touchmove para visualización mobile

// elElementoAEditar.addEventListener('touchmove', arrastrar);

// buscamos en el dom la caja que esta editando el documento

let laCajaQueEdita = document.getElementById(idCajaEdita);

// console.log('la caja que edita: ',laCajaQueEdita);

// console.log('valor la caja que edita: ',laCajaQueEdita.value);

// console.log('el elemento a editar: ', elElementoAEditar);

// cambiamos el contenido del div a editar (divDestino) por el valor que contenga la caja que edita

elDivElementoAEditar.innerHTML =laCajaQueEdita.value;

// console.log('cambiarElemento elDivElementoAEditar: ', elDivElementoAEditar);

// ubicamos en el DOM el elemento a editar

let elElementoAEditar = document.getElementById(id);

// el evento dragstart tendrá asociado la función de callback de arrastrar. Es necesario asociar de nuevo el event listener porque se pierde al hacer cambios al elemento dinámicamente

elElementoAEditar.addEventListener('dragstart', arrastrar);

// el evento touchstart tendrá asociado la función de callback de tocar. Es necesario asociar de nuevo el event listener porque se pierde al hacer cambios al elemento dinámicamente

elElementoAEditar.addEventListener('touchstart', tocar);

// console.log(elElementoAEditar);

}

catch(error){

console.log(`Es posible que se presenten errores porque permitimos modificación en tiempo real. ERROR: ${error}`);

}

}

// la funcion recibe el elemento que será editado, y su id, y el id del Div destino para modificarle su innerHTML

function enviarCodigoHTML(elemento, id, idDivDestino){

// console.log(elemento);

// console.log(typeof elemento);

// console.log('enviarCodigoHTML idDivDestino:', idDivDestino);

// Extraemos el HTML del elemento arrastrado con la propiedad outerHTML y la convertimos en String

let htmlExtraido = String(elemento.outerHTML);

// console.log('htmlExtraido',htmlExtraido);

// ubicamos la caja de código en el DOM

let laCajaDeCodigo = document.getElementById('cajaDeCodigo');

// identificamos el ancho de la pantalla para determinar el size de la caja que se usará para escribir código pensando en móviles

let anchoPantalla = window.innerWidth;

let sizeCajaCodigo = '';

if(anchoPantalla <= 768){

// es pantalla pequeña

sizeCajaCodigo = '30';

}

else{

// es pantalla grande

sizeCajaCodigo = '40';

}

// creamos una nueva caja de texto que poseerá el código recién creado

let nuevaCajaTexto = document.createElement('input');

// establecemos los atributos de la nueva caja de texto indicando que es de tipo texto

nuevaCajaTexto.setAttribute('type', 'text');

// establecemos los atributos de la nueva caja de texto para que tengan un tamaño definido de 40 caracteres

nuevaCajaTexto.setAttribute('size', sizeCajaCodigo);

// establecemos los atributos de la nueva caja de texto para que no tengan bordes aplicando una clase personalizda

nuevaCajaTexto.setAttribute('class', 'sin-bordes');

// generamos un id sintetico para la caja, de manera que se pueda ubicar en el DOM

let idSinteticoCajaEdicion = 'editar'+id;

// creamos una caja de span que identifique la posición del elemento html que se está arrastrando

let nuevoSpanPosicion = document.createElement('span');

let idSinteticoSpanPosicion = 'posicion'+id;

// asignamos el id sintetico al nuevo parrafo generado

nuevoSpanPosicion.id = idSinteticoSpanPosicion;

// asignamos un fontSize de 7 a la posicion

nuevoSpanPosicion.style.fontSize = '9px';

nuevoSpanPosicion.innerText = idDivDestino;

// asignamos el id sintetico a la nueva caja generada

nuevaCajaTexto.id = idSinteticoCajaEdicion;

// le asignamos un event listener para que resalte con un color azul claro la caja que se esta editando

nuevaCajaTexto.addEventListener('mouseover', resaltarCajaCodigo );

// le asignamos un event listener para que deje de resaltar la caja que se esta editando cuando se salga del foco

nuevaCajaTexto.addEventListener('mouseout', noResaltarCajaCodigo );

// hacemos que todas las cajas de texto reaccionen al evento onkeyup, es decir que reaccione cuando se escriba en las cajas pasando como argumentos el id del elemento que se va a editar y el id de la caja que edita

// cuando se requiere llamar una función de callback con parámetros, se llama una función anonima que internamente llama a la función de callback con parámetros

nuevaCajaTexto.addEventListener('keyup',function() { cambiarElemento(idDivDestino, idSinteticoCajaEdicion, id);} );

//asignamos como valor en la caja de código el html extraido

nuevaCajaTexto.value = htmlExtraido;

// creamos un nuevo salto de linea

let nuevoSalto = document.createElement('br');

// añadimos a la caja de codigo el nuevo elemento recién creado de span

laCajaDeCodigo.appendChild(nuevoSpanPosicion);

// añadimos a la caja de codigo el nuevo elemento recién creado

laCajaDeCodigo.appendChild(nuevaCajaTexto);

// añadimos a la caja de código el nuevo salto de linea

laCajaDeCodigo.appendChild(nuevoSalto);

}

function soltar(ev) {

// console.log(ev);

// console.log(ev.srcElement.firstElementChild.outerHTML);

// console.log(ev.srcElement.lastChild);

ev.preventDefault();

// borramos el borde de ubicación cuando se suelte el elemento

this.style.borderStyle = 'none';

try{

// si la cantidad de elementos es menor que el límite, se pueden seguir arrastrando más elementos

if(cantidadElementos < limiteElementos ){

// en el manejador soltar obtenemos el id del elemento que está siendo arrastrado y lo usamos para moverlo al contenedor destino

// data es el elemento que está siendo arrastrado

let data = ev.dataTransfer.getData("text");

// console.log('soltar data: ',data);

// console.log('soltar ev.target: ',ev.target);

// console.log('soltar ultimaTargeId: ', ultimoTargetId);

// evaluamos si data esta vacio. Si esta vacio es porque se ha editado el elemento

if(data == ''){

// console.log('soltar data esta vacio');

// puede que data este vacío por haber editado el elemento, así que lo reconstruimos

// El objeto DataTransfer no se puede modificar asi como en la siguiente linea

// ev.dataTransfer.setData("text/plain", ev.target.id);

// hacemos data igual al valor almacenado en el ultimoTargetId

data = ultimoTargetId;

// agregamos (appendChild) al target(destino) el elemento identificado mediante su id y referenciado en la variable data, que al estar vacia, enviamos el ultimo targetId

ev.target.appendChild(document.getElementById(data));

}

else{

// agregamos (appendChild) al target(destino) el elemento identificado mediante su id y referenciado en la variable data

ev.target.appendChild(document.getElementById(data));

}

// identificamos el destino (target), especificamente su id

let idDivDestino = ev.target.id;

// console.log('soltar idDivDestino: ',idDivDestino);

// invocamos al método generarNuevoId para que decida si se genera nuevo id y se reubique el elemento base en la sección de controles

generarNuevoId(data, idDivDestino);

}else{

// si la cantidad de elementos es mayor al limite mostramos alerta

alert('No puede arrastrar más elementos. Ha superado el límite');

}

}

catch(error){

console.log('error en la función arrastrar atrapado: ',error); // pasa el objeto de la excepción al manejador de errores

}

}

function soltarParaEliminar(ev) {

// console.log(ev);

// console.log(ev.srcElement.firstElementChild.outerHTML);

// console.log(ev.srcElement.lastChild);

ev.preventDefault();

// en el manejador soltar obtenemos el id del elemento que está siendo arrastrado y lo usamos para eliminarlo del DOM

let data = ev.dataTransfer.getData("text");

// console.log(data);

// extraemos el ultimo elemento del idAnterior, pues viene en formato letraNumero

let oldIdString = data.slice(-1);

// si el ultimo elemento del id es diferente de cero, eliminamos los elementos del lienzo, y de los arreglos

// de lo contrario no hacemos nada, pues si el ultimo elemento del id es cero, significa que es el elemento de base

// y no queremos eliminar el elemento de base

if(oldIdString != 0){

let elElementoAeliminar = document.getElementById(data);

// eliminamos(remove) del source(origen) el elemento identificado mediante su id y referenciado en la variable data

elElementoAeliminar.remove();

//eliminamos del arreglo correspondiente el elemento arrastrado

EliminarDelArreglo(data);

//eliminamos de la caja de codigo el elemento arrastrado

EliminarDeLaCajaCodigo(data);

}

}

function tocarParaEliminar(idElementoAEliminar) {

// en el manejador tocar obtenemos el id del elemento que está siendo tocado y lo usamos para eliminarlo del DOM

// console.log('tocarParaEliminar: idElementoEliminar',idElementoAEliminar);

// extraemos el ultimo elemento del idElementoAEliminar, pues viene en formato letraNumero

let oldIdString = idElementoAEliminar.slice(-1);

// si el ultimo elemento del id es diferente de cero, eliminamos los elementos del lienzo, y de los arreglos

// de lo contrario no hacemos nada, pues si el ultimo elemento del id es cero, significa que es el elemento de base

// y no queremos eliminar el elemento de base

if(oldIdString != 0){

// validamos que el idElementoAEliminar no esté vacio o nulo por alguna razón

if(idElementoAEliminar != null){

// realizamos las operaciones de eliminar si los elementos no están vacios o nulos

let elElementoAeliminar = document.getElementById(idElementoAEliminar);

if(elElementoAeliminar != null){

// hacemos una segunda validacion pues puede que el idElementoAEliminar tenga contenido, pero en el DOM no exista y esté null

// eliminamos(remove) del source(origen) el elemento identificado mediante su id

elElementoAeliminar.remove();

//eliminamos del arreglo correspondiente el elemento tocado

EliminarDelArreglo(idElementoAEliminar);

//eliminamos de la caja de codigo el elemento tocado

EliminarDeLaCajaCodigo(idElementoAEliminar);

}// fin elElementoAEliminar != null

}// fin idElementoEliminar null

}// fin oldIdString

}

function dentroContenedor(){

// cambiar las clases de bootstrap causa que los elementos se oculten en los div

// this.classList.remove('bg-warning');

// this.classList.add('bg-success');

// this.style.backgroundColor = 'lightBlue';

// mostramos borde punteado cuando se posiciona el elemento arrastrado sobre un div del grid system

this.style.borderStyle = 'dotted';

}

function fueraContenedor(){

// cambiar las clases de bootstrap causa que los elementos se oculten en los div

// this.classList.remove('bg-success');

// this.classList.add('bg-warning');

// this.style.backgroundColor = 'yellow';

// borramos el borde punteado después de salir del div del gri system

this.style.borderStyle = 'none';

}

function borrarTodosLosArreglos(){

// Establecemos todos los arreglos componentes con sus valores iniciales

// arreglo global para llevar registro de botones

arrBotones = ['button0'];

// arreglo global para llevar registro de cajas de texto

arrTexto = ['text0'];

// arreglo global para llevar registro de radio botones

arrRadio = ['radio0'];

// arreglo global para llevar registro de labels

arrLabel = ['label0'];

// arreglo global para llevar registro de checkbox

arrCheckbox = ['checkbox0'];

// arreglo global para llevar registro de submit

arrSubmit = ['submit0'];

// arreglo global para llevar registro de independiente

arrIndependiente = ['independiente0'];

// borramos la caja de código para recibir nuevos elementos

// ubicamos la caja de código en el DOM

let laCajaDeCodigo = document.getElementById('cajaDeCodigo');

laCajaDeCodigo.innerText = '';

// ubicamos las variables de cantidad de elementos y limites para reiniciarlos

limiteElementos = 0;

cantidadElementos = 0;

}

function generarLienzo12x12(){

// console.log(lienzo12x12);

// Borramos todos los arreglos para que comience de nuevo el conteo de elementos

// que sea lo primero para que luego se setee el limite

borrarTodosLosArreglos();

// establecemos el límite de elementos a arrastrar

limiteElementos = 144;

// console.log('generarLienzo12x12 limiteElementos: ', limiteElementos);

// mostramos la seccion de controles

habilitarSeccionControles();

// buscamos el lienzo padre en el DOM

let elLienzoPadre = document.getElementById('lienzoPadre');

// enviamos al lienzo padre la variable global con el html del lienzo 12x12

elLienzoPadre.innerHTML = lienzo12x12;

//Asignamos los eventos para el lienzo, de manera que se pueda arrastrar y soltar sobre ellos

// Se debe realizar dentro de este método, porque si lo hacemos en el onload, al no existir los elementos, no se asignará nada y no servirán para arrastrar y soltar

//buscamos los doce contenedores en el DOM donde se puede arrastrar el elemento

let arregloContenedores = document.getElementsByClassName('contenedoresArrastraSuelta');

// recorremos los doce contenedores y les asignamos los eventos dragover y drop

// el evento dragover tendrá asociado la función de callback de permitirSoltar

// el evento drop tendrá asociado la función de callback de soltar

for(let i=0; i<arregloContenedores.length;i++){

arregloContenedores[i].addEventListener('dragover', permitirSoltar);

arregloContenedores[i].addEventListener('drop', soltar);

// evento para tocar en version mobile para cada div generado dinamicamente

arregloContenedores[i].addEventListener('touchstart', tocar);

/\* No funciona bien cambiar colores de los div en los eventos mouseover y mouseout\*/

arregloContenedores[i].addEventListener('dragenter', dentroContenedor);

arregloContenedores[i].addEventListener('dragleave', fueraContenedor);

}

}

function generarLienzo2x12(){

// console.log(lienzo2x12);

// Borramos todos los arreglos para que comience de nuevo el conteo de elementos

// que sea lo primero para que luego se setee el limite

borrarTodosLosArreglos();

// establecemos el límite de elementos a arrastrar

limiteElementos = 24;

// console.log('generarLienzo2x12 limiteElementos: ', limiteElementos);

// mostramos la seccion de controles

habilitarSeccionControles();

// buscamos el lienzo padre en el DOM

let elLienzoPadre = document.getElementById('lienzoPadre');

// enviamos al lienzo padre la variable global con el html del lienzo 2x12de

elLienzoPadre.innerHTML = lienzo2x12;

//Asignamos los eventos para el lienzo, de manera que se pueda arrastrar y soltar sobre ellos

// Se debe realizar dentro de este método, porque si lo hacemos en el onload, al no existir los elementos, no se asignará nada y no servirán para arrastrar y soltar

//buscamos los doce contenedores en el DOM donde se puede arrastrar el elemento

let arregloContenedores = document.getElementsByClassName('contenedoresArrastraSuelta');

// recorremos los doce contenedores y les asignamos los eventos dragover y drop

// el evento dragover tendrá asociado la función de callback de permitirSoltar

// el evento drop tendrá asociado la función de callback de soltar

for(let i=0; i<arregloContenedores.length;i++){

arregloContenedores[i].addEventListener('dragover', permitirSoltar);

arregloContenedores[i].addEventListener('drop', soltar);

// evento para tocar en version mobile para cada div generado dinamicamente

arregloContenedores[i].addEventListener('touchstart', tocar);

/\* No funciona bien cambiar colores de los div en los eventos mouseover y mouseout\*/

arregloContenedores[i].addEventListener('dragenter', dentroContenedor);

arregloContenedores[i].addEventListener('dragleave', fueraContenedor);

}

}

function generarLienzo3x12(){

// console.log(lienzo3x12);

// Borramos todos los arreglos para que comience de nuevo el conteo de elementos

// que sea lo primero para que luego se setee el limite

borrarTodosLosArreglos();

// establecemos el límite de elementos a arrastrar

limiteElementos = 36;

// console.log('generarLienzo3x12 limiteElementos: ', limiteElementos);

// mostramos la seccion de controles

habilitarSeccionControles();

// buscamos el lienzo padre en el DOM

let elLienzoPadre = document.getElementById('lienzoPadre');

// enviamos al lienzo padre la variable global con el html del lienzo 2x12de

elLienzoPadre.innerHTML = lienzo3x12;

//Asignamos los eventos para el lienzo, de manera que se pueda arrastrar y soltar sobre ellos

// Se debe realizar dentro de este método, porque si lo hacemos en el onload, al no existir los elementos, no se asignará nada y no servirán para arrastrar y soltar

//buscamos los doce contenedores en el DOM donde se puede arrastrar el elemento

let arregloContenedores = document.getElementsByClassName('contenedoresArrastraSuelta');

// recorremos los doce contenedores y les asignamos los eventos dragover y drop

// el evento dragover tendrá asociado la función de callback de permitirSoltar

// el evento drop tendrá asociado la función de callback de soltar

for(let i=0; i<arregloContenedores.length;i++){

arregloContenedores[i].addEventListener('dragover', permitirSoltar);

arregloContenedores[i].addEventListener('drop', soltar);

// evento para tocar en version mobile para cada div generado dinamicamente

arregloContenedores[i].addEventListener('touchstart', tocar);

/\* No funciona bien cambiar colores de los div en los eventos mouseover y mouseout\*/

arregloContenedores[i].addEventListener('dragenter', dentroContenedor);

arregloContenedores[i].addEventListener('dragleave', fueraContenedor);

}

}

function deshabilitarSeccionControles(){

// buscamos el contenedor de controles

let elContenedorControles = document.getElementById('contenedorControles');

// ocultamos la seccion de controles

elContenedorControles.hidden = true;

}

function habilitarSeccionControles(){

// buscamos el contenedor de controles

let elContenedorControles = document.getElementById('contenedorControles');

// mostramos la seccion de controles

elContenedorControles.hidden = false;

}

function tocar(ev){

// console.log('tocar ev: ', ev);

// definimos lógica para mover elementos por pares

//si el contadorTouch está en cero, incrementamos uno y guardamos el primer elemento tocado

if(contadorTouch == 0){

// incrementamos en uno el contador touch

contadorTouch+=1;

// guardamos el elemento tocado en la variable elementoTouch1

elementoTouch1 = ev.target.id;

} else if( contadorTouch == 1){

// regresamos el contadorTouch a cero, para que al hacer click en otro elemento comience de nuevo el proceso

contadorTouch=0;

// guardamos el elemento tocado en la variable elementoTouch2

elementoTouch2 = ev.target.id;

}

// console.log('tocar elementoTouch1: ', elementoTouch1);

// console.log('tocar elementoTouch2: ', elementoTouch2);

// logica para identificar el elemento y el div

// extraemos el primer caracter del elementoTouch1, pues si es div, debe comenzar con la letra f

// slice extrae desde la posicion cero un caracter

let primerCaracterTouch1 = elementoTouch1.slice(0,1);

// console.log('tocar primerCaracterTouch1', primerCaracterTouch1);

// extraemos el primer caracter del elementoTouch1, pues si es div, debe comenzar con la letra f

// slice extrae desde la posicion cero un caracter

let primerCaracterTouch2 = elementoTouch2.slice(0,1);

// console.log('tocar primerCaracterTouch2', primerCaracterTouch2);

// variables para determinar el idDiDestino y el elemento a mover

let idDivDestino = '';

let idElementoAMover = '';

let autorizaMover = false;

// evaluamos si el primer caracter es una f y el segundo es diferente de f para identificar origen y destino

// por f comienzan los div

// diferente de f pueden ser controles

// si comienza por c es el contenedor de controles

if(primerCaracterTouch1 == 'f' && primerCaracterTouch2 != 'f' ){

// validamos si el segundo caracter es diferente de c, para descartar que sea el div de controles

if(primerCaracterTouch2 != 'c'){

// identificamos el destino (target), especificamente su id

idDivDestino = elementoTouch1;

// console.log('tocar el div de destino es : ', idDivDestino);

// identificamos el elemento a mover, especificamente su id

idElementoAMover = elementoTouch2;

// console.log('tocar el elemento a mover es : ', idElementoAMover);

//cambiamos el flag de autoriza mover

autorizaMover = true;

}

else{

//cambiamos el flag de autoriza mover pues una de las cajas seleccionadas era la de controles

autorizaMover = false;

}

}

else if(primerCaracterTouch1 != 'f' && primerCaracterTouch2 == 'f'){

// validamos si el primer caracter es diferente de c, para descartar que sea el div de controles

if(primerCaracterTouch1 != 'c'){

// identificamos el destino (target), especificamente su id

idDivDestino = elementoTouch2;

// console.log('tocar el div de destino es : ', idDivDestino);

// identificamos el elemento a mover, especificamente su id

idElementoAMover = elementoTouch1;

// console.log('tocar el elemento a mover es : ', idElementoAMover);

//cambiamos el flag de autoriza mover

autorizaMover = true;

}

else{

//cambiamos el flag de autoriza mover pues una de las cajas seleccionadas era la de controles

autorizaMover = false;

}

}

else if(primerCaracterTouch1 == 'c' && primerCaracterTouch2 != 'f'){

// si el primer caracter es c , validamos si el segundo es diferente de c, porque puede que se haya presionado el contenedor de controles dos veces

if(primerCaracterTouch2 != 'c'){

// si el segundo caracter es diferente de c significa que se quiere eliminar un control

// console.log('Elemento a eliminar identificado es: ', elementoTouch2);

tocarParaEliminar(elementoTouch2);

//cambiamos el flag de autoriza mover pues una de las cajas seleccionadas era la de controles

autorizaMover = false;

}

}

else if(primerCaracterTouch1 != 'f' && primerCaracterTouch2 == 'c'){

// si el segundo caracter es c , validamos si el primero es diferente de c, porque puede que se haya presionado el contenedor de controles dos veces

if(primerCaracterTouch1 != 'c'){

// si el primer caracter es diferente de c significa que se quiere eliminar un control

// console.log('Elemento a eliminar identificado es: ', elementoTouch1);

tocarParaEliminar(elementoTouch1);

//cambiamos el flag de autoriza mover pues una de las cajas seleccionadas era la de controles

autorizaMover = false;

}

}

else{

// console.log('tocar ambos elementos son divs o ambos son controles');

//mantenemos el flag de autoriza mover en false

autorizaMover = false;

}

// si autoriza mover está en true, permitimos trasladar el elemento

if( autorizaMover){

// validamos que los idDivDestino y los elementos a mover idElementoAMover no estén vacios o nulos

if(idDivDestino != null){

if(idElementoAMover != null){

// identificamos el div destino

let elDivDestino = document.getElementById(idDivDestino);

// identificamos el elemento a mover

let elElementoAMover = document.getElementById(idElementoAMover);

// realizamos una nueva validación, pues puede que los id tengan contenido, pero los elementos no existan en el DOM

if(elDivDestino != null && elElementoAMover !=null){

// agregamos (appendChild) al target(destino) el elemento a mover identificado mediante su id

elDivDestino.appendChild(elElementoAMover);

// invocamos al método generarNuevoId para que decida si se genera nuevo id y se reubique el elemento base en la sección de controles

generarNuevoId(idElementoAMover, idDivDestino);

}

}

}

}// fin autoriza mover

// identificamos el destino (target), especificamente su id

idDivDestino = ev.target.id;

// console.log('soltar idDivDestino: ',idDivDestino);

}

function asignarEventos(){

// buscamos el contenedor de controles para permitir eliminar elementos que regresen de vuelta

let elContenedorControles = document.getElementById('contenedorControles');

elContenedorControles.addEventListener('dragover', permitirSoltar);

elContenedorControles.addEventListener('drop', soltarParaEliminar);

// hacemos que el contenedor de controles reaccione al evento tocar

elContenedorControles.addEventListener('touchstart', tocar);

// buscamos el botón de base en el DOM

let elBotonBase = document.getElementById('button0');

// el evento dragstart tendrá asociado la función de callback de arrastrar

elBotonBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchstart para visualización mobile

elBotonBase.addEventListener('touchstart', tocar);

// buscamos la caja de texto de base en el DOM

let elTextoBase = document.getElementById('text0');

// el evento dragstart tendrá asociado la función de callback de arrastrar

elTextoBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchmove para visualización mobile

elTextoBase.addEventListener('touchstart', tocar);

// buscamos el radio boton de base en el DOM

let elRadioBase = document.getElementById('radio0');

// el evento dragstart tendrá asociado la función de callback de arrastrar

elRadioBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchmove para visualización mobile

elRadioBase.addEventListener('touchstart', tocar);

// buscamos el label de base en el DOM

let elLabelBase = document.getElementById('label0');

// el evento dragstart tendrá asociado la función de callback de arrastrar

elLabelBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchmove para visualización mobile

elLabelBase.addEventListener('touchstart', tocar);

// buscamos el checkbox de base en el DOM

let elCheckBase = document.getElementById('checkbox0');

// el evento dragstart tendrá asociado la función de callback de arrastrar

elCheckBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchmove para visualización mobile

elCheckBase.addEventListener('touchstart', tocar);

// buscamos el submit de base en el DOM

let elSubmitBase = document.getElementById('submit0');

// el evento dragstart tendrá asociado la función de callback de arrastrar

elSubmitBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchmove para visualización mobile

elSubmitBase.addEventListener('touchstart', tocar);

// buscamos el independiente de base en el DOM

let elIndependienteBase = document.getElementById('independiente0');

// el evento dragstart tendrá asociado la función de callback de arrastrar

elIndependienteBase.addEventListener('dragstart', arrastrar);

// asignamos al nuevo elemento el eventListener touchmove para visualización mobile

elIndependienteBase.addEventListener('touchstart', tocar);

// buscamos el boton de 12x12

let elBoton12x12 = document.getElementById('btn12x12');

elBoton12x12.addEventListener('click', generarLienzo12x12);

// buscamos el boton de 2x12

let elBoton2x12 = document.getElementById('btn2x12');

elBoton2x12.addEventListener('click', generarLienzo2x12);

// buscamos el boton de 3x12

let elBoton3x12 = document.getElementById('btn3x12');

elBoton3x12.addEventListener('click', generarLienzo3x12);

// invocamos la función de ocultar controles. No se muestran hasta que se seleccione un layout

deshabilitarSeccionControles();

}

var lienzo12x12 = `

<!-- Contenedor de 12 columnas por 12 filas para arrastrar elementos-->

<div class="container">

<!-- fila 1-->

<div class="row">

<div id="f1c1" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f1c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c4" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f1c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 2-->

<div class="row">

<div id="f2c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c2" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f2c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c4" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f2c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 3-->

<div class="row">

<div id="f3c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c3" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f3c4" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f3c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 4-->

<div class="row">

<div id="f4c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c4" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f4c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f4c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 5-->

<div class="row">

<div id="f5c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c4" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f5c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f5c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 6-->

<div class="row">

<div id="f6c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c4" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f6c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f6c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 7-->

<div class="row">

<div id="f7c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c4" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f7c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f7c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 8-->

<div class="row">

<div id="f8c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c4" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f8c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f8c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 9-->

<div class="row">

<div id="f9c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c4" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f9c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c19" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f9c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 10-->

<div class="row">

<div id="f10c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c4" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f10c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f10c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 11-->

<div class="row">

<div id="f11c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c4" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f11c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f11c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 12-->

<div class="row">

<div id="f12c1" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c2" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c3" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c4" class="col-md-1 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f12c5" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c6" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c7" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c8" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c9" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c10" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c11" class="col-md-1 contenedoresArrastraSuelta">

</div>

<div id="f12c12" class="col-md-1 contenedoresArrastraSuelta">

</div>

</div>

</div><!-- fin container-->

`;

var lienzo2x12 = `

<!-- Contenedor de 2 columnas por 12 filas para arrastrar elementos-->

<div class="container">

<!-- fila 1-->

<div class="row">

<div id="f1c1" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f1c2" class="col-md-6 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 2-->

<div class="row">

<div id="f2c1" class="col-md-6 contenedoresArrastraSuelta">

</div>

<div id="f2c2" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

</div>

<!-- fila 3-->

<div class="row">

<div id="f3c1" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f3c2" class="col-md-6 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 4-->

<div class="row">

<div id="f4c1" class="col-md-6 contenedoresArrastraSuelta">

</div>

<div id="f4c2" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

</div>

<!-- fila 5-->

<div class="row">

<div id="f5c1" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f5c2" class="col-md-6 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 6-->

<div class="row">

<div id="f6c1" class="col-md-6 contenedoresArrastraSuelta">

</div>

<div id="f6c2" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

</div>

<!-- fila 7-->

<div class="row">

<div id="f7c1" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f7c2" class="col-md-6 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 8-->

<div class="row">

<div id="f8c1" class="col-md-6 contenedoresArrastraSuelta">

</div>

<div id="f8c2" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

</div>

<!-- fila 9-->

<div class="row">

<div id="f9c1" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f9c2" class="col-md-6 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 10-->

<div class="row">

<div id="f10c1" class="col-md-6 contenedoresArrastraSuelta">

</div>

<div id="f10c2" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

</div>

<!-- fila 11-->

<div class="row">

<div id="f11c1" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f11c2" class="col-md-6 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 12-->

<div class="row">

<div id="f12c1" class="col-md-6 contenedoresArrastraSuelta">

</div>

<div id="f12c2" class="col-md-6 contenedoresArrastraSuelta">

&nbsp;

</div>

</div>

</div><!-- fin container-->

`;

var lienzo3x12 = `

<!-- Contenedor de 3 columnas por 12 filas para arrastrar elementos-->

<div class="container">

<!-- fila 1-->

<div class="row">

<div id="f1c1" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f1c2" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f1c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 2-->

<div class="row">

<div id="f2c1" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f2c2" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f2c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 3-->

<div class="row">

<div id="f3c1" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f3c2" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f3c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 4-->

<div class="row">

<div id="f4c1" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f4c2" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f4c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 5-->

<div class="row">

<div id="f5c1" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f5c2" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f5c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 6-->

<div class="row">

<div id="f6c1" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f6c2" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f6c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 7-->

<div class="row">

<div id="f7c1" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f7c2" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f7c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 8-->

<div class="row">

<div id="f8c1" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f8c2" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f8c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 9-->

<div class="row">

<div id="f9c1" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f9c2" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f9c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 10-->

<div class="row">

<div id="f10c1" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f10c2" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f10c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 11-->

<div class="row">

<div id="f11c1" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f11c2" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f11c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

<!-- fila 12-->

<div class="row">

<div id="f12c1" class="col-md-4 contenedoresArrastraSuelta">

</div>

<div id="f12c2" class="col-md-4 contenedoresArrastraSuelta">

&nbsp;

</div>

<div id="f12c3" class="col-md-4 contenedoresArrastraSuelta">

</div>

</div>

</div><!-- fin container-->

`;