Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

Laboratorio 7

[Construcción de un cliente 'grueso' con un API REST, HTML5, Javascript y CSS3. Parte II.](https://github.com/ARSW2023-2/ARSW_Lab07#construci%C3%B3n-de-un-cliente-grueso-con-un-api-rest-html5-javascript-y-css3-parte-ii)

url repositorio:

<https://github.com/ARSW2023-2/ARSW_Lab07.git>

Integrantes:

Luisa Fernanda Bermúdez Girón

Karol Daniela Ladino Ladino

Squad:

Inside Out

Profesor:

Javier Iván Toquica Barrera

Curso:

ARSW – 1

Fecha De Entrega:

27-10-2023

1. **Introducción**

En el contexto del desarrollo de aplicaciones web interactivas, la captura de eventos del usuario es esencial para proporcionar una experiencia de usuario atractiva y funcional. En este laboratorio, se aborda la implementación de un manejador de eventos en un canvas de una página web, permitiendo la captura de 'clicks' realizados tanto a través del mouse como a través de una pantalla táctil. Este proceso se realiza teniendo en cuenta el uso de eventos de tipo 'PointerEvent', que, aunque aún no son compatibles con todos los navegadores, ofrecen una funcionalidad valiosa para el desarrollo interactivo en la web.

El objetivo principal de este laboratorio es modularizar adecuadamente la inicialización de los manejadores de eventos, siguiendo un enfoque más organizado y eficiente, como se ilustra en el ejemplo proporcionado en el codepen. La modularización permite un código más mantenible y escalable, lo que es crucial en proyectos web de cualquier envergadura.

Además de la captura de eventos, este laboratorio aborda la gestión de puntos en un canvas, permitiendo que los nuevos puntos se agreguen a la secuencia de puntos del canvas actual, pero solo en la memoria de la aplicación, sin afectar el API. También se aborda la implementación de botones como "Save/Update" y "Create new blueprint", que realizan peticiones PUT y POST al API para actualizar o crear planos, respectivamente. Estas operaciones se realizan de manera asincrónica y ordenada gracias al uso de promesas en JavaScript.

El laboratorio también incluye la implementación de un botón "DELETE" que permite borrar el canvas, eliminar un recurso correspondiente en el servidor y obtener la lista actualizada de planos disponibles. A lo largo de este proceso, se enfatiza la importancia de manejar las operaciones basadas en callbacks en un orden específico para garantizar un flujo de trabajo coherente.

1. **Desarrollo del laboratorio**
2. Agregue al canvas de la página un manejador de eventos que permita capturar los 'clicks' realizados, bien sea a través del mouse, o a través de una pantalla táctil. Para esto, tenga en cuenta [este ejemplo de uso de los eventos de tipo 'PointerEvent'](https://mobiforge.com/design-development/html5-pointer-events-api-combining-touch-mouse-and-pen) (aún no soportado por todos los navegadores) para este fin. Recuerde que a diferencia del ejemplo anterior (donde el código JS está incrustado en la vista), se espera tener la inicialización de los manejadores de eventos correctamente modularizado, tal [como se muestra en este codepen](https://codepen.io/hcadavid/pen/BwWbrw).

* Modificación index.htmlCaptura de pantalla de computadora

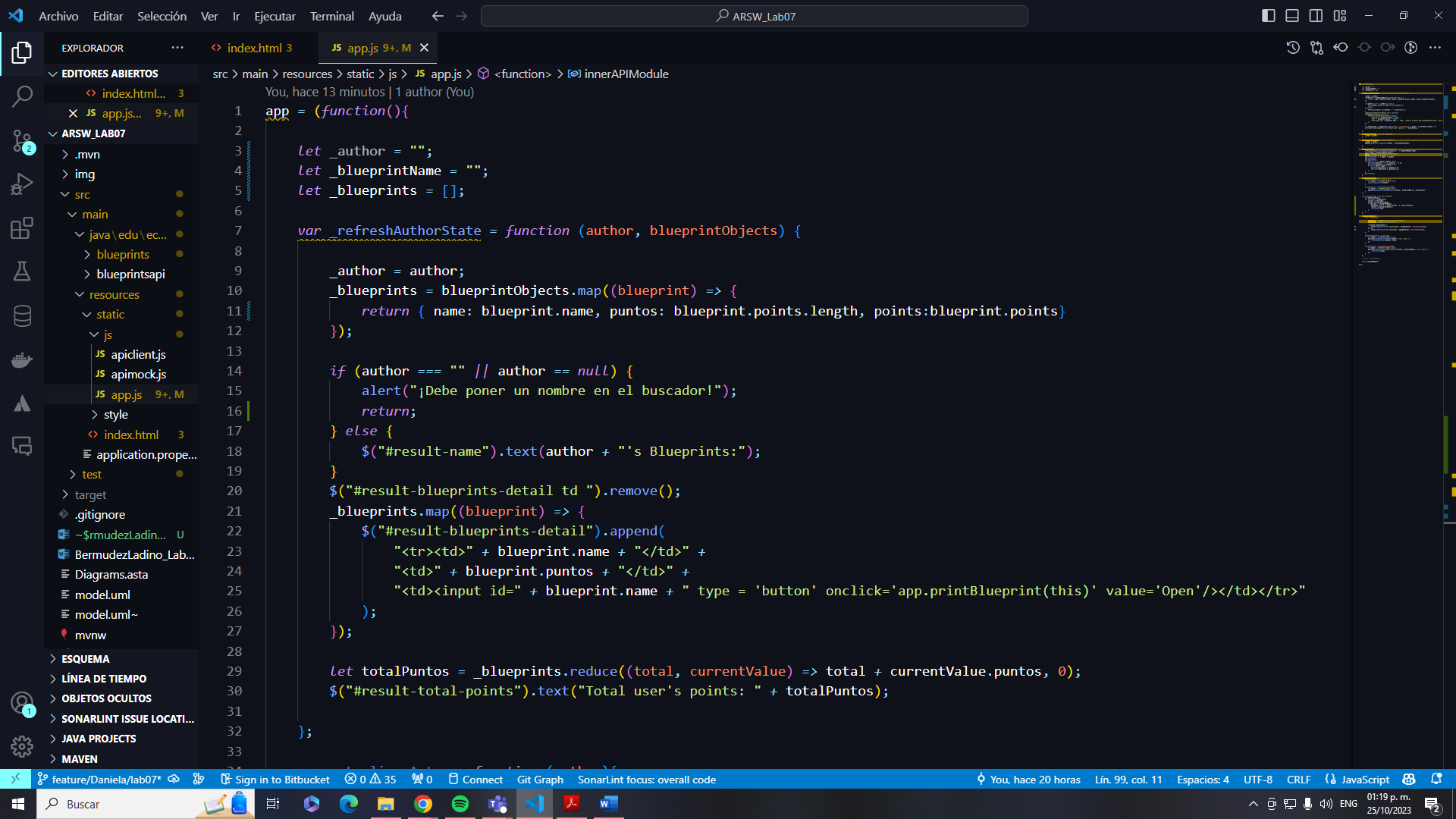
  Descripción generada automáticamente
* Modificación app.jsCaptura de pantalla de computadora

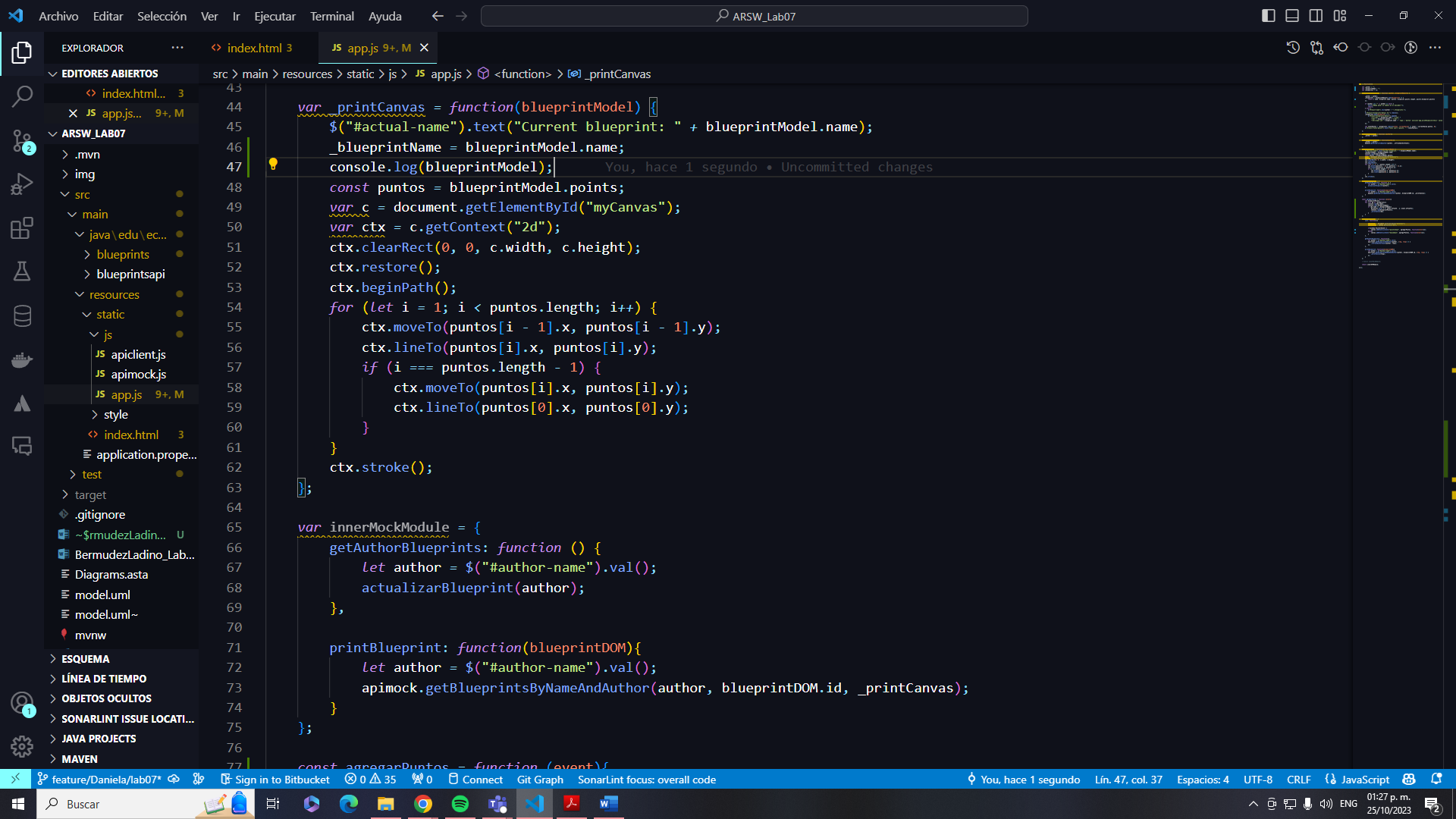
  Descripción generada automáticamente
* Resultados ejecución:Interfaz de usuario gráfica, Aplicación

  Descripción generada automáticamente

1. Agregue lo que haga falta en sus módulos para que cuando se capturen nuevos puntos en el canvas abierto (si no se ha seleccionado un canvas NO se debe hacer nada):
   1. Se agregue el punto al final de la secuencia de puntos del canvas actual (sólo en la memoria de la aplicación, AÚN NO EN EL API!).

* Modificación app.js





Captura de pantalla de computadora

Descripción generada automáticamente

* 1. Se repinte el dibujo.Texto

     Descripción generada automáticamente con confianza media

1. Agregue el botón Save/Update. Respetando la arquitectura de módulos actual del cliente, haga que al oprimirse el botón:
   1. Se haga PUT al API, con el plano actualizado, en su recurso REST correspondiente.
   2. Se haga GET al recurso /blueprints, para obtener de nuevo todos los planos realizados.
   3. Se calculen nuevamente los puntos totales del usuario.

Para lo anterior tenga en cuenta:

* jQuery no tiene funciones para peticiones PUT o DELETE, por lo que es necesario 'configurarlas' manualmente a través de su API para AJAX. Por ejemplo, para hacer una peticion PUT a un recurso /myrecurso:

Texto, Carta

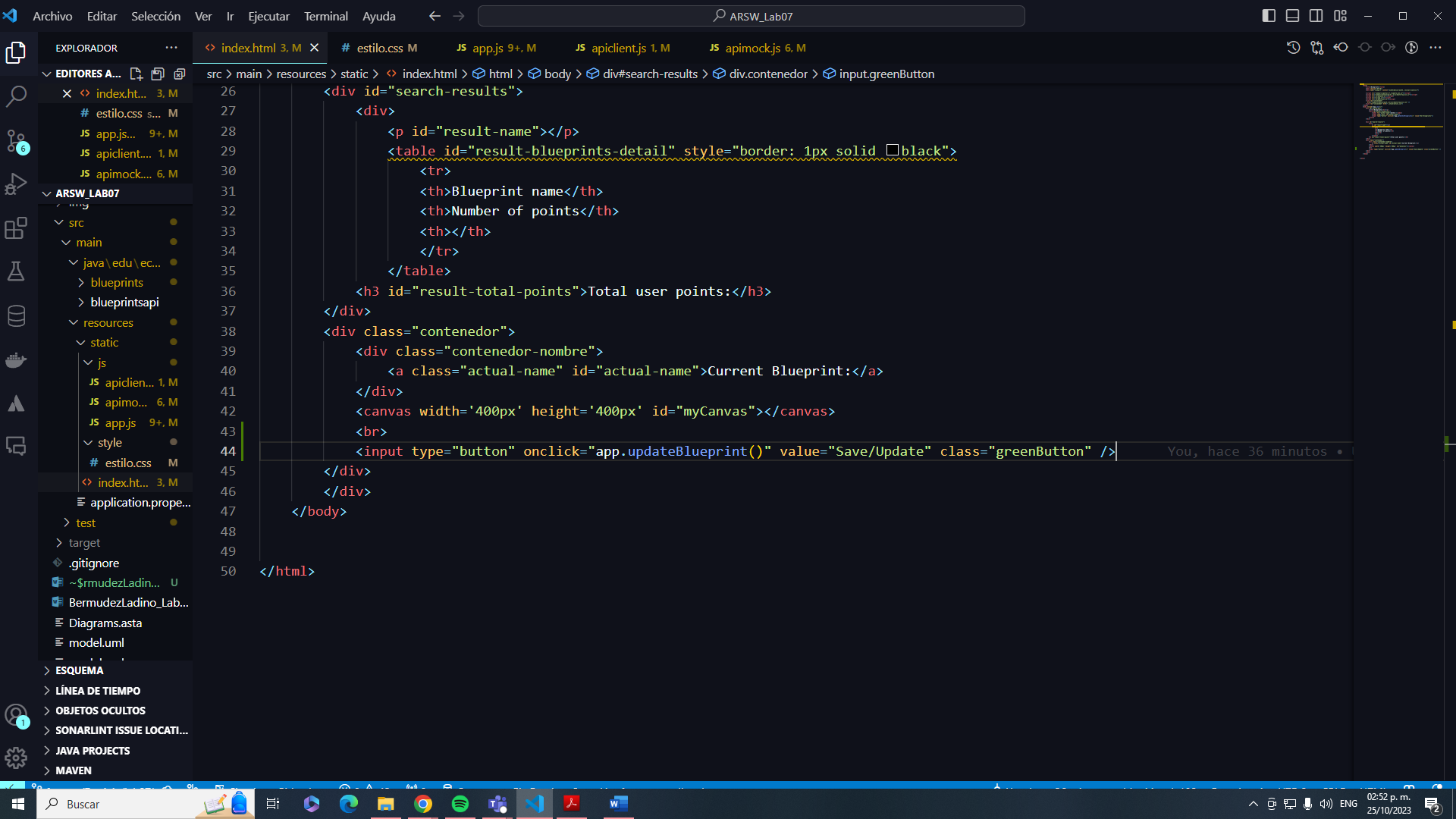
Descripción generada automáticamente

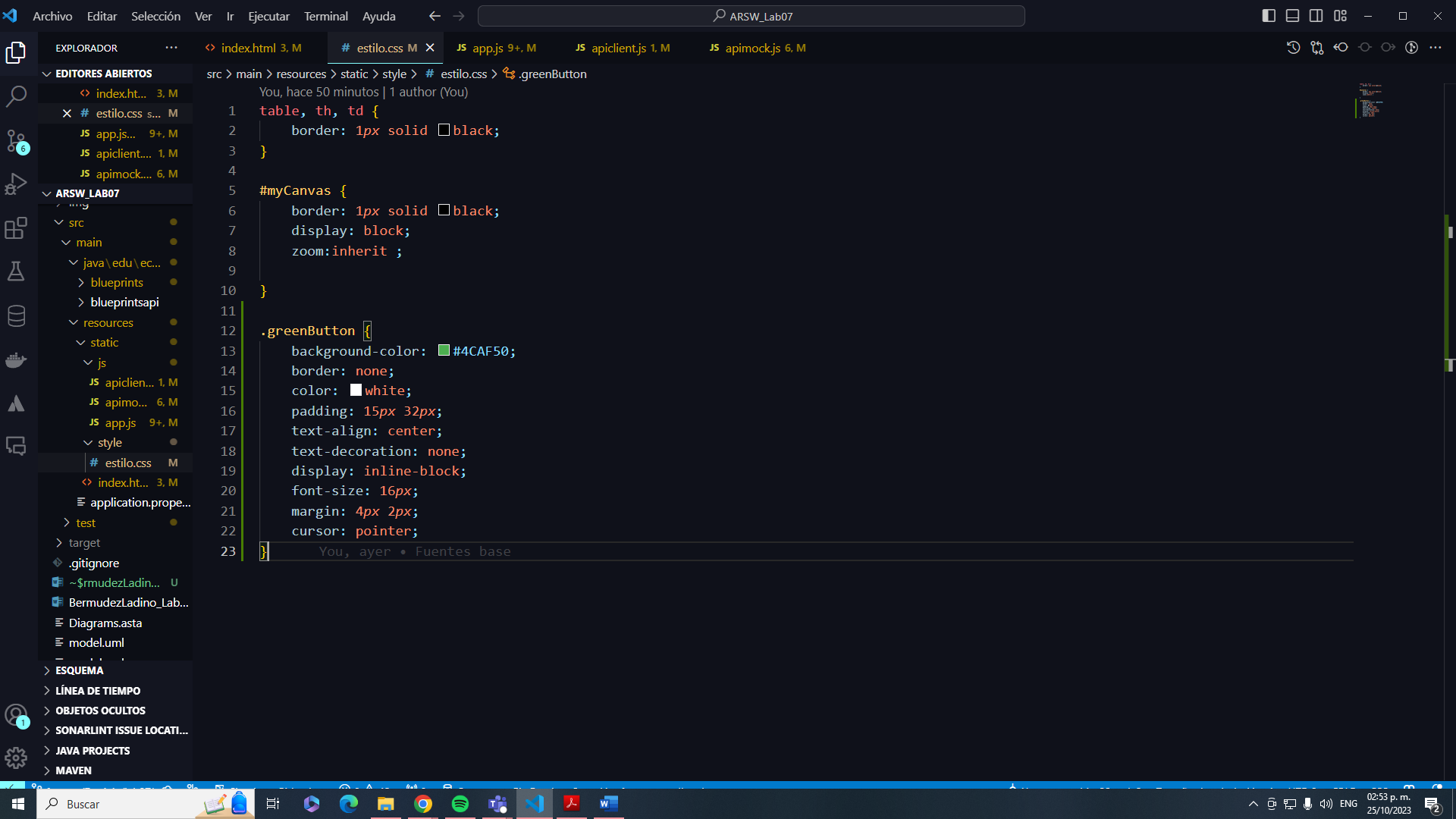
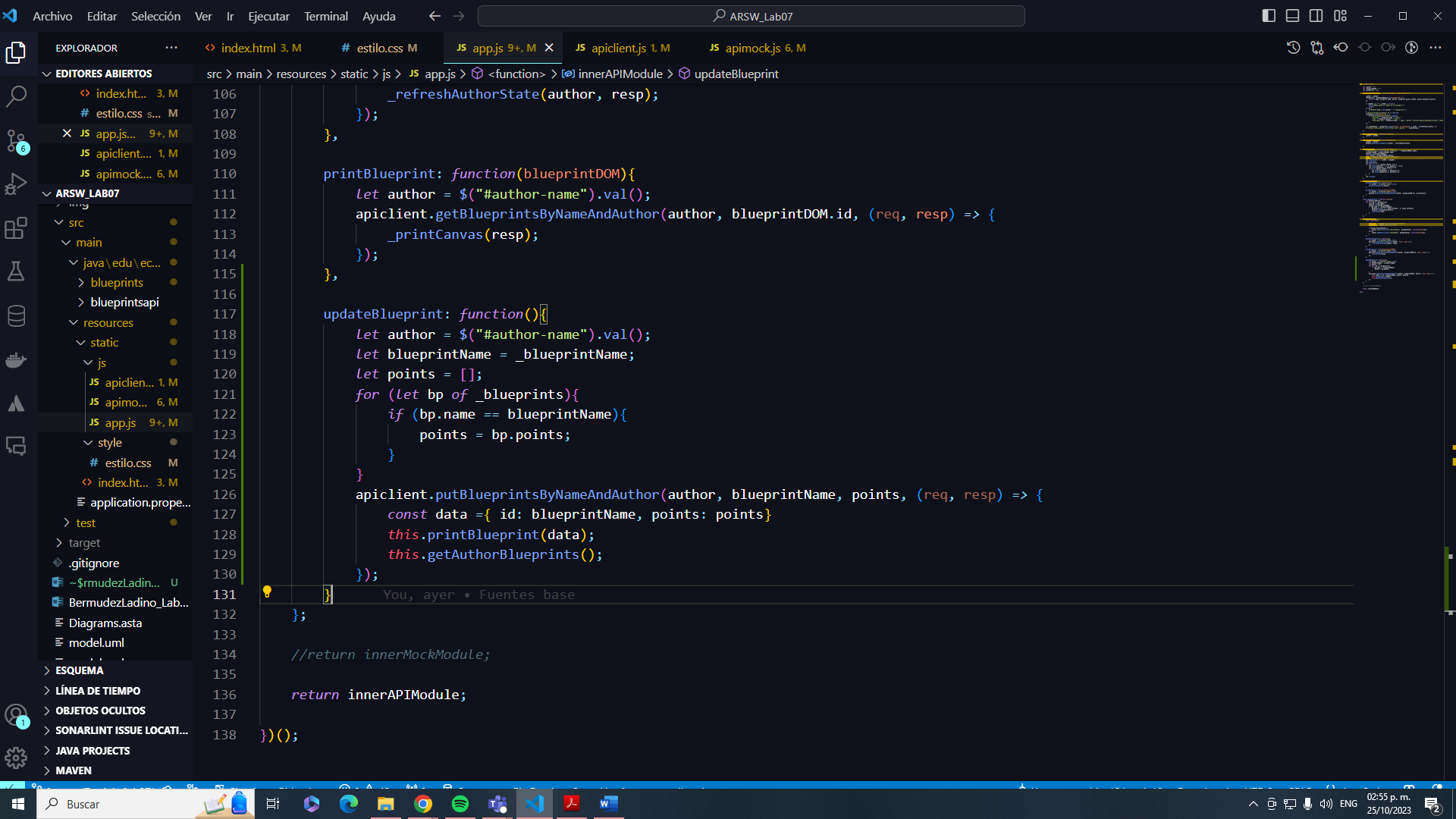
Para éste note que la propiedad 'data' del objeto enviado a $.ajax debe ser un objeto jSON (en formato de texto). Si el dato que quiere enviar es un objeto JavaScript, puede convertirlo a jSON con:

Imagen que contiene Forma

Descripción generada automáticamente

* Como en este caso se tienen tres operaciones basadas en callbacks, y que las mismas requieren realizarse en un orden específico, tenga en cuenta cómo usar las promesas de JavaScript [mediante alguno de los ejemplos disponibles](http://codepen.io/hcadavid/pen/jrwdgK).
* Modificación index.html



* Modificación estilo.css
* Modificación app.js  
  
* Modificación apiclient.jsCaptura de pantalla de computadora

  Descripción generada automáticamente
* Modificación apimock.js  
  Captura de pantalla de computadora

  Descripción generada automáticamente
* Resultados ejecución:Interfaz de usuario gráfica, Texto

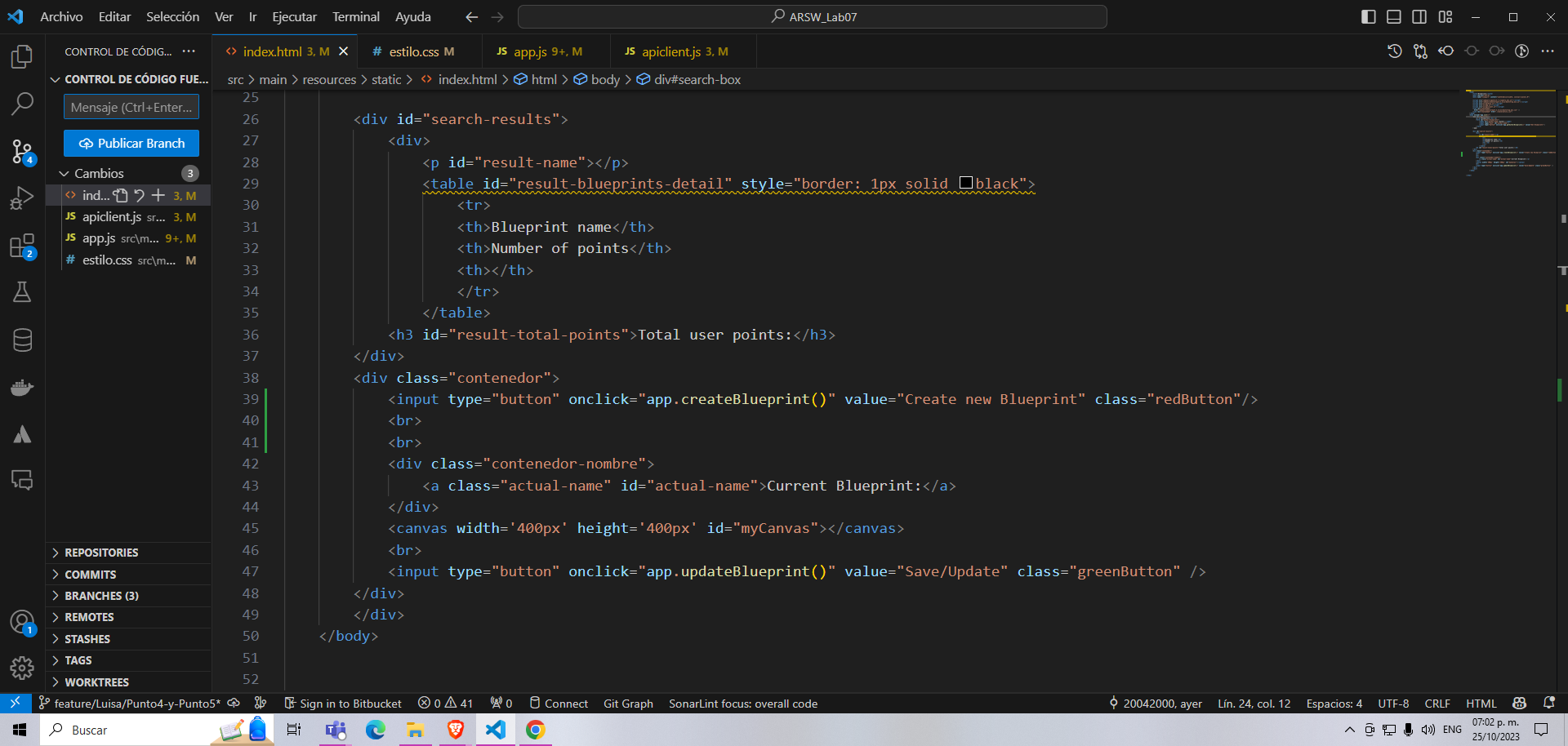
  Descripción generada automáticamente con confianza media

1. Agregue el botón 'Create new blueprint', de manera que cuando se oprima:

* Se borre el canvas actual.
* Se solicite el nombre del nuevo 'blueprint' (usted decide la manera de hacerlo).

Esta opción debe cambiar la manera como funciona la opción 'save/update', pues en este caso, al oprimirse la primera vez debe (igualmente, usando promesas):

* 1. Hacer POST al recurso /blueprints, para crear el nuevo plano.
  2. Hacer GET a este mismo recurso, para actualizar el listado de planos y el puntaje del usuario.
* Modificación index.html



* Modificación estilo.css

Captura de pantalla de computadora

Descripción generada automáticamente

* Modificación app.js

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

* Modificación apiclient.js

Captura de pantalla de computadora

Descripción generada automáticamente

* Resultados ejecución:

Creamos un nuevo Blueprient llamado PlanoPunto4

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Como podemos observar en las siguientes imágenes se actualiza el listado de planos y el puntaje del usuario.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

No se permite la creacion de un Blueprint con el mismo numbre, como se puede visualizar en los siguientes screenshots al intentar crear el Blueprint Plano4, nos dice que ya existe un plano con este nombre

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

1. Agregue el botón 'DELETE', de manera que (también con promesas):

* Borre el canvas.
* Haga DELETE del recurso correspondiente.
* Haga GET de los planos ahora disponibles.

1. **Conclusiones**