Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

Laboratorio 6

Construcción de un Cliente ‘grueso’ con un API REST, HTML5, JavaScript y CSS3. Parte I

url repositorio:

<https://github.com/ARSW2023-2/ARSW_Lab06.git>

Integrantes:

Luisa Fernanda Bermúdez Girón

Karol Daniela Ladino Ladino

Squad:

Inside Out

Profesor:

Javier Iván Toquica Barrera

Curso:

ARSW – 1

Fecha De Entrega:

06-10-2023

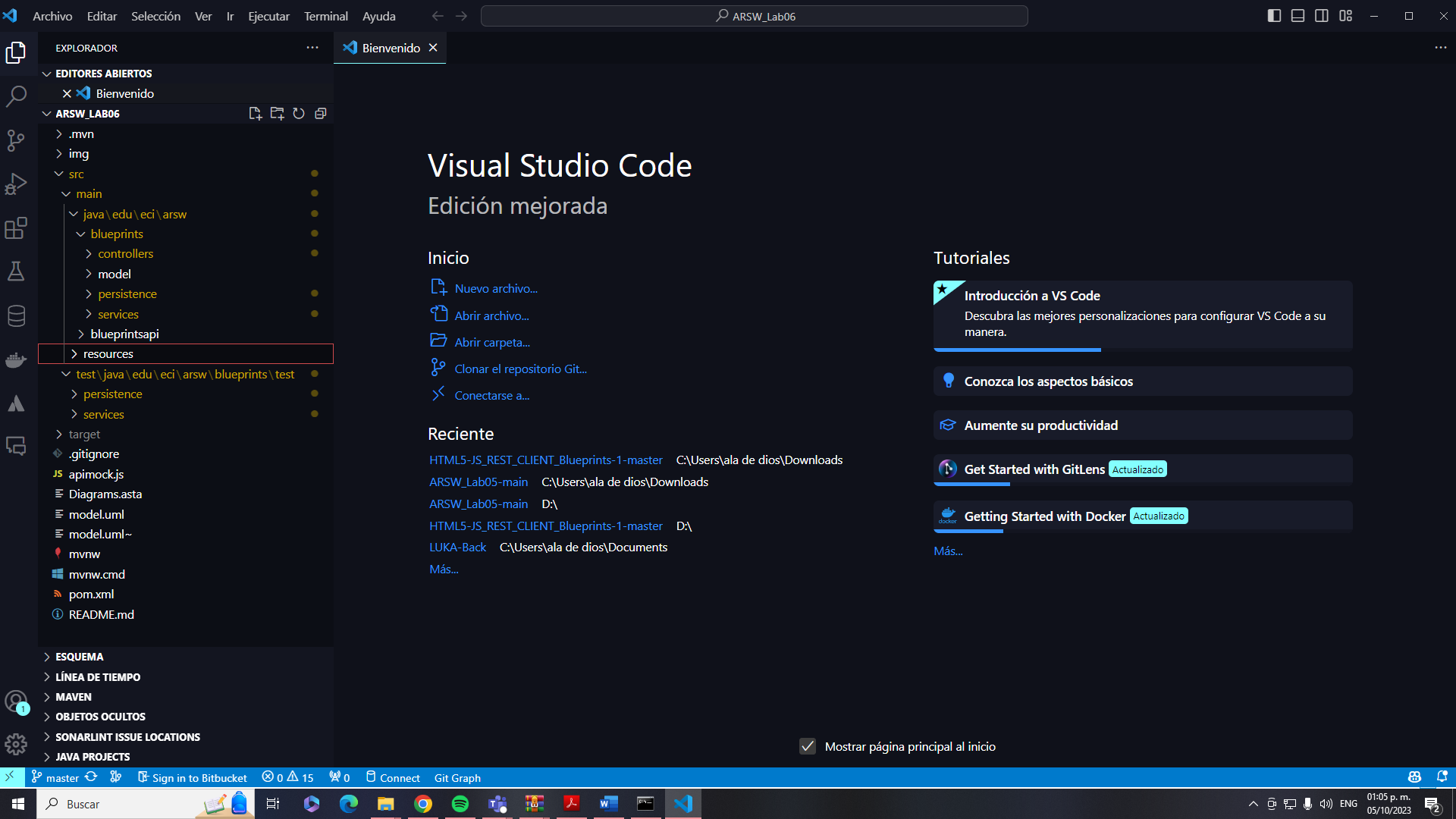
1. **Introducción**

En el campo de las Arquitecturas de Software, la construcción de aplicaciones web que interactúan con APIs REST se ha vuelto fundamental para proporcionar a los usuarios una experiencia eficiente y atractiva. Este laboratorio se centra en el desarrollo de un cliente "grueso" que utiliza tecnologías web modernas como HTML5, JavaScript y CSS3 para interactuar con un servicio de consulta de planos de autor. La aplicación permitirá a los usuarios buscar y visualizar planos, ofreciendo información detallada sobre los mismos.

1. **Desarrollo del laboratorio**

**Ajustes Backend**

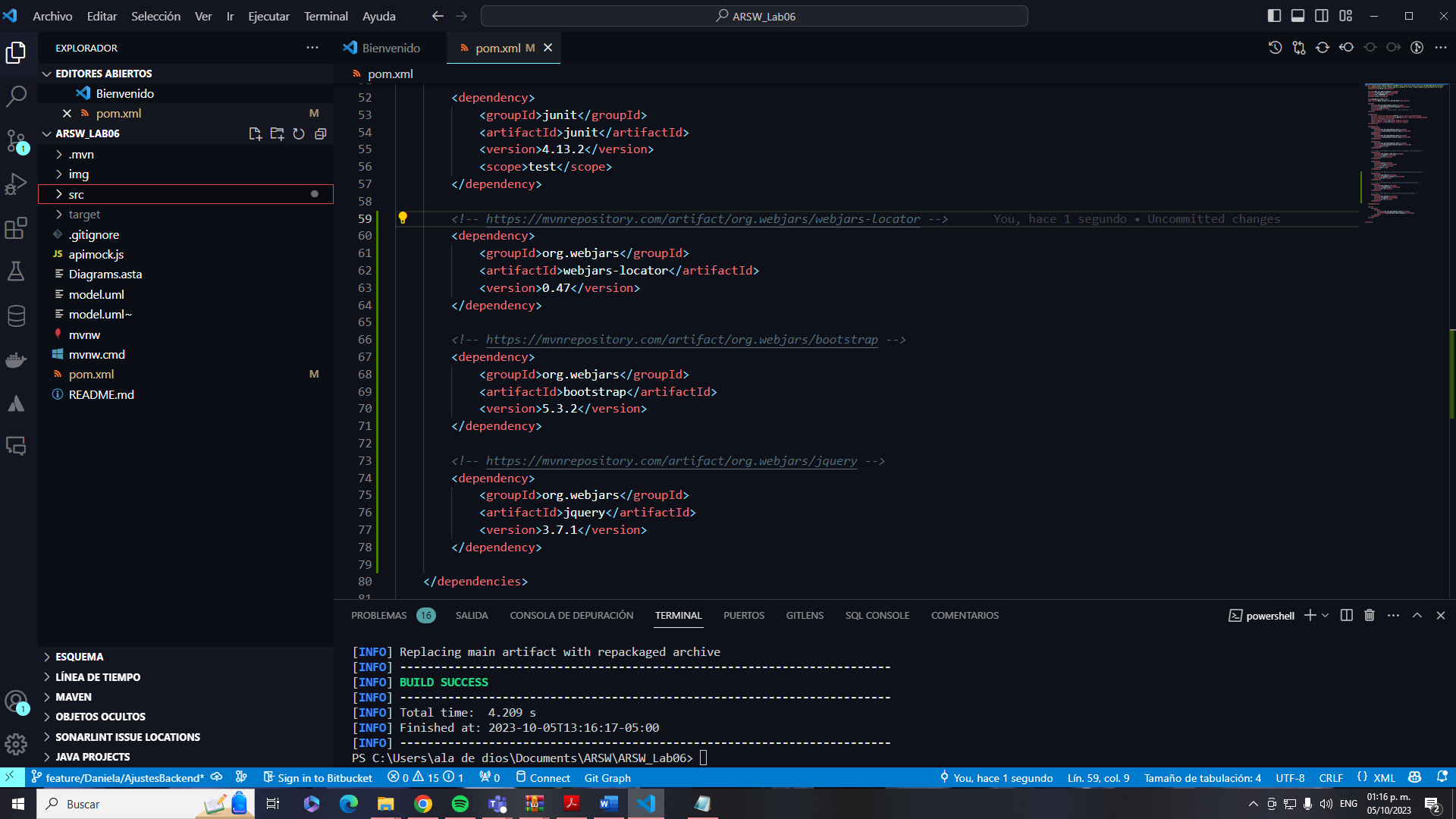
* + - 1. Trabaje sobre la base del proyecto anterior (en el que se hizo el API REST).



* + - 1. Incluya dentro de las dependencias de Maven los 'webjars' de jQuery y Bootstrap (esto permite tener localmente dichas librerías de JavaScript al momento de construir el proyecto):Interfaz de usuario gráfica, Texto, Aplicación

         Descripción generada automáticamente
* Modificación pom

Incluimos las dependencias mencionadas con las ultimas versiones de cada una de estas

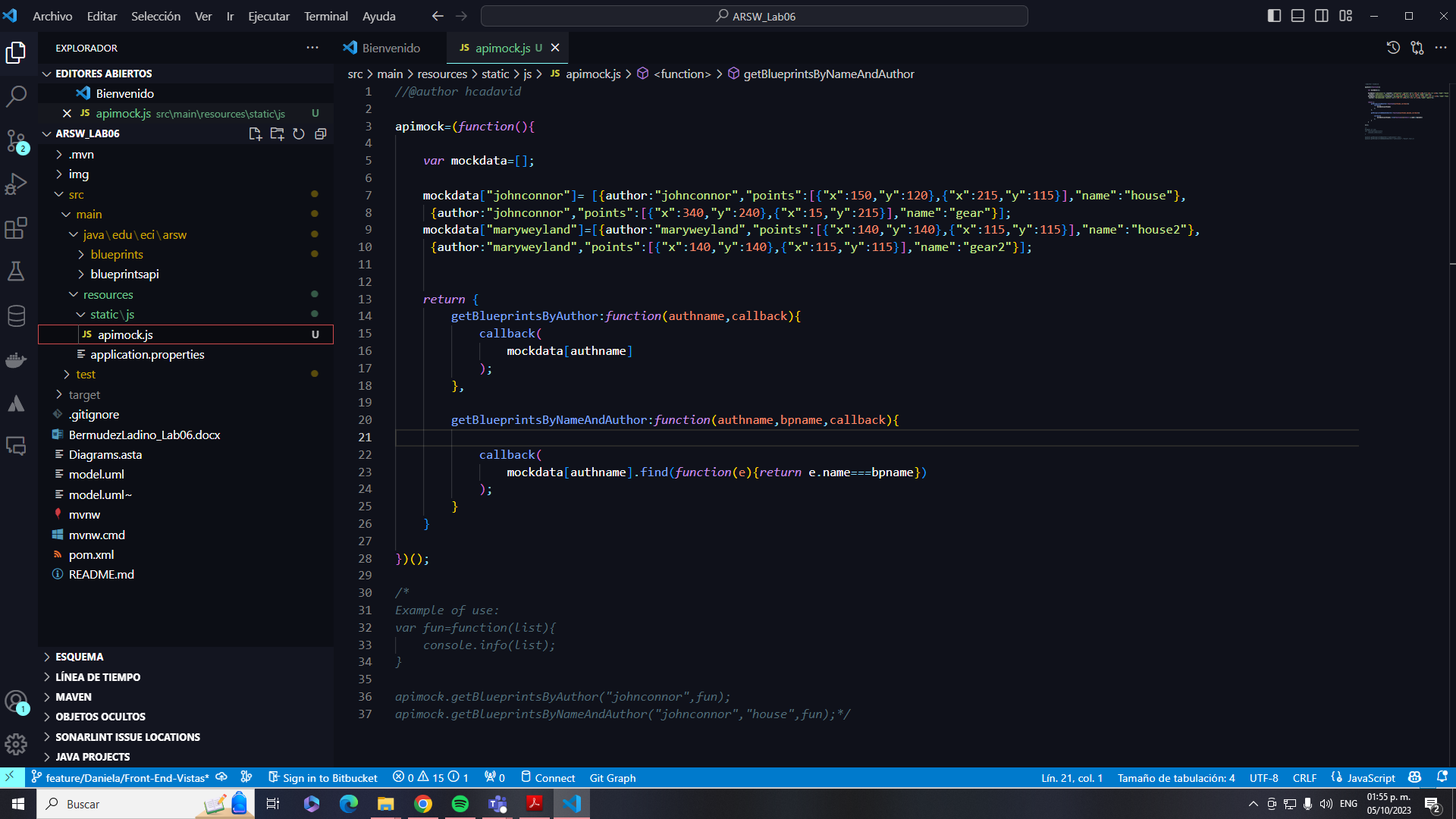


**Front-End-Vistas**

Cree el directorio donde residirá la aplicación JavaScript. Como se está usando SpringBoot, la ruta para poner en el mismo contenido estático (páginas Web estáticas, aplicaciones HTML5/JS, etc) es:

Imagen que contiene Logotipo

Descripción generada automáticamente



Cree, en el directorio anterior, la página index.html, sólo con lo básico: título, campo para la captura del autor, botón de 'Get blueprints', campo donde se mostrará el nombre del autor seleccionado, [la tabla HTML](https://www.w3schools.com/html/html_tables.asp) donde se mostrará el listado de planos (con sólo los encabezados), y un campo donde se mostrará el total de puntos de los planos del autor. Recuerde asociarles identificadores a dichos componentes para facilitar su búsqueda mediante selectores.

* + - 1. En el elemento <head> de la página, agregue las referencias a las librerías de jQuery, Bootstrap y a la hoja de estilos de Bootstrap.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* + - 1. Suba la aplicación (mvn spring-boot:run), y rectifique:

Que la página sea accesible desde:



* + 1. Al abrir la consola de desarrollador del navegador, NO deben aparecer mensajes de error 404 (es decir, que las librerías de JavaScript se cargaron correctamente).

**Front-End-Lógica**

* + - 1. Ahora, va a crear un Módulo JavaScript que, a manera de controlador, mantenga los estados y ofrezca las operaciones requeridas por la vista. Para esto tenga en cuenta el [patrón Módulo de JavaScript](https://toddmotto.com/mastering-the-module-pattern/), y cree un módulo en la ruta static/js/app.js.
      2. Copie el módulo provisto (apimock.js) en la misma ruta del módulo antes creado. En éste agréguele más planos (con más puntos) a los autores 'quemados' en el código.
      3. Agregue la importación de los dos nuevos módulos a la página HTML (después de las importaciones de las librerías de jQuery y Bootstrap):

Texto

Descripción generada automáticamente

* + - 1. Haga que el módulo antes creado mantenga de forma privada:
* El nombre del autor seleccionado.
* El listado de nombre y tamaño de los planos del autor seleccionado. Es decir, una lista objetos, donde cada objeto tendrá dos propiedades: nombre de plano, y número de puntos del plano.

Junto con una operación pública que permita cambiar el nombre del autor actualmente seleccionado.

* + - 1. Agregue al módulo 'app.js' una operación pública que permita actualizar el listado de los planos, a partir del nombre de su autor (dado como parámetro). Para hacer esto, dicha operación debe invocar la operación 'getBlueprintsByAuthor' del módulo 'apimock' provisto, enviándole como callback una función que:
* Tome el listado de los planos, y le aplique una función 'map' que convierta sus elementos a objetos con sólo el nombre y el número de puntos.
* Sobre el listado resultante, haga otro 'map', que tome cada uno de estos elementos, y a través de jQuery agregue un elemento <tr> (con los respectvos <td>) a la tabla creada en el punto 4. Tenga en cuenta los [selectores de jQuery](https://www.w3schools.com/JQuery/jquery_ref_selectors.asp) y [los tutoriales disponibles en línea](https://www.tutorialrepublic.com/codelab.php?topic=faq&file=jquery-append-and-remove-table-row-dynamically). Por ahora no agregue botones a las filas generadas.
* Sobre cualquiera de los dos listados (el original, o el transformado mediante 'map'), aplique un 'reduce' que calcule el número de puntos. Con este valor, use jQuery para actualizar el campo correspondiente dentro del DOM.
  + - 1. Asocie la operación antes creada (la de app.js) al evento 'on-click' del botón de consulta de la página.
      2. Verifique el funcionamiento de la aplicación. Inicie el servidor, abra la aplicación HTML5/JavaScript, y rectifique que, al ingresar un usuario existente, se cargue el listado del mismo.

**Para la próxima semana**

* + - 1. A la página, agregue un [elemento de tipo Canvas](https://www.w3schools.com/html/html5_canvas.asp), con su respectivo identificador. Haga que sus dimensiones no sean demasiado grandes para dejar espacio para los otros componentes, pero lo suficiente para poder 'dibujar' los planos.
      2. Al módulo app.js agregue una operación que, dado el nombre de un autor, y el nombre de uno de sus planos dados como parámetros, haciendo uso del método getBlueprintsByNameAndAuthor de apimock.js y de una función callback:
* Consulte los puntos del plano correspondiente, y con los mismos dibuje consectivamente segmentos de recta, haciendo uso [de los elementos HTML5 (Canvas, 2DContext, etc) disponibles](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_canvas_tut_path)\* Actualice con jQuery el campo donde se muestra el nombre del plano que se está dibujando (si dicho campo no existe, agruéguelo al DOM).
  + - 1. Verifique que la aplicación ahora, además de mostrar el listado de los planos de un autor, permita seleccionar uno de éstos y graficarlo. Para esto, haga que en las filas generadas para el punto 5 incluyan en la última columna un botón con su evento de clic asociado a la operación hecha anteriormente (enviándo como parámetro los nombres correspondientes).
      2. Verifique que la aplicación ahora permita: consultar los planos de un auto y graficar aquel que se seleccione.
      3. Una vez funcione la aplicación (sólo front-end), haga un módulo (llámelo 'apiclient') que tenga las mismas operaciones del 'apimock', pero que para las mismas use datos reales consultados del API REST. Para lo anterior revise [cómo hacer peticiones GET con jQuery](https://api.jquery.com/jquery.get/), y cómo se maneja el esquema de callbacks en este contexto.
      4. Modifique el código de app.js de manera que sea posible cambiar entre el 'apimock' y el 'apiclient' con sólo una línea de código.
      5. Revise la [documentación y ejemplos de los estilos de Bootstrap](https://v4-alpha.getbootstrap.com/examples/) (ya incluidos en el ejercicio), agregue los elementos necesarios a la página para que sea más vistosa, y más cercana al mock dado al inicio del enunciado.

1. **Conclusiones**