

**TECNOLÓGICO DE ESTUDIOS SUPERIORES DE ECATEPEC**

# DIVISIÓN DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

**BASE DE DATOS PARA DISPOSITIVOS MÓVILES**

**INTEGRACIÓN FUNCIONAL DEL MÓDULO**

**GRUPO: 5851**

**PRESENTA:**

GARCÍA CRUZ JACIEL AARÓN

LÓPEZ ALCÁNTARA ALAN JESÚS

LUCIO ALBA BRANDON DANIEL

**PROFESOR:**

CORTES BARRERA GRISELDA

ENERO 21, 2022

**INTEGRACIÓN FUNCIONAL DEL MÓDULO**

Para la integración del módulo, que en particular era la implementación de una protección de rutas o bloqueo de usuario se realizó por medio de la funcionalidad de autentificación de un usuario por medio de su registro previo al sistema y su posterior intento de acceso por medio de su nombre de usuario y contraseña. De esta manera la protección de las rutas se ejerce por medio de la autentificación de estos parámetros. Si la información registrada inicialmente del usuario es correcta al loguear, este tendrá acceso al Home, de lo contrario el sistema negará el acceso o lo bloqueará en caso de ser necesario.

Para la implementación de esta funcionalidad a nuestro módulo se utilizó la estructura principal del módulo que menciona el uso de GUARDS, que son interfaces que permiten la protección de las rutas y especifican que tipo de usuarios pueden tener acceso a cierta dirección y que usuarios no.

Pero para probar esta funcionalidad, se realizó el **Frontend** la creación de un formulario tanto de Login, como de Registro, esto para la extracción de los datos del usuario y que en el **Backend** se realicen todas las operaciones de autentificación necesarias para la protección de las rutas.

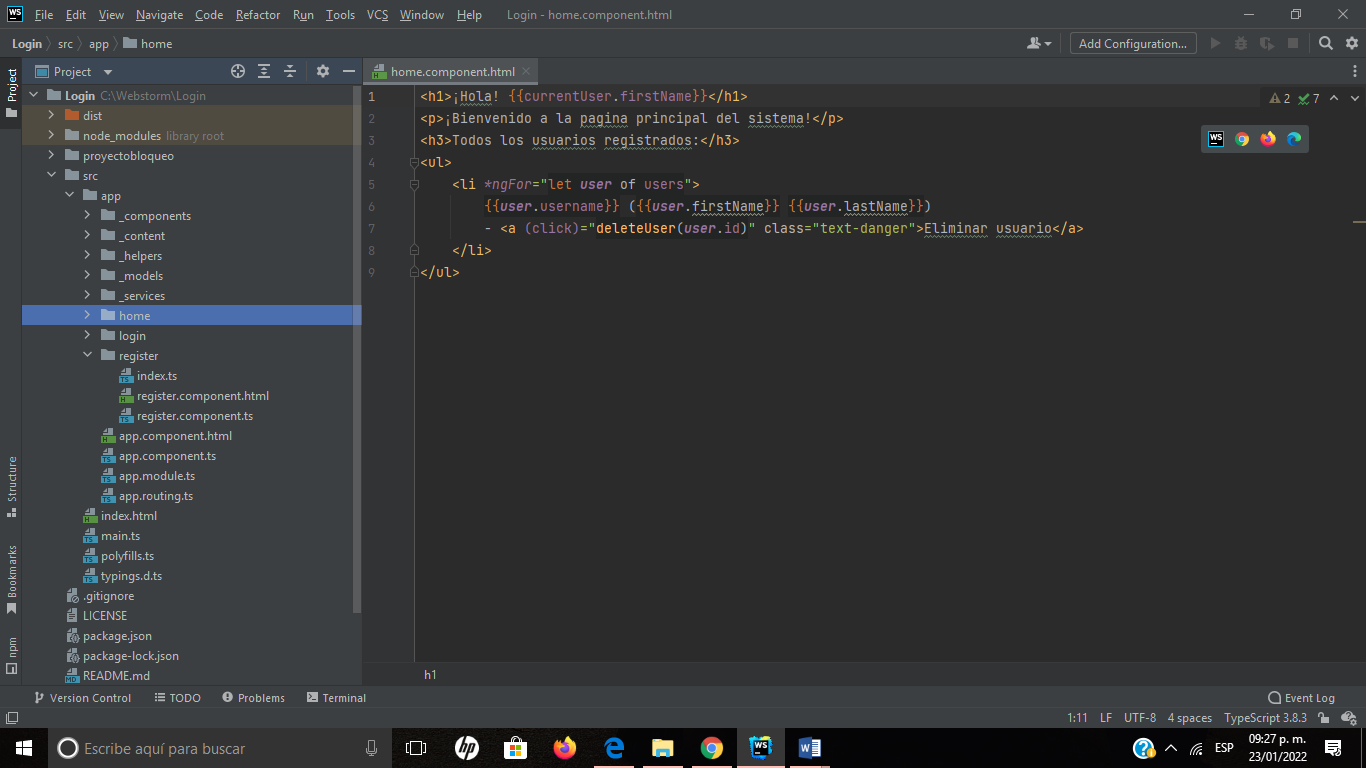
Pero para ello será necesario primero describir cada uno de los elementos que componen el Backend del proyecto.

**BACKEND**

Para la creación de este elemento se crearon directorios que contienen alertas y elementos visuales que permiten que la interfaz con la que interactúa el usuario sea más entendible.

Pero enfocándonos en los directorios más importantes, tenemos tres de ellos, que son: Home, Login y Register.

Estos directorios cuentan cada uno de ellos con sus tres archivos principales como el index, el html y el component.ts.



**DIRECTORIO HOME**

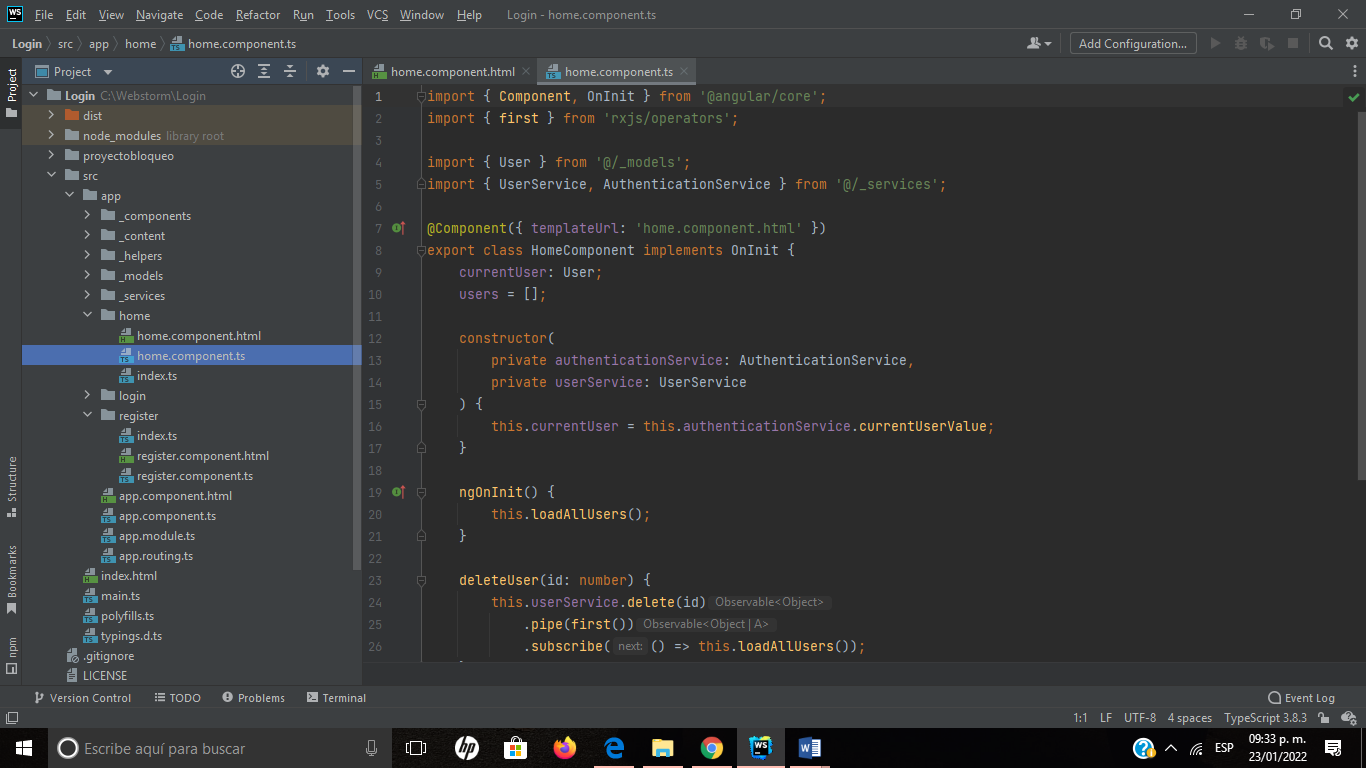
*Home.component.html*

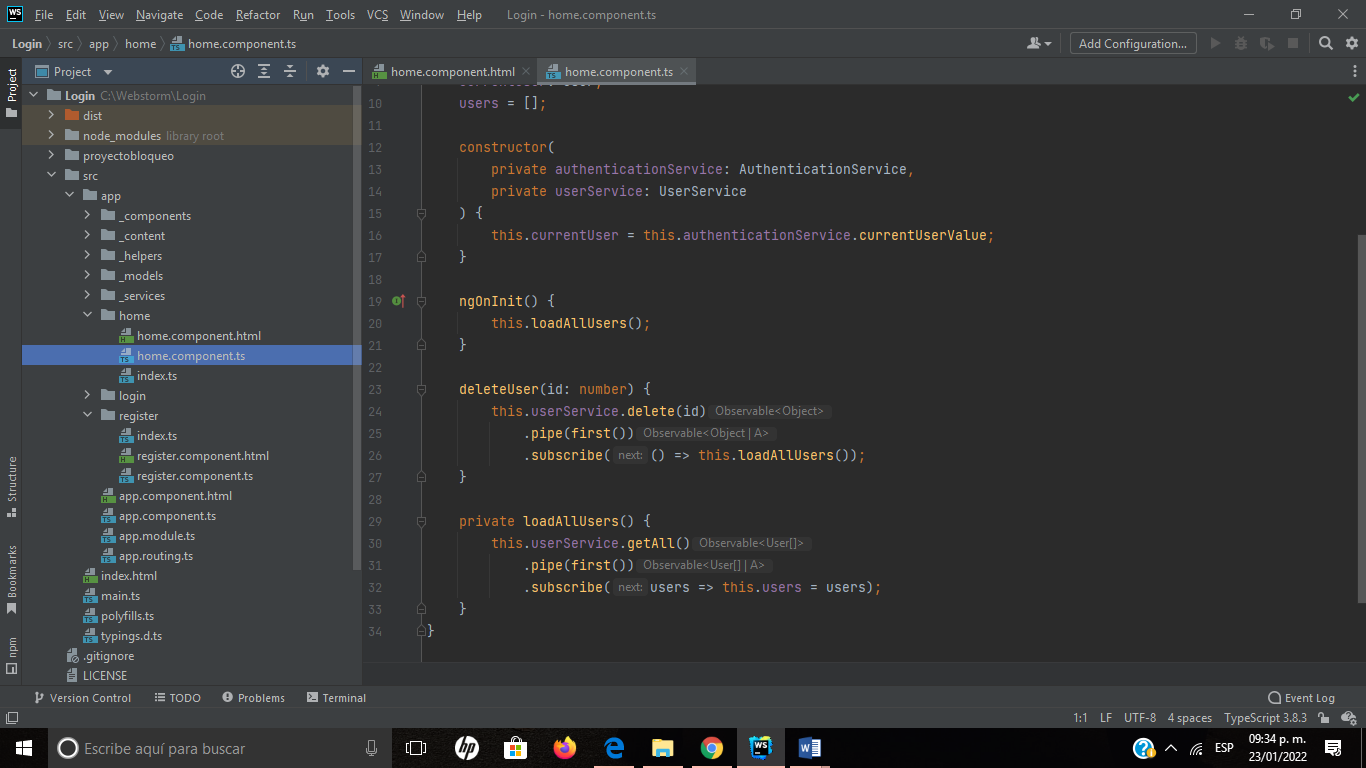
Estas funciones van a permitirnos dar la bienvenida al usuario, registrar los usuarios que se encuentran registrados y nos servirá como ruta principal de protección para usuarios no logueados.



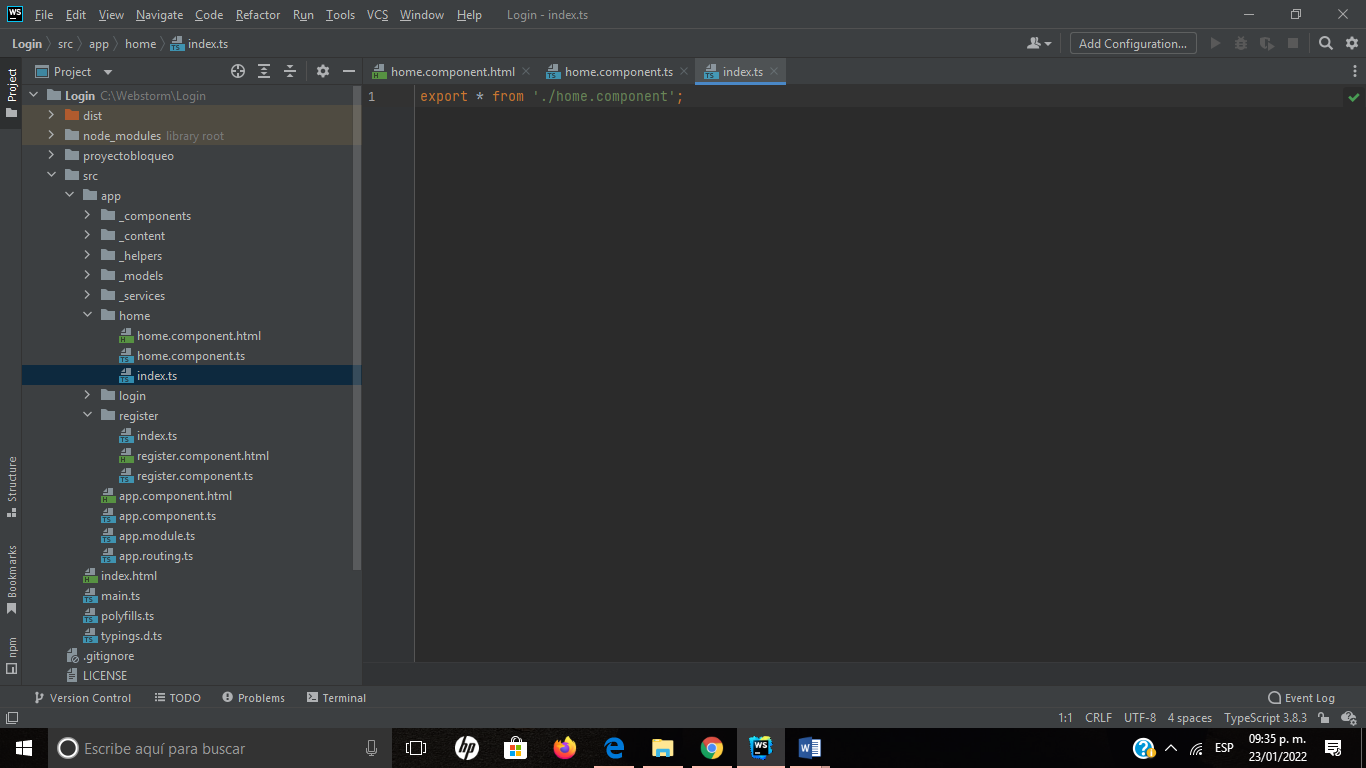
*Home.component.ts*

En este bloque se van a alojar los datos de los usuarios que se registran en el sistema y que pueden ser manipulados directamente desde el sitio para su eliminación.





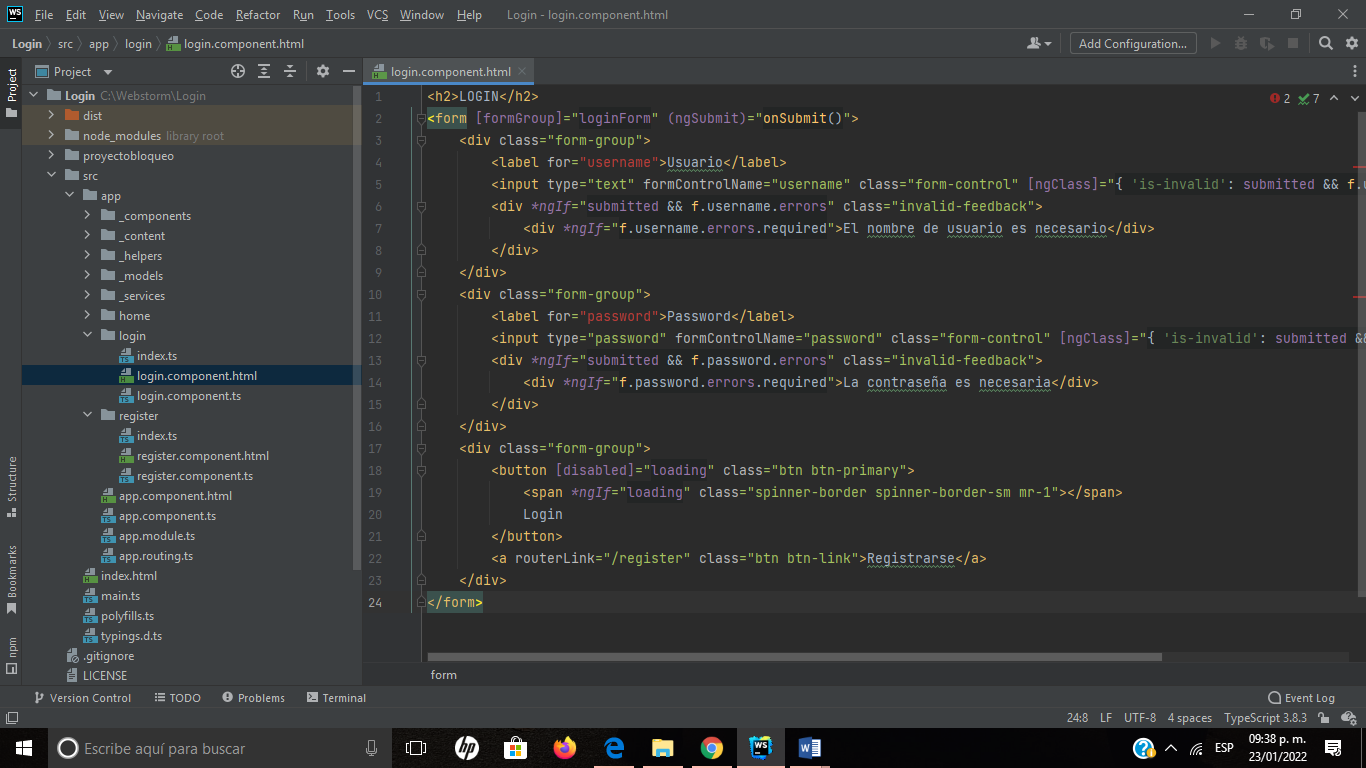
Para el archivo index únicamente exportamos el contenido del componente.



**DIRECTORIO LOGIN**

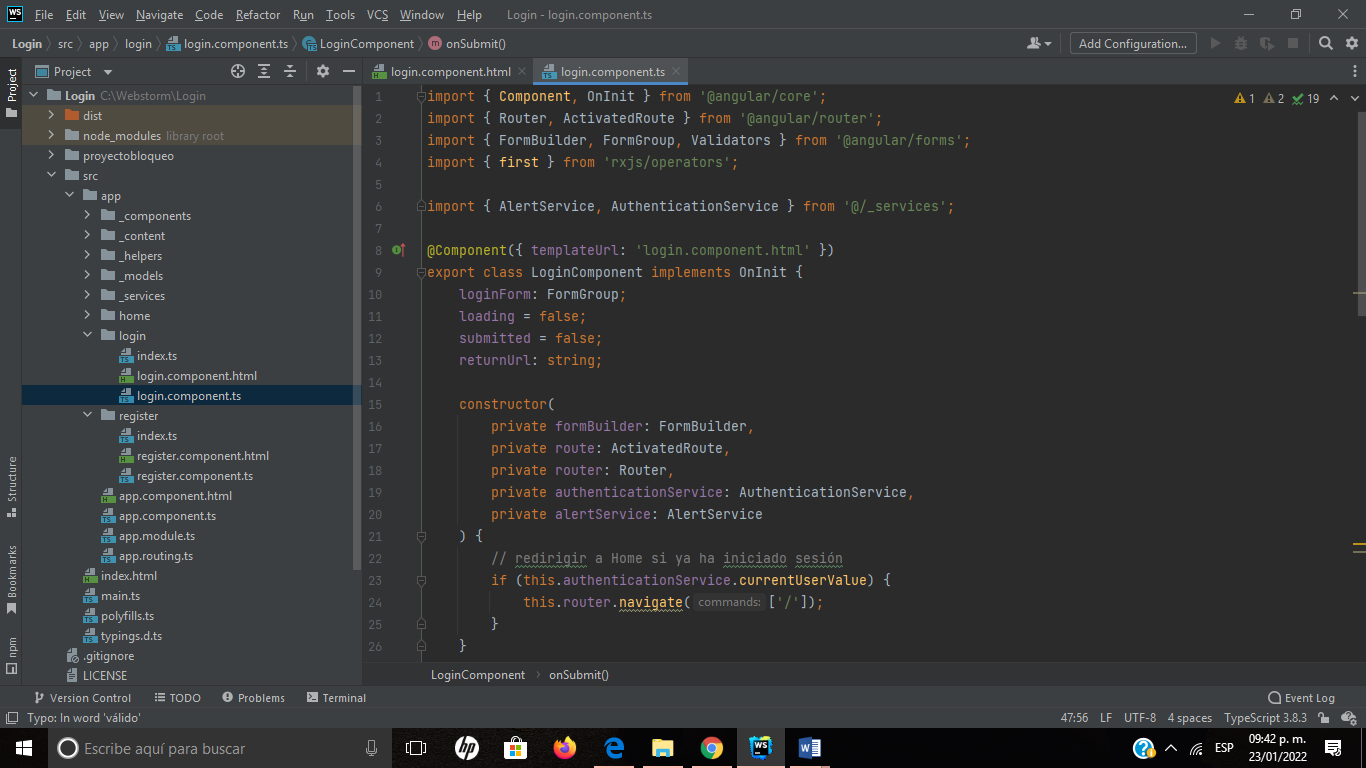
*Login.component.html*

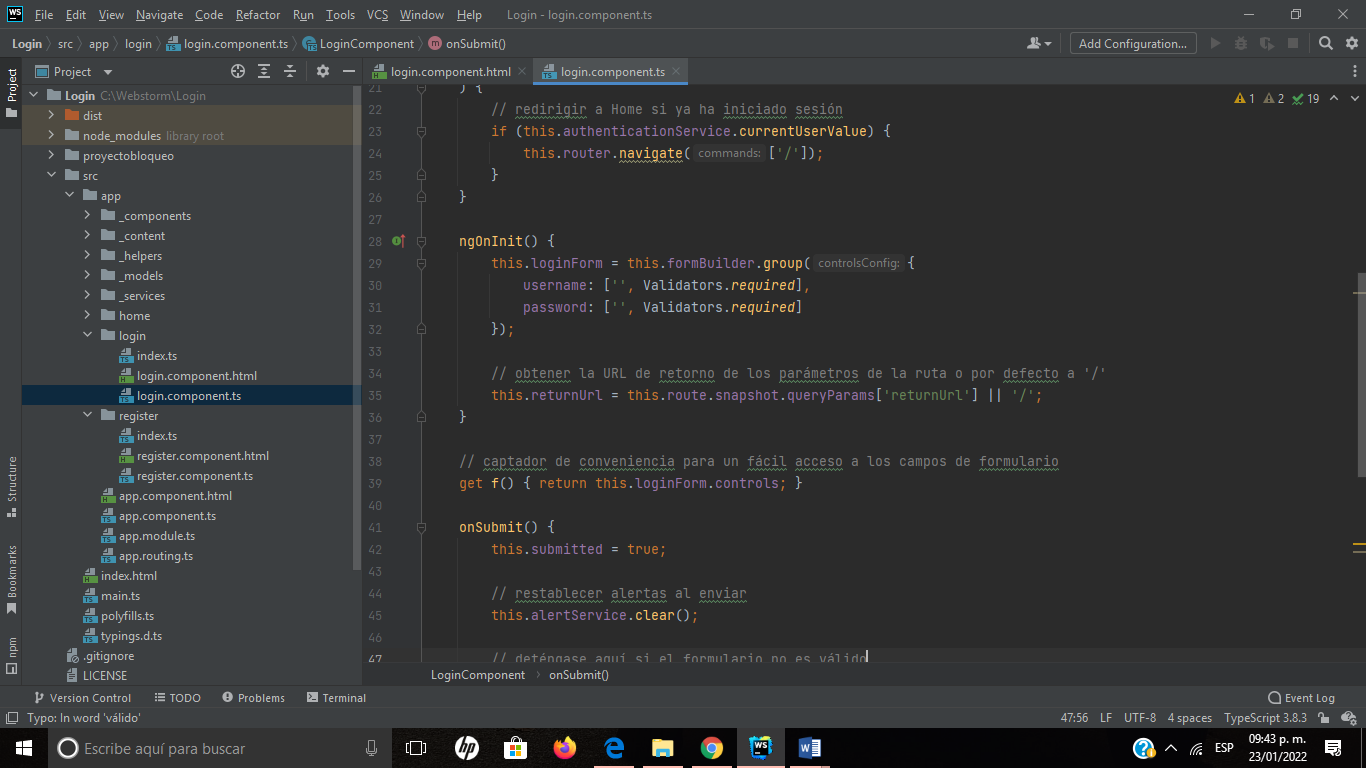
En este archivo se contienen todas las etiquetas de la interfaz, los campos a llenar y algunas alertas y restricciones que se le asignan al usuario, como la asignación del llenado de los campos o del número mínimo de caracteres del password. También contiene las redirecciones al registro de nuevos usuarios.



*Login.component.ts*

En este archivo se contienen los elementos de retorno y parámetros de autentificación, se liga con las alertas y se cumplen las funcionalidades, en caso de que el usuario y el password sean correctos se redirigirá a el Home o página principal y en caso de ser incorrectos aparecerán las alertas y los mensajes que indiquen que el usuario o contraseña son incorrectas.





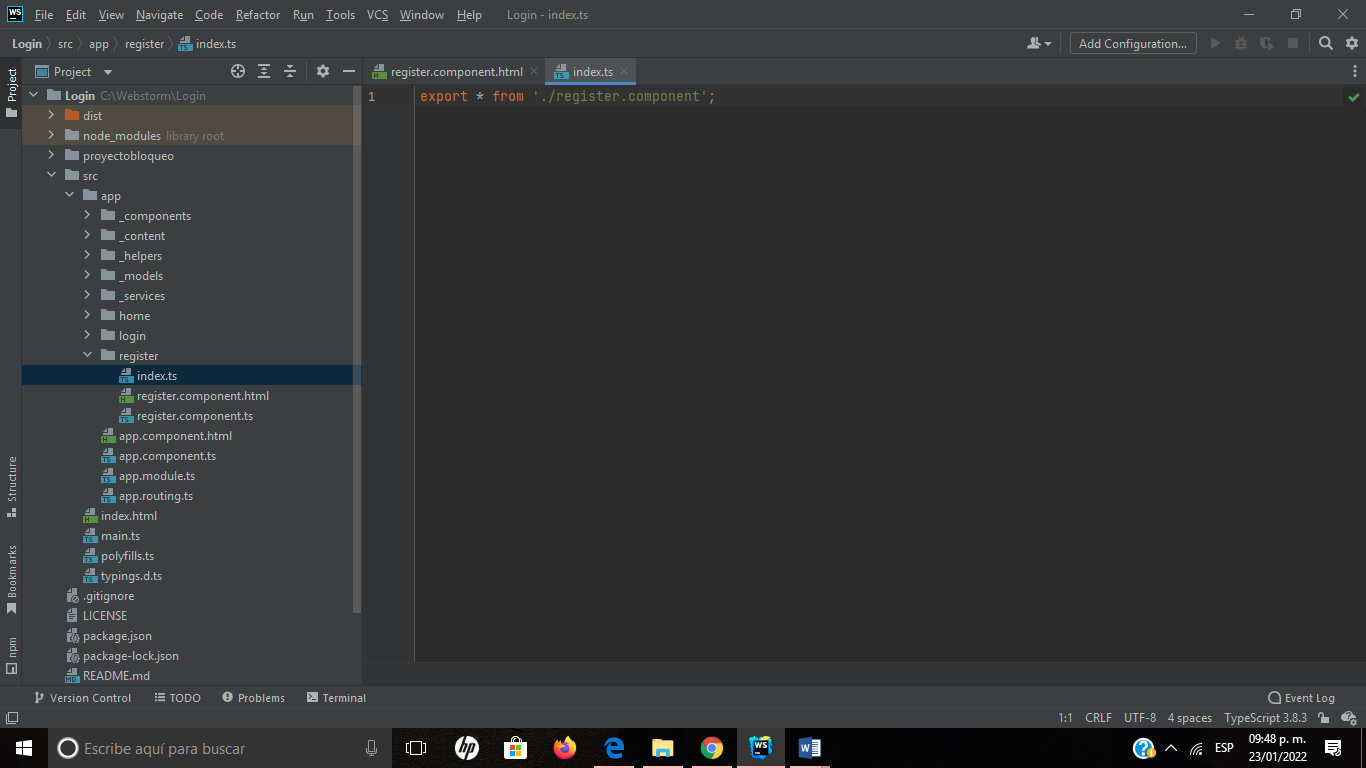


Y de igual forma el index contiene la exportación del componente.



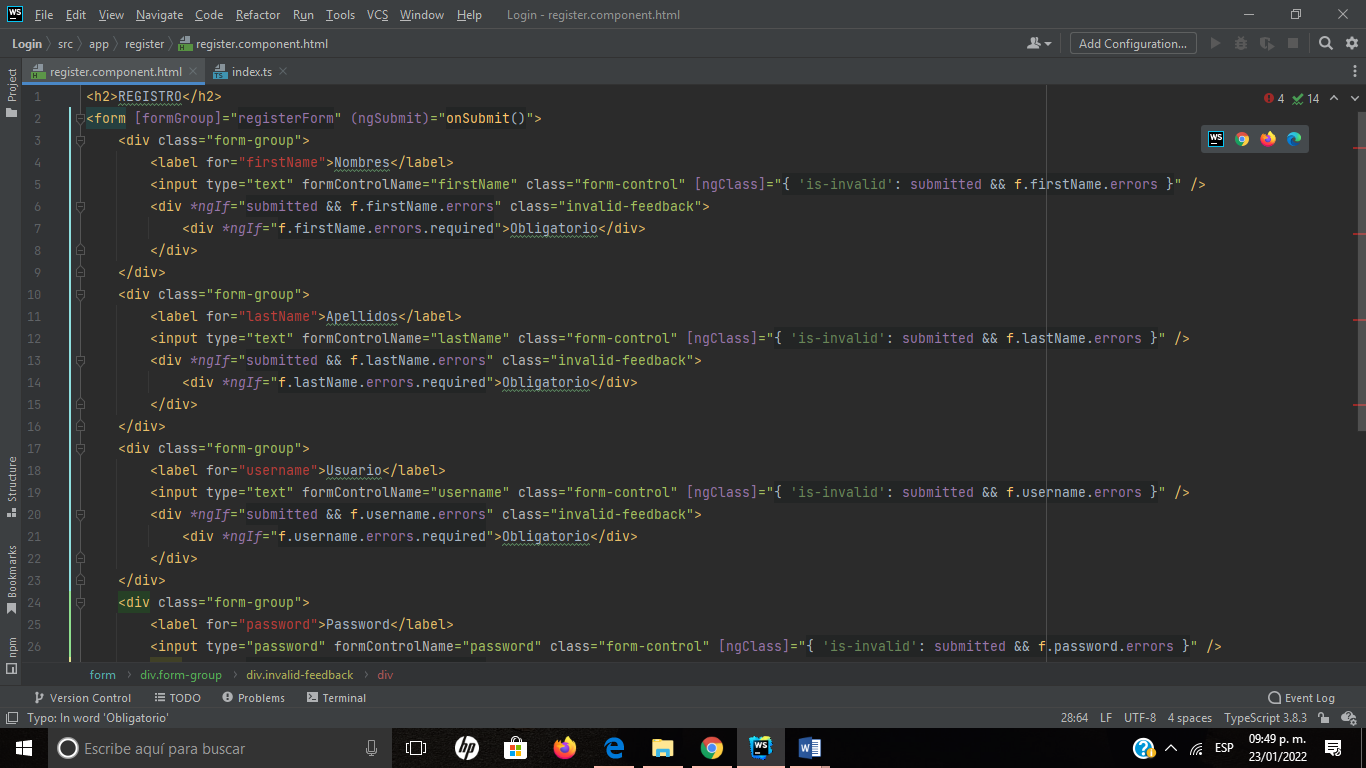
**DIRECTORIO REGISTRO**

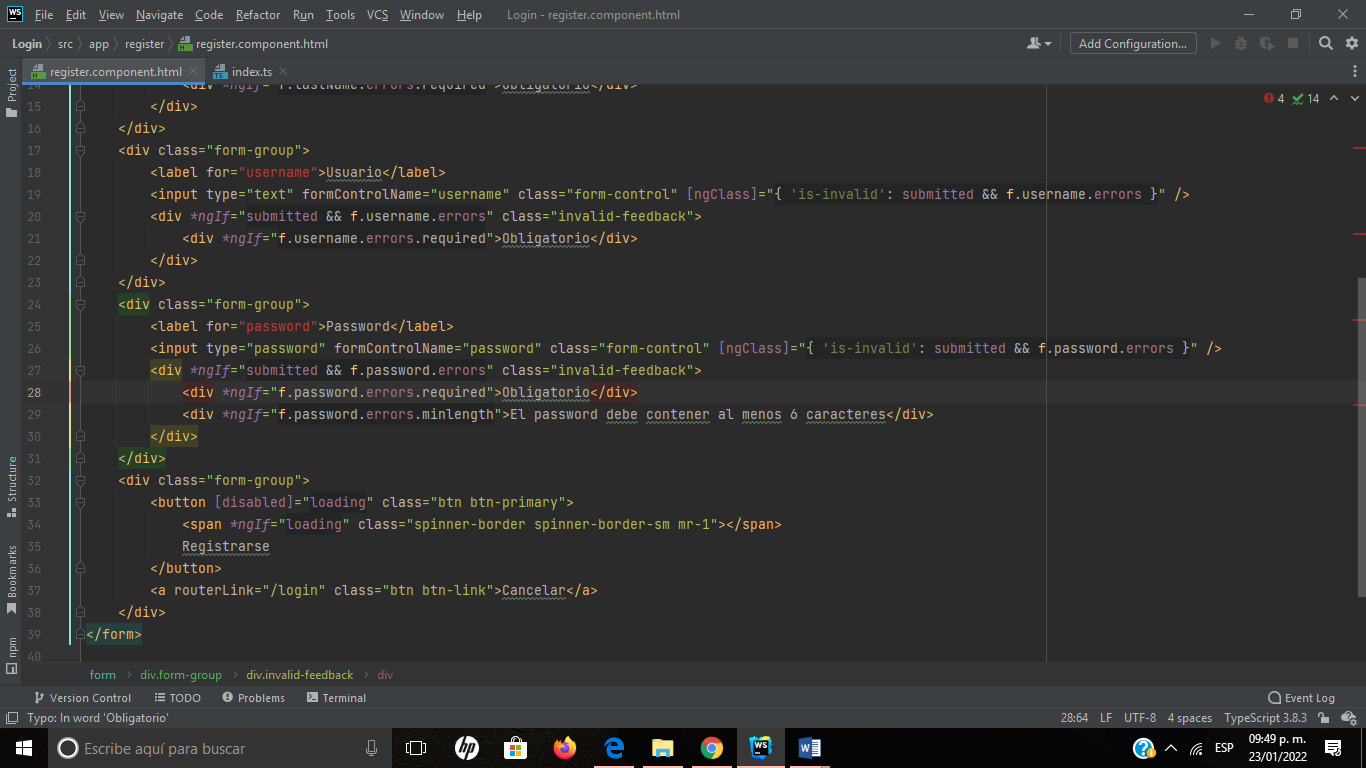
Inicialmente, de igual manera que con los otros componentes, este se exporta en el index.



*Register.component.html*

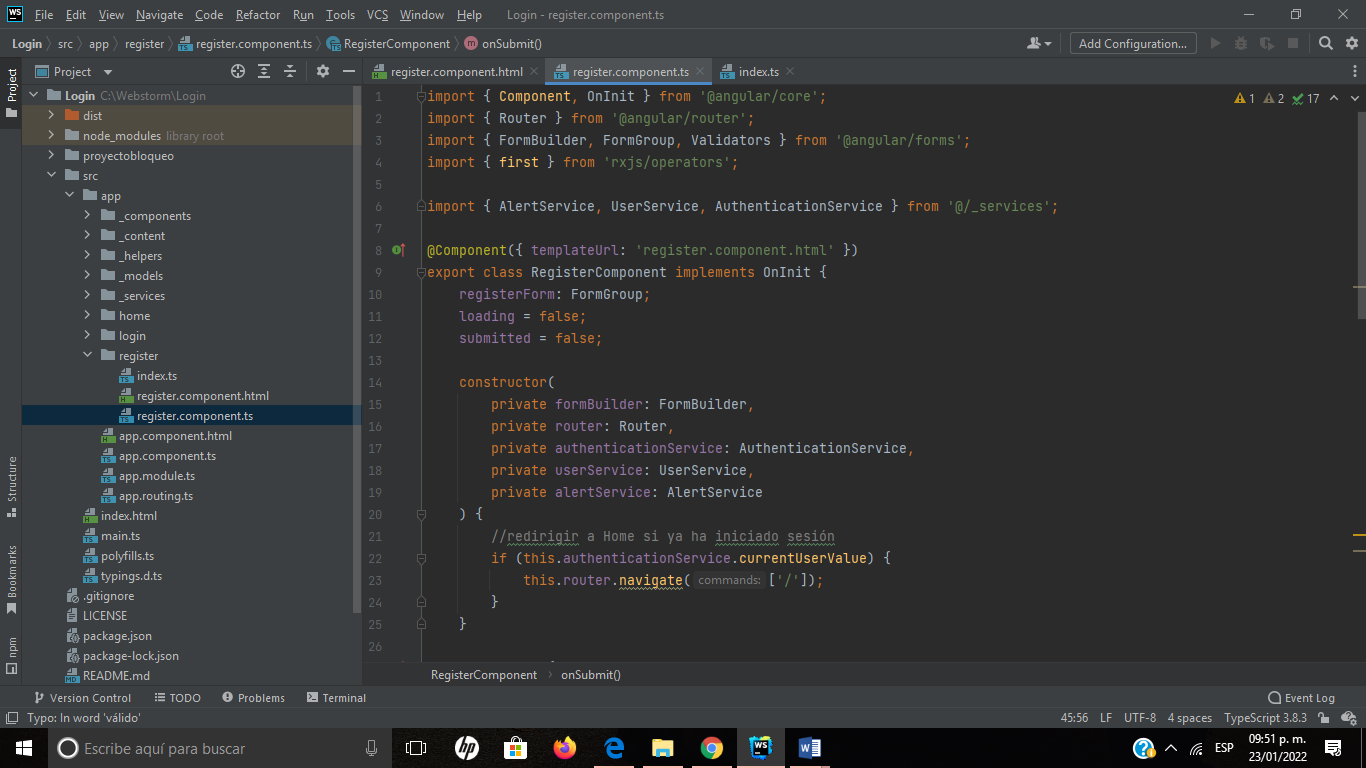
Este archivo contiene todos los elementos necesarios para el registro de los usuarios, los campos en donde el usuario añade sus datos y los botones de retorno o de registro. El usuario recibirá las alertas sobre los campos que son obligatorios o las sentencias requeridas, así como mensajes que puedan determinar al usuario si su registro fue exitoso o si el usuario ya se encuentra registrado.

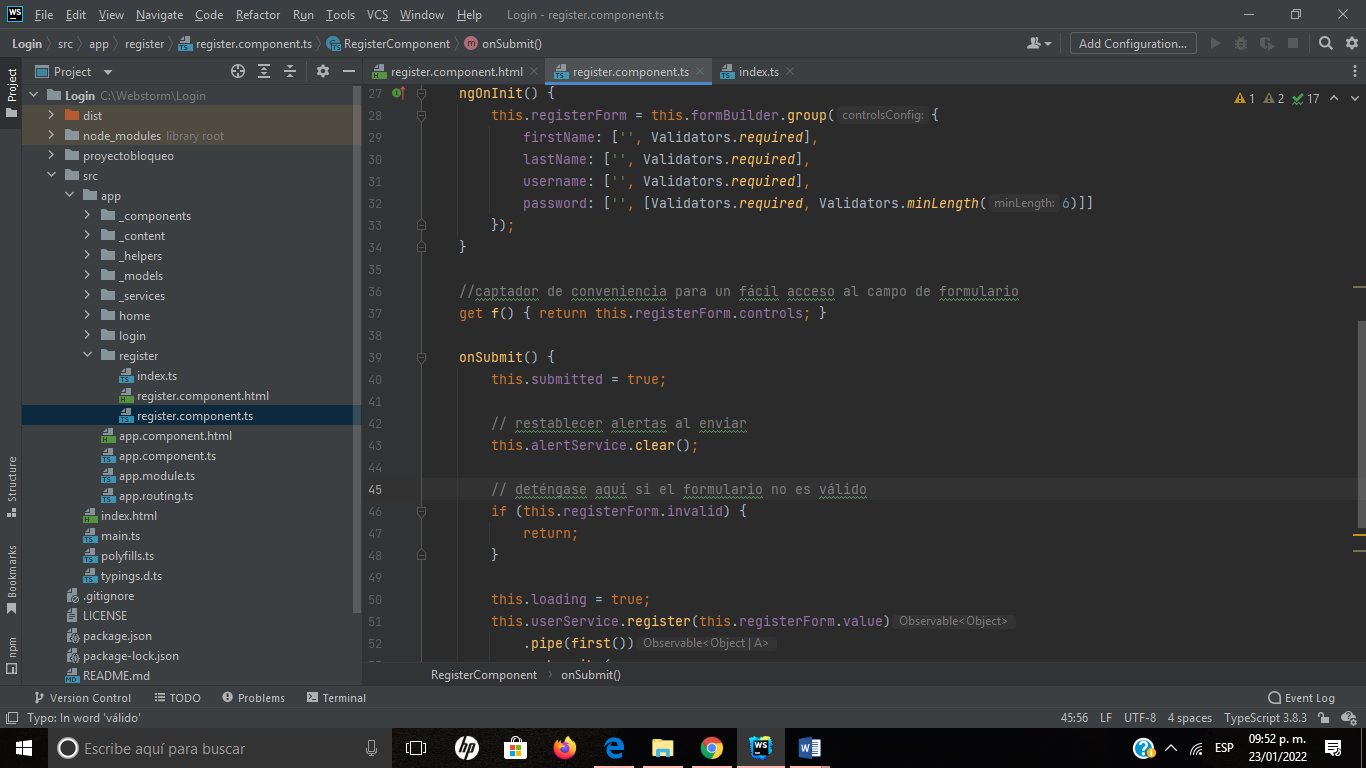


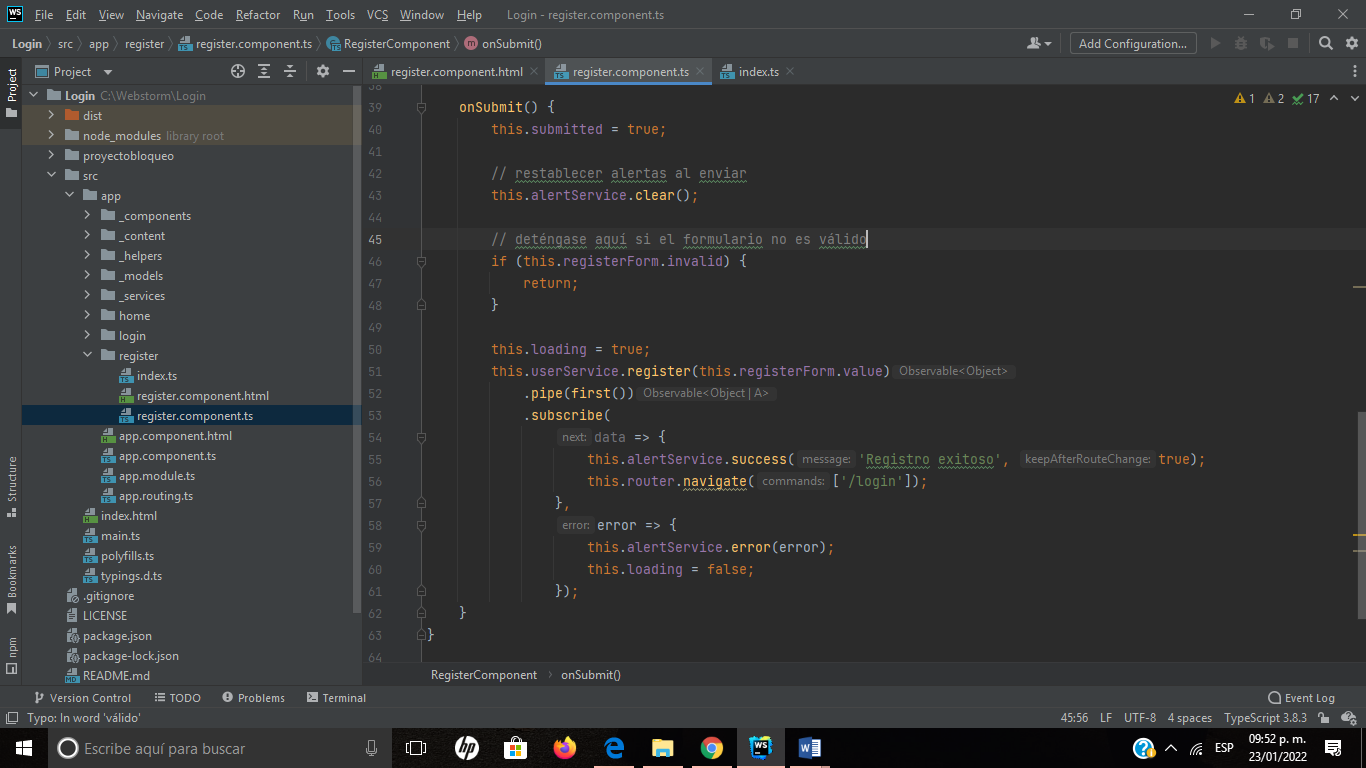


*Register.component.ts*

Este archivo contiene los elementos necesarios que validen los campos y los datos que se introducen en ellos, así como su extracción y registro de usuarios. Es el encargado de enviar los mensajes al usuario sobre su registro y almacena la información necesaria para la autentificación.

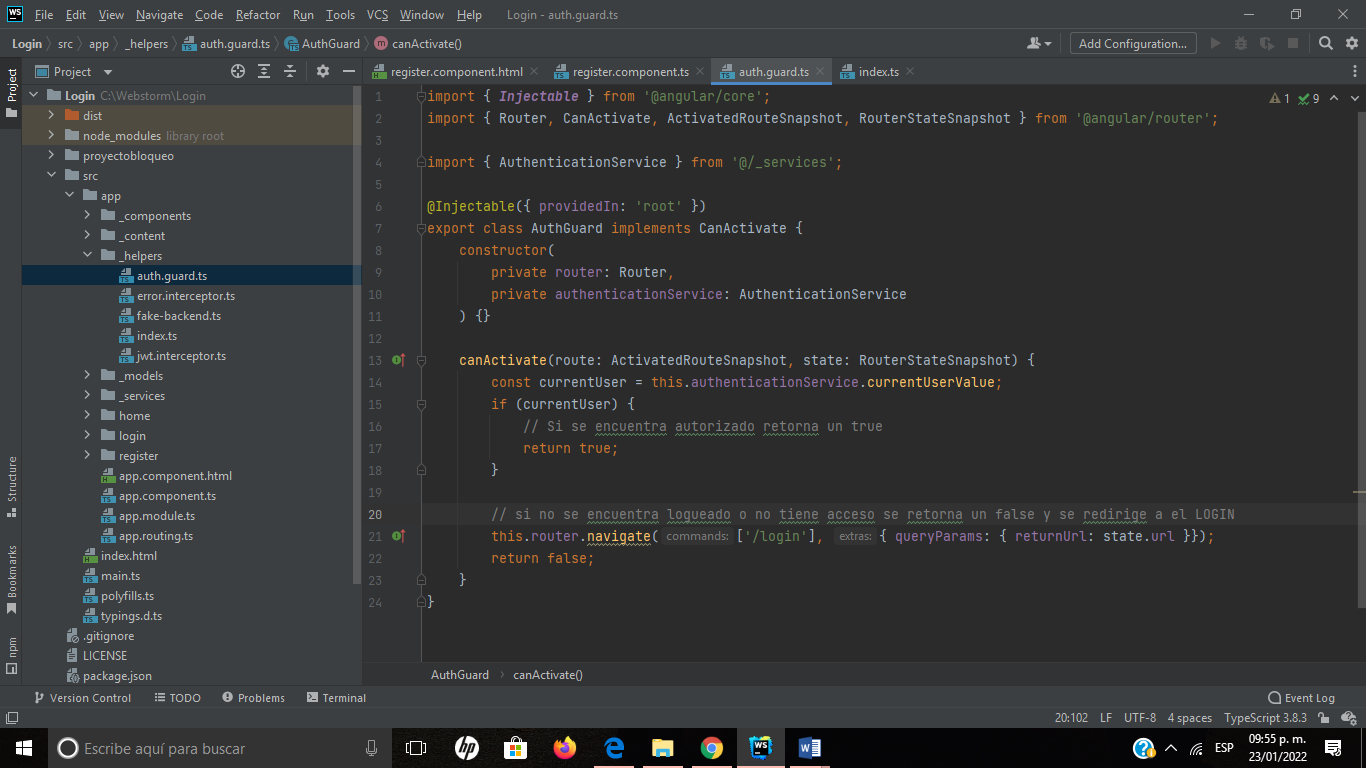






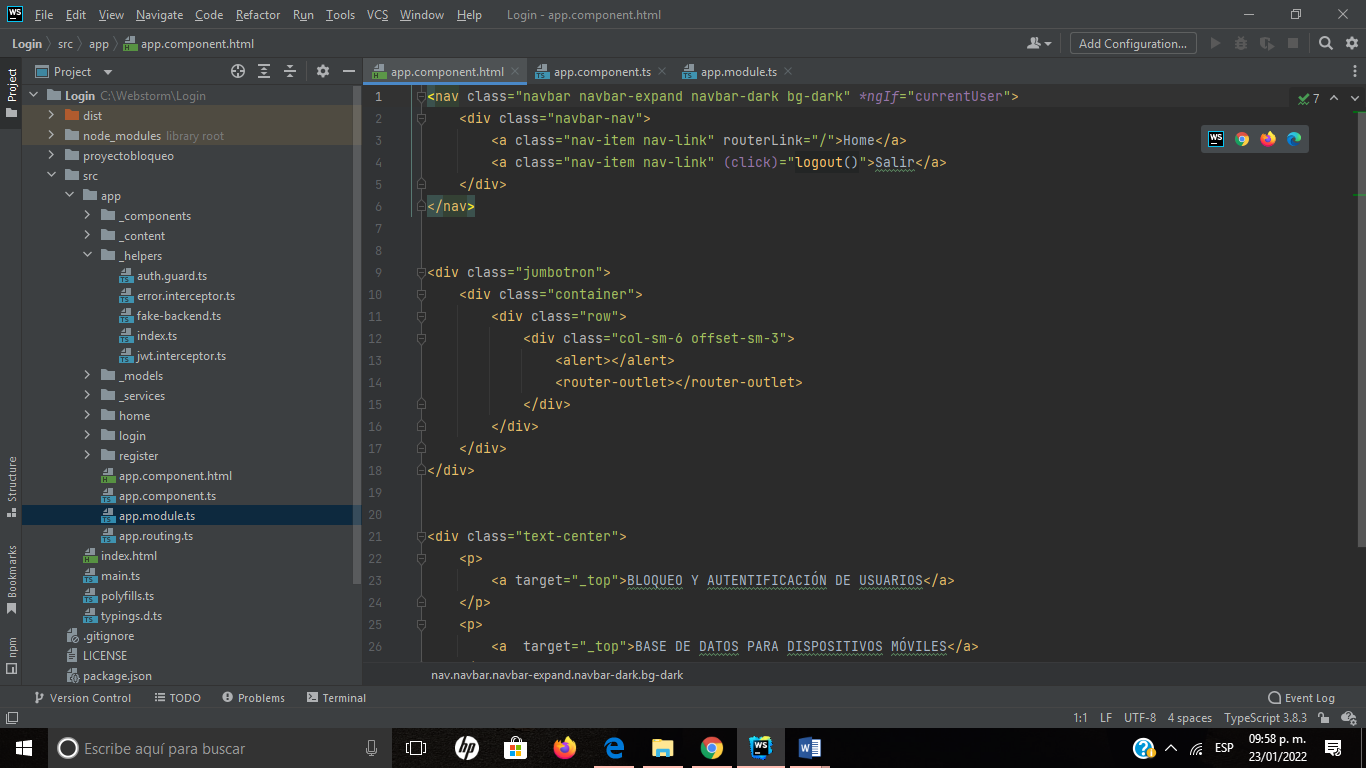
**AUTENTIFICACIÓN DEL USUARIO**

Para ello como lo habíamos mencionado inicialmente, se implementó un GUARD, esto con CanActivate, esta sentencia funciona con valores booleanos, se verifica que inicialmente estamos en la ruta del Login, se introducen los datos, si estos se encuentran registrados en el sistema y el usuario y password coinciden, retorna un true y redirige al usuario al HOME, de lo contrario retorna un false y lo redirige nuevamente al LOGIN.

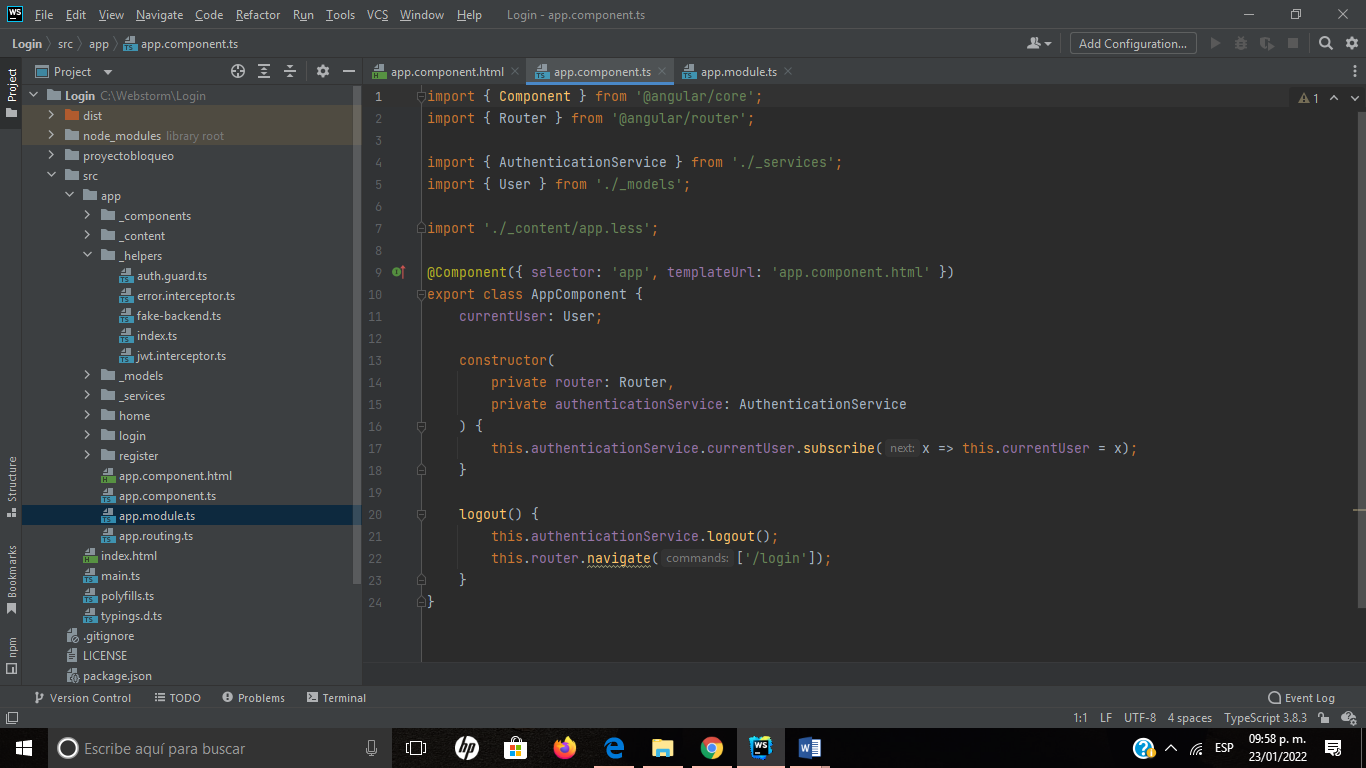


Finalmente tenemos los archivos generales de la aplicación como lo son el html, component.ts, module.ts y el archivo de routing.ts

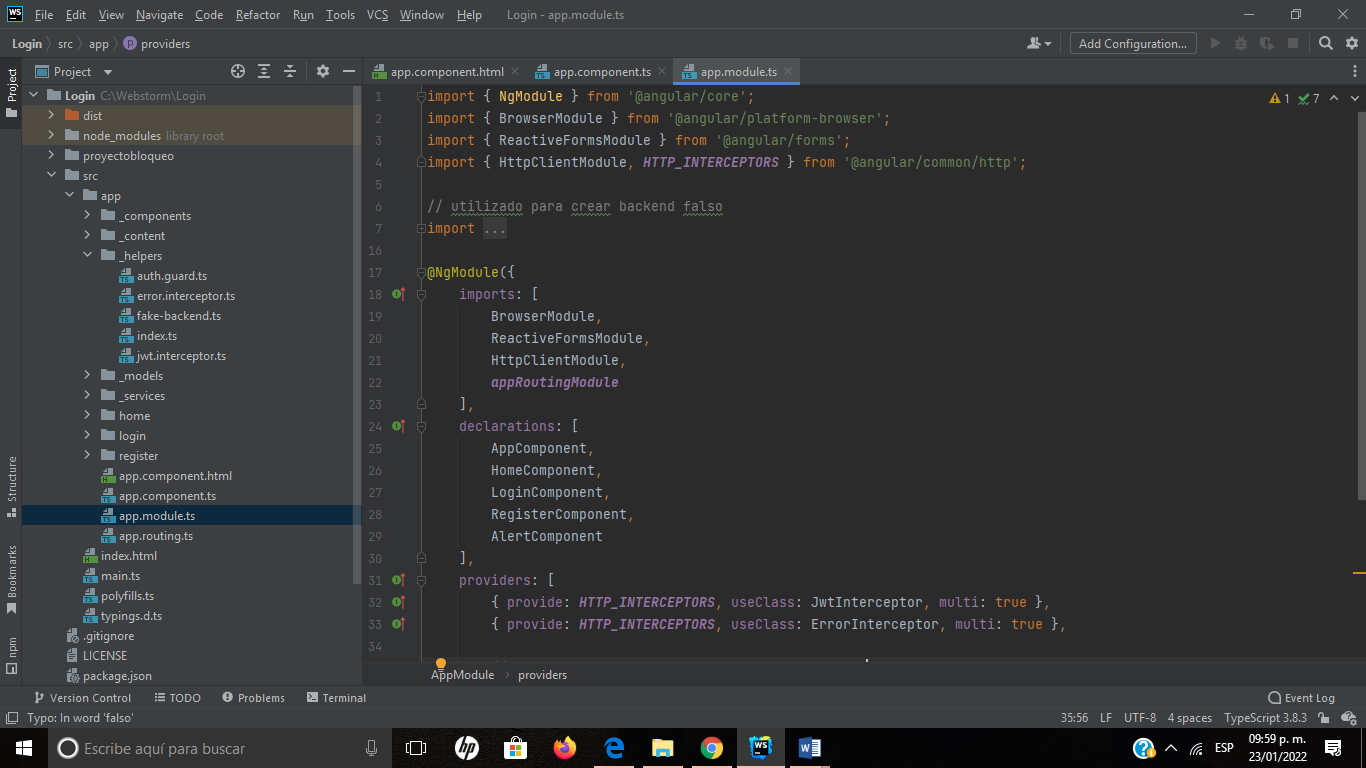
*App.component.html*

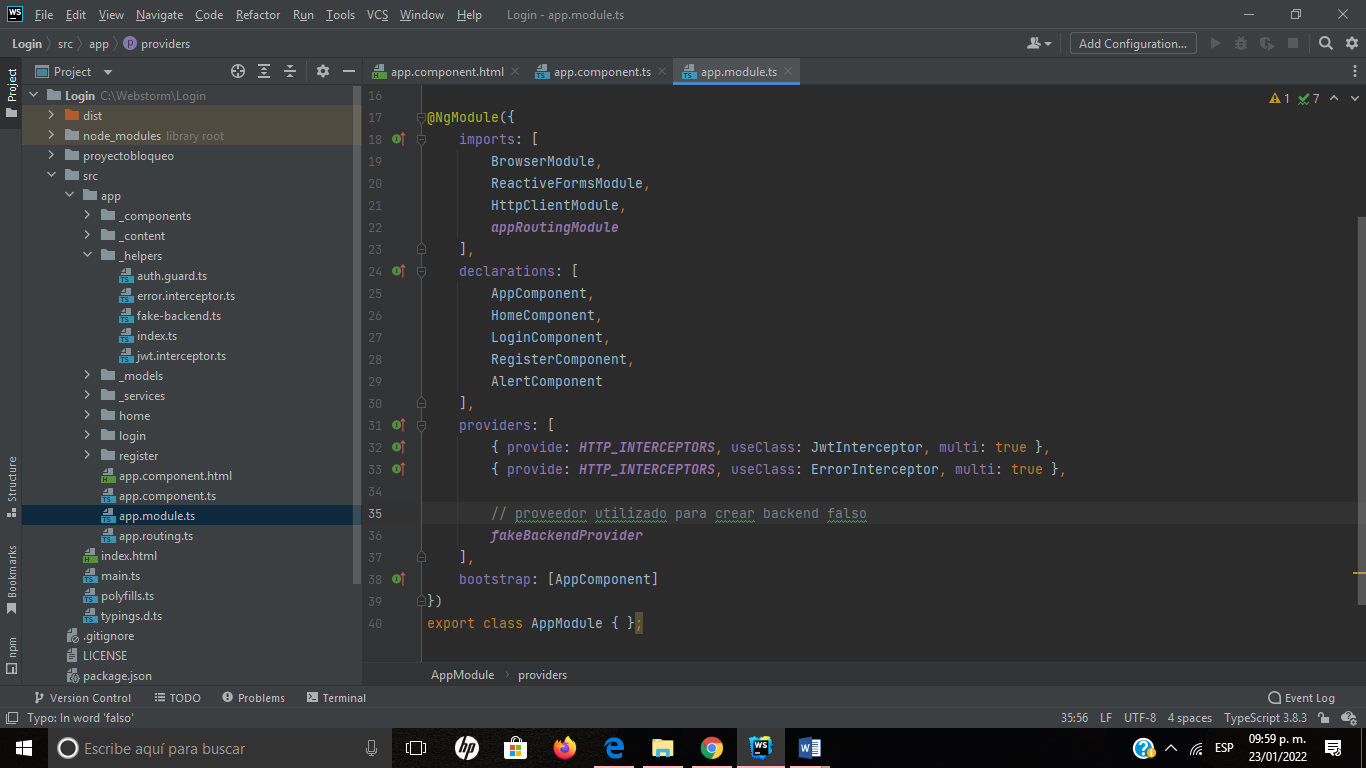


*App.component.ts*



*App.module.ts*





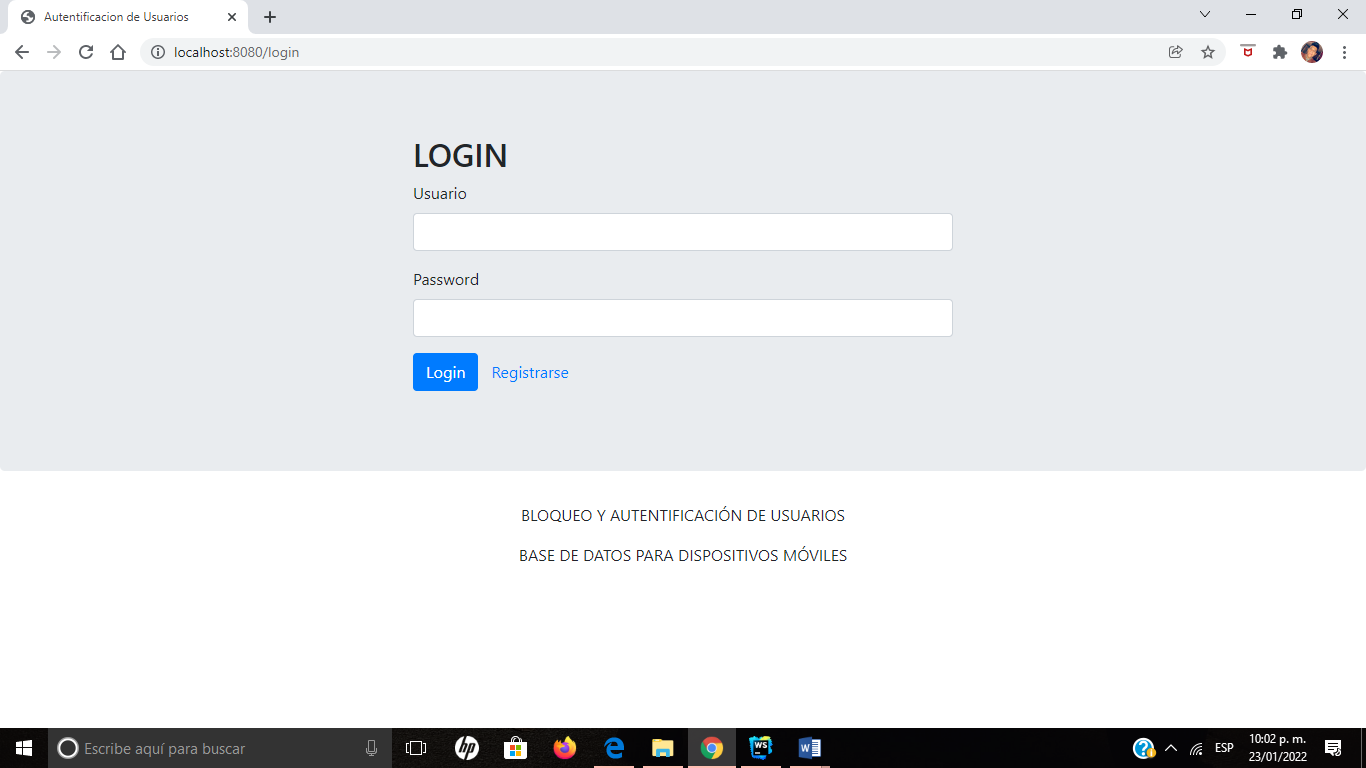
*App.routing.ts*



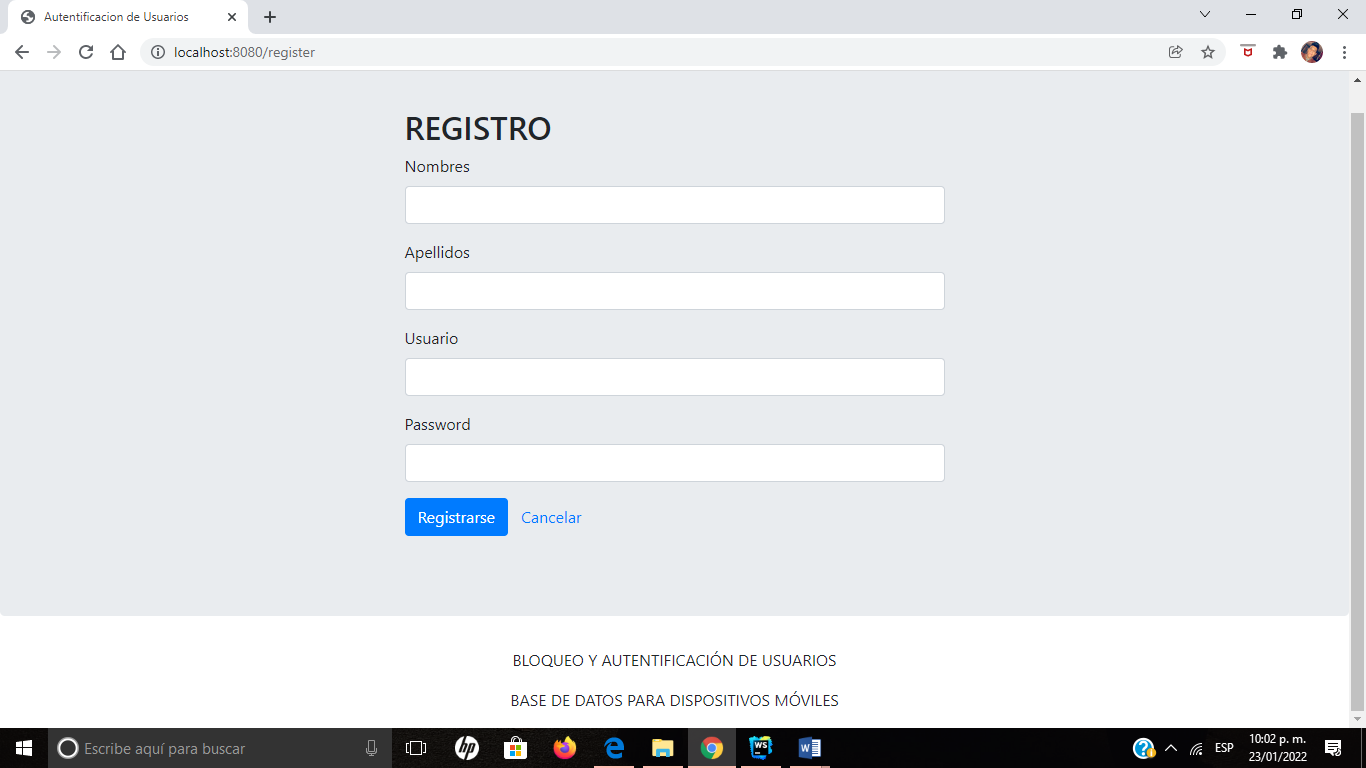
**FRONTEND**

En este apartado mostramos la interfaz con la que el usuario interactúa, sin identificar las funcionalidades. Para esto mostramos inicialmente las siguientes pantallas.

**INTERFAZ LOGIN**

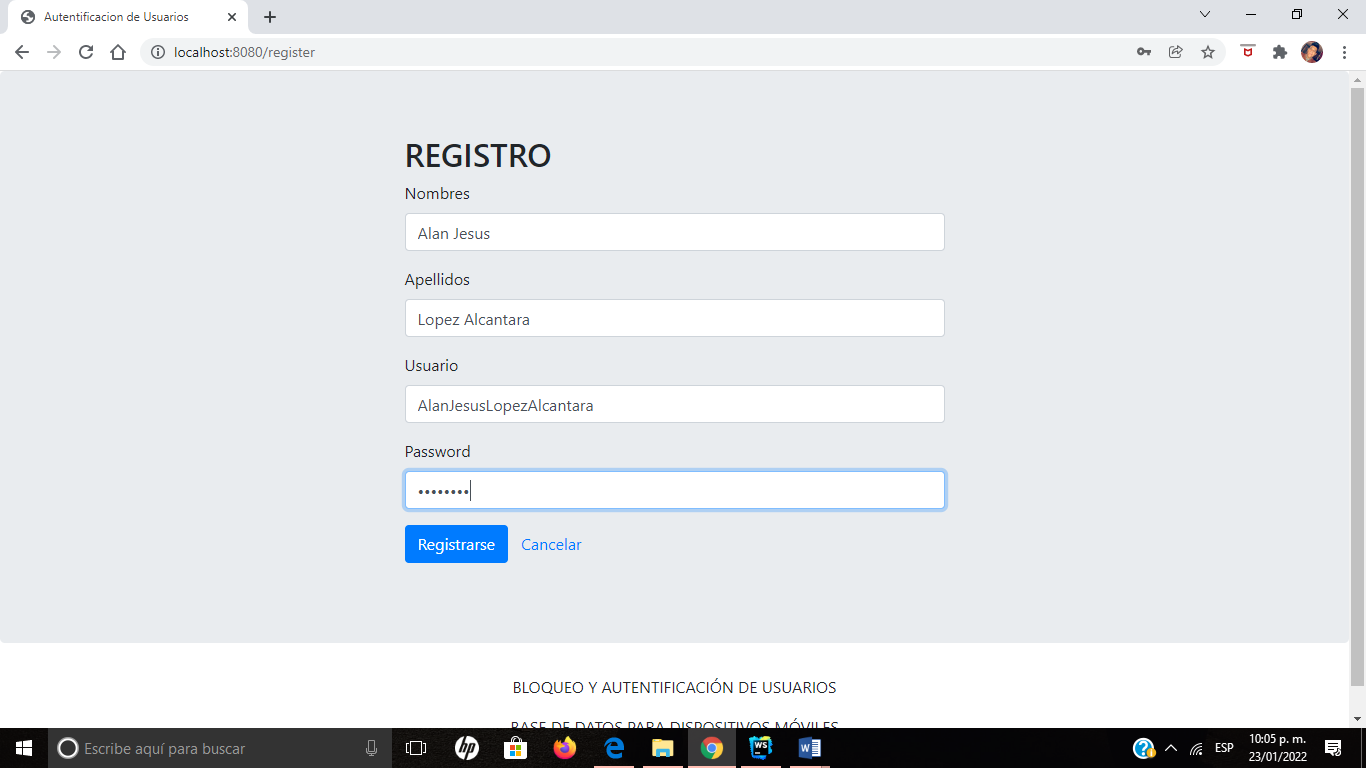


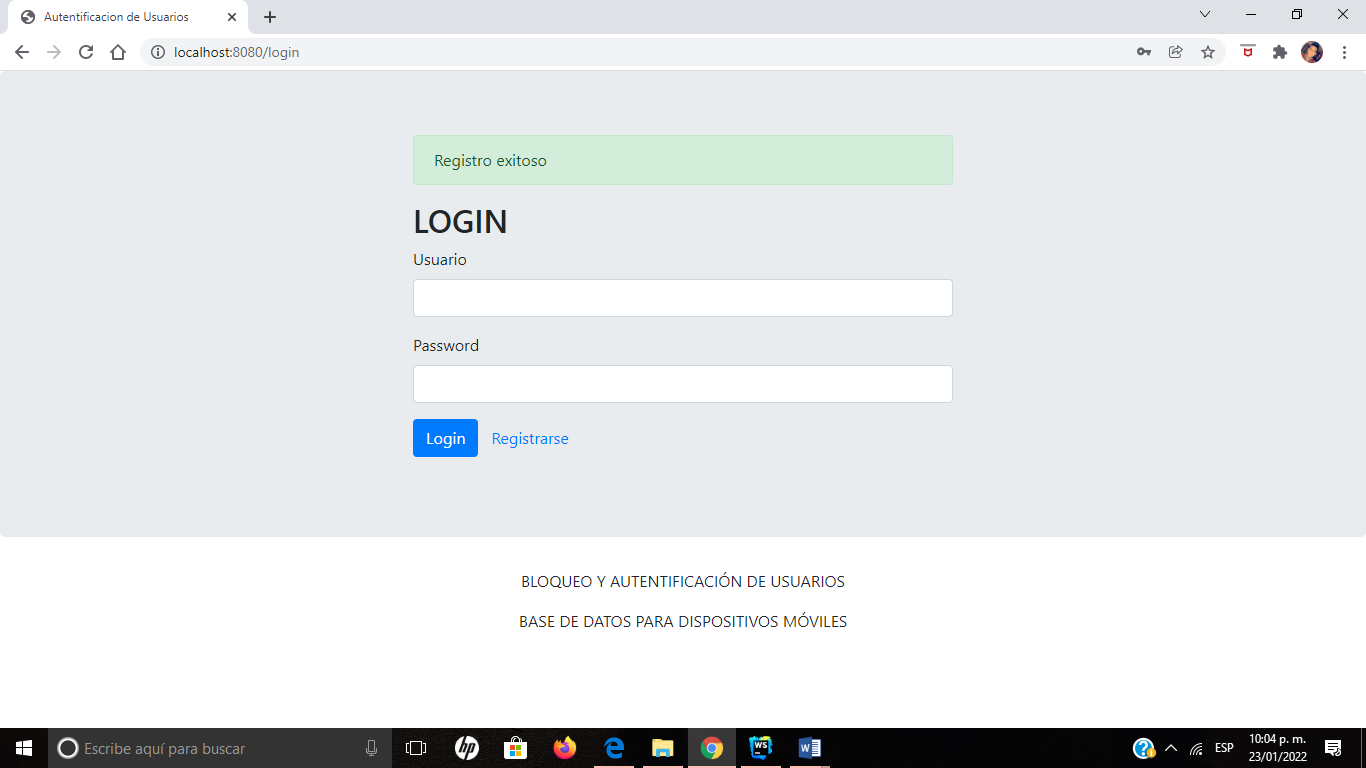
**INTERFAZ REGISTRO**



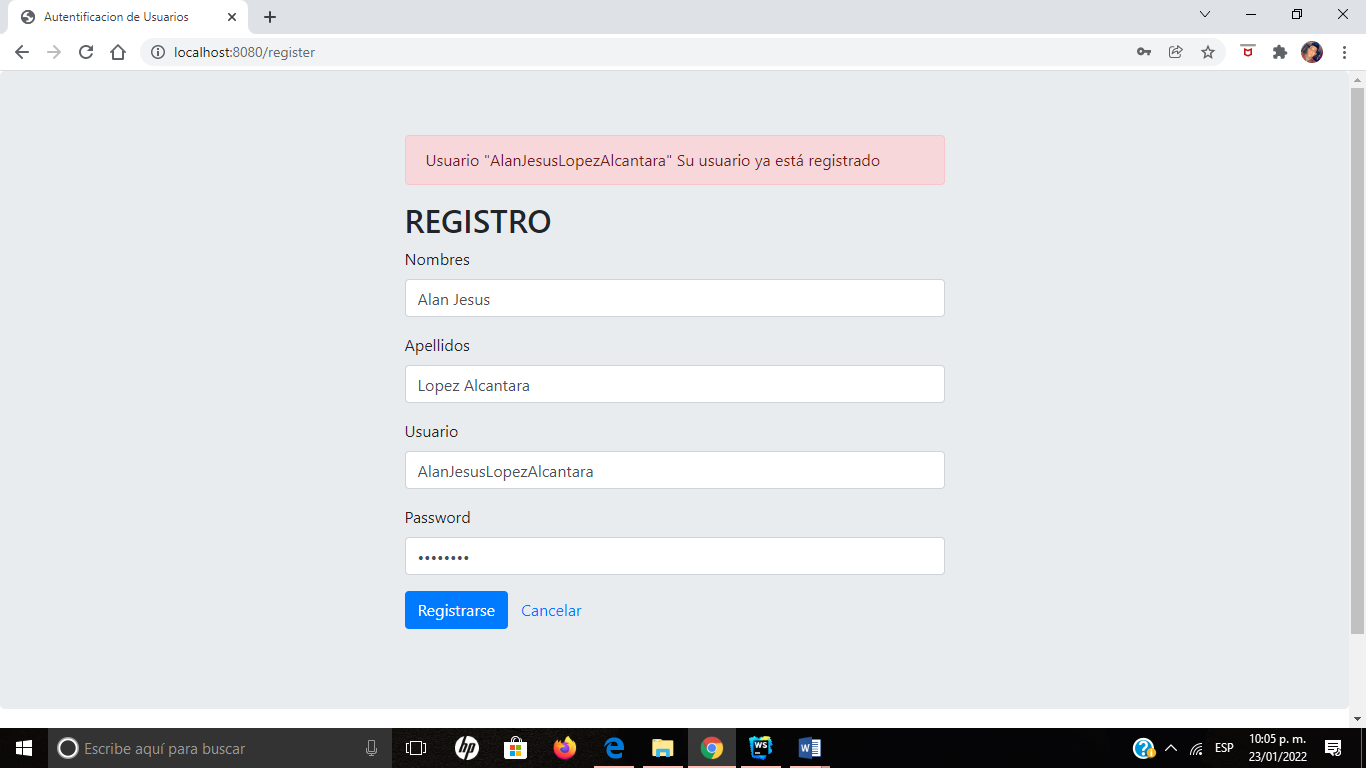
Estas pantallas nos muestran los campos de acceso y de registro al sistema. Ahora identificaremos el funcionamiento mediante una prueba de registro de un usuario.

**REGISTRO USUARIO**

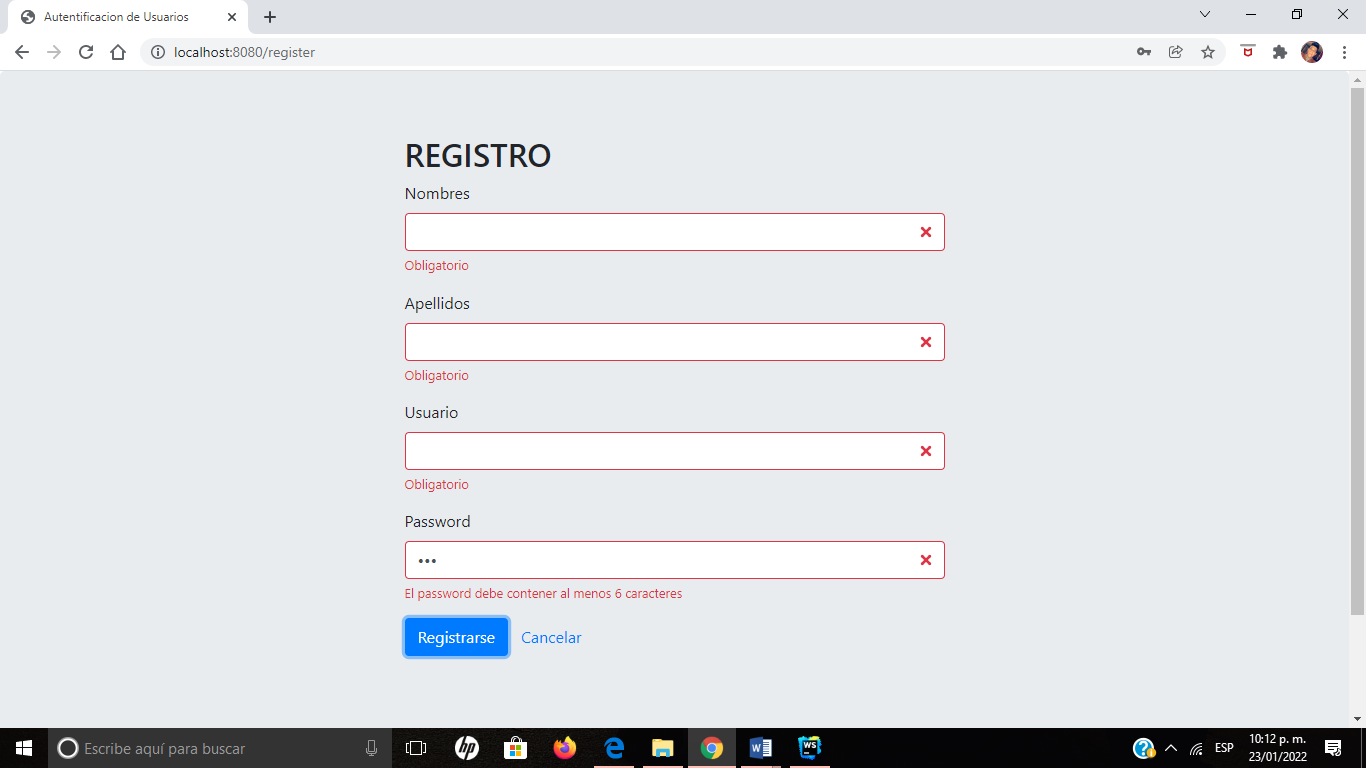




**REGISTRO DE USUARIO YA ALMACENADO**

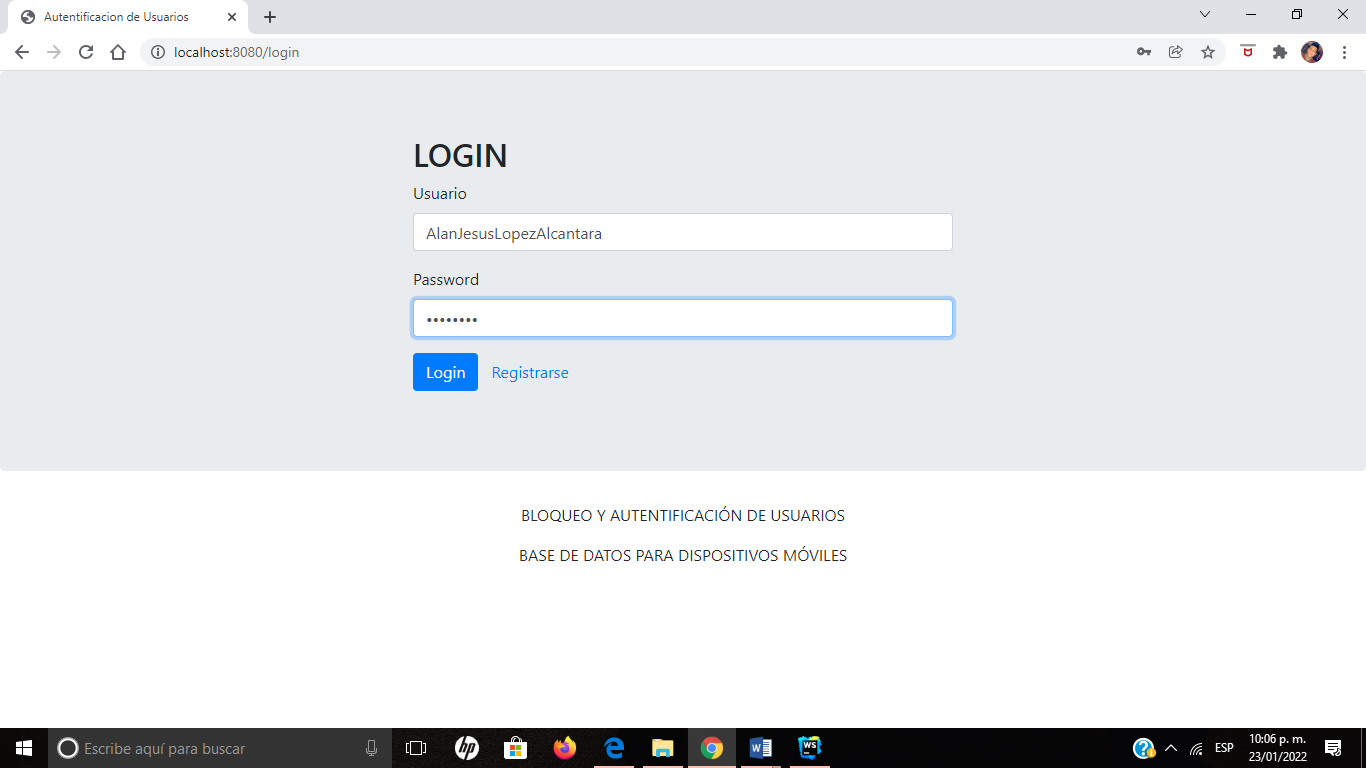


**ALERTAS DE REGISTRO**

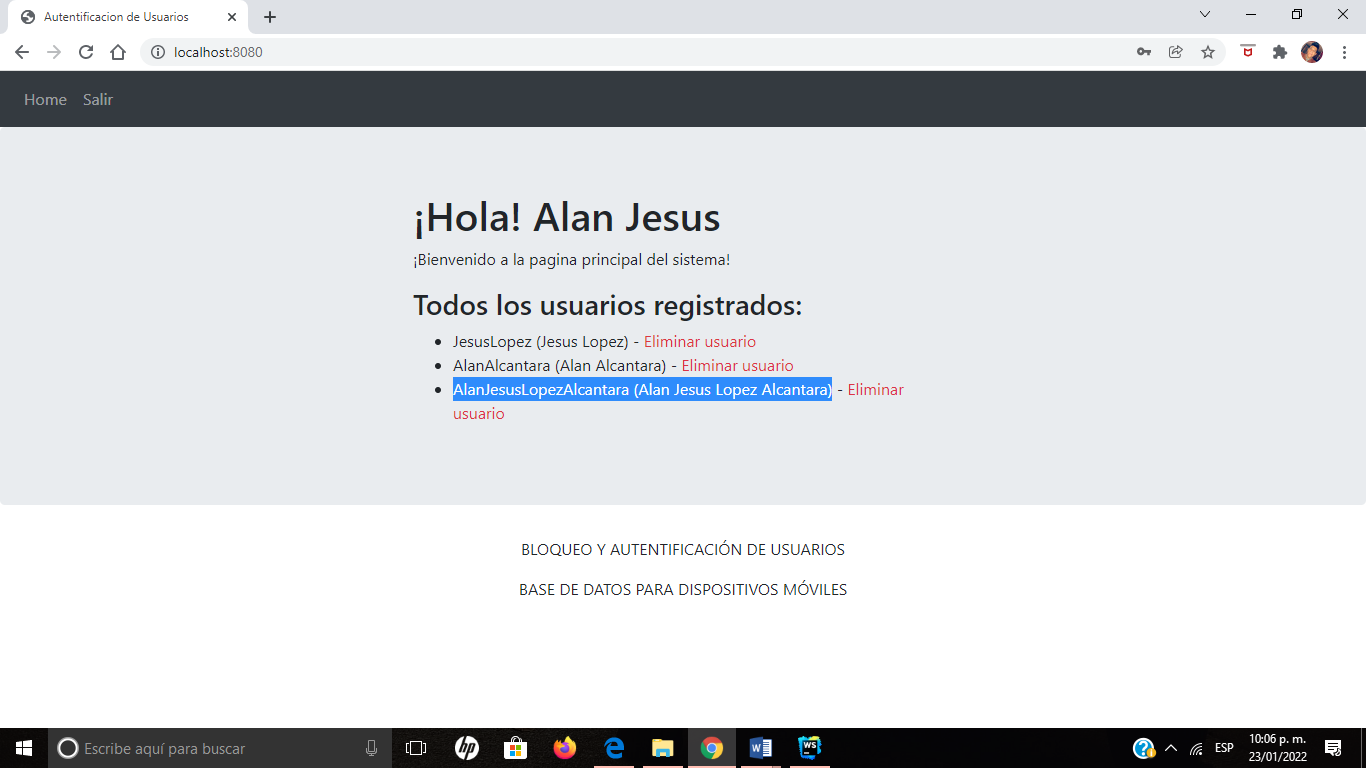


Ahora comprobaremos que el acceso puede darse mediante el Login.

**LOGIN DE USUARIO REGISTRADO**

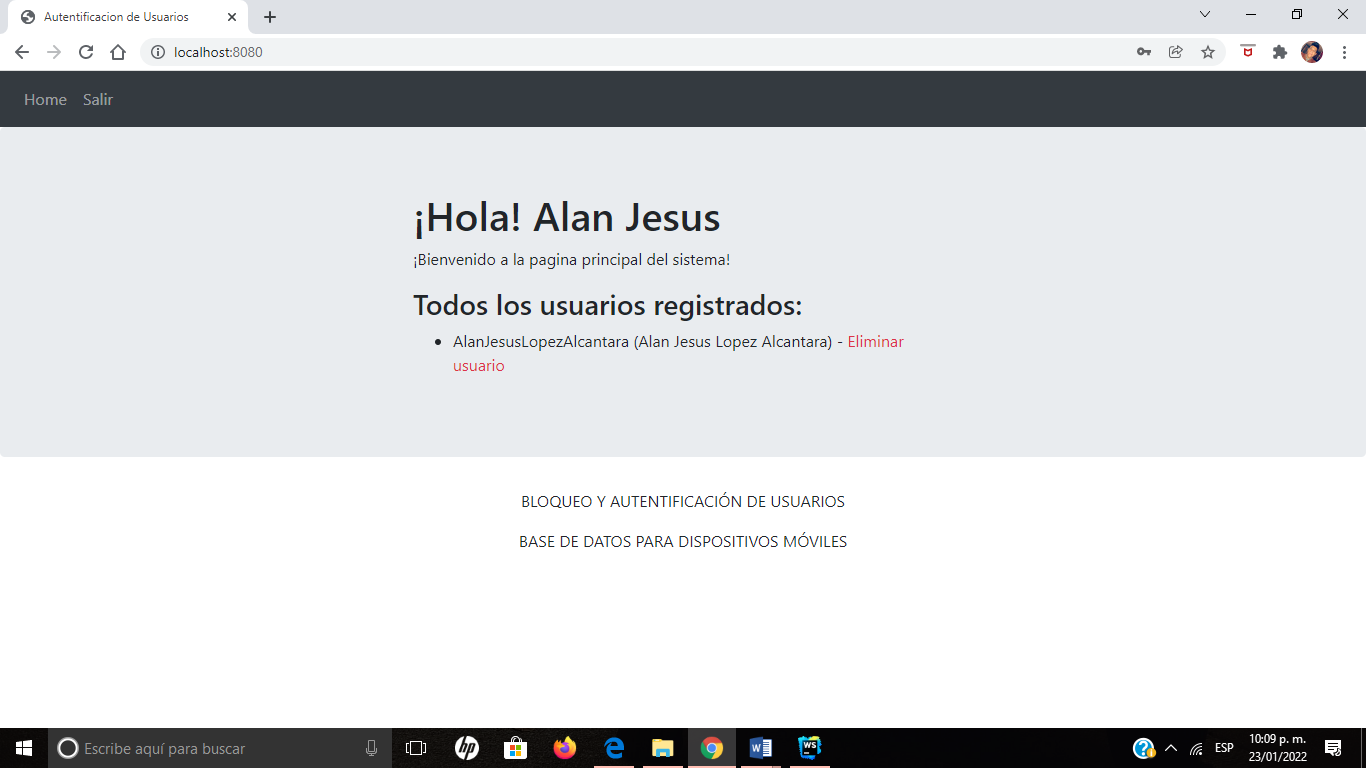


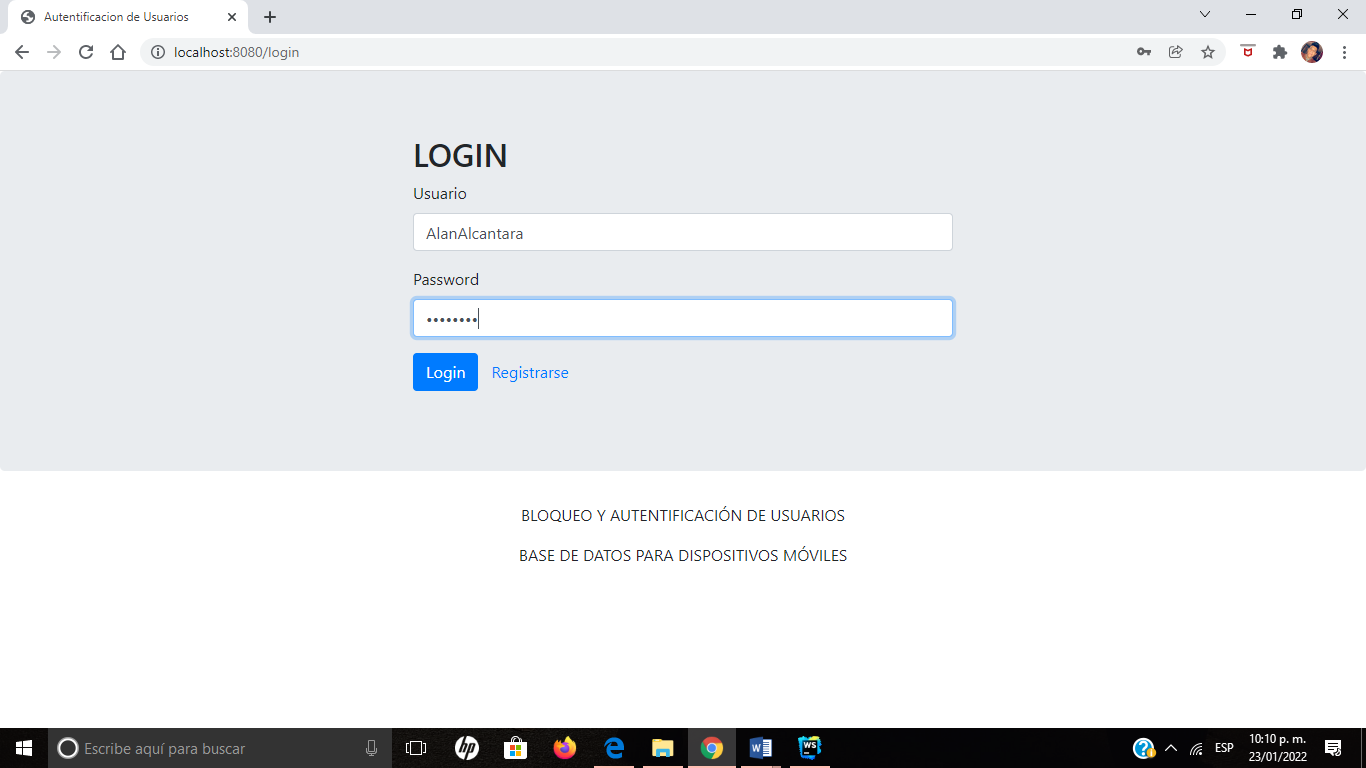
En la siguiente pantalla podemos comprobar que el acceso a la ruta mediante el Login fue exitoso y de igual manera se almacenaron tanto nuestros datos de usuario, como los nombres y Apeidos. (Aparecen tres registros por pruebas preliminares).

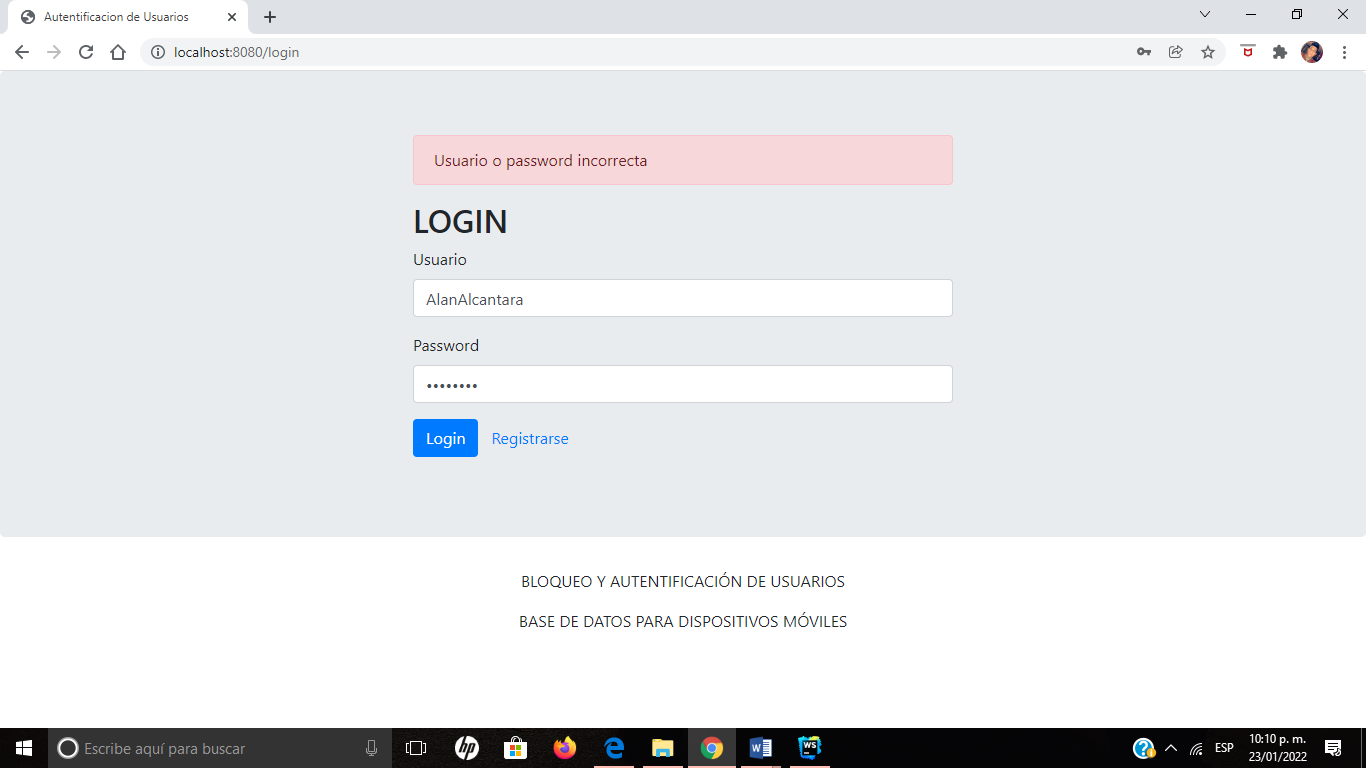


Pero al dar clic en Eliminar Usuario podemos sacarlo del sistema. Y posteriormente al intentar acceder por medio de este mismo el sistema solicitará que vuelva a registrarse.

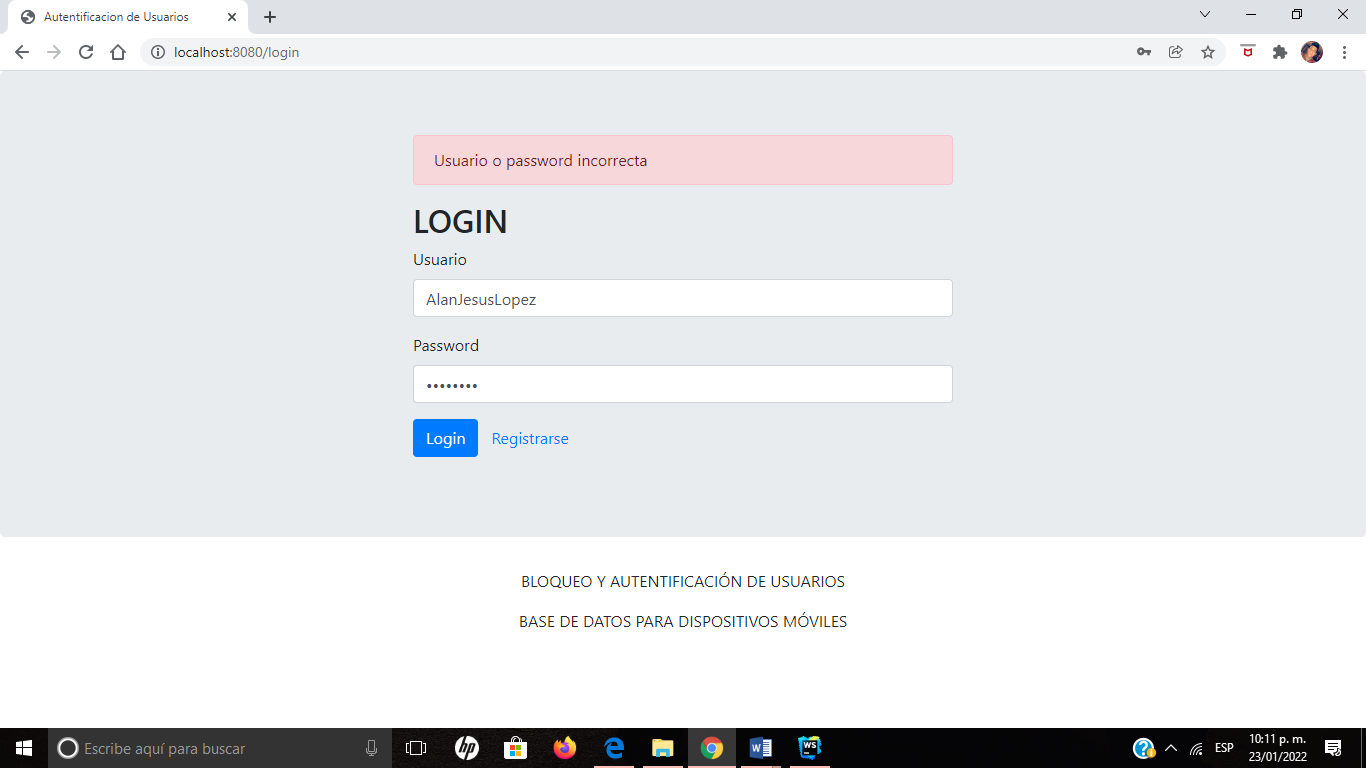
**ELIMINACIÓN DE USUARIO**

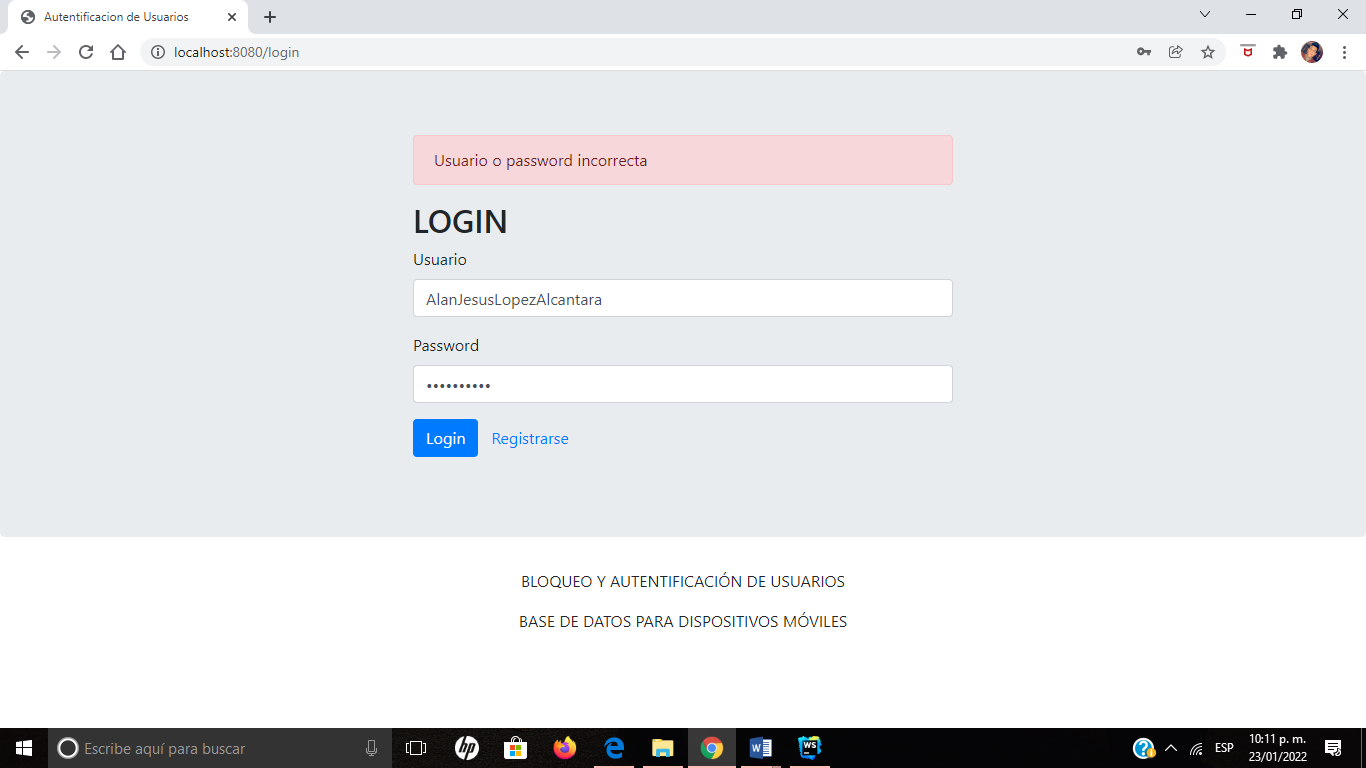






**ACCESO USUARIO REGISTRADO CON PASSWORD O USUARIO INCORRECTO**





**CONCLUSIONES**

Finalmente podemos concluir que la implementación de la protección de las rutas, fue un tanto más eficaz por medio de la autentificación de los usuarios por medio de los parámetros necesarios. La creación del Login o del formulario de registro no eran módulos que correspondían a nuestro equipo, sin embargo para poder ejemplificar la funcionalidad de la autentificación de un GUARD, era importante contar con estos módulos para que el frontend fuera una interfaz más asemejada a un contexto de la vida diaria.

La implementación de estas autentificaciones tiene que ver con la extracción de los datos (POST) y posteriormente el análisis y comparación de los mismos (GET), es decir la realización de una consulta que verifique que los datos de los parámetros requeridos son los mismos y que mediante esto se puede o no tener acceso a una ruta específica, en este caso el HOME, también se permite al tener acceso al HOME la eliminación de los usuarios (DELETE), todos estos mismos casos que usamos en prácticas anteriores con POSTMAN.

El añadir un GUARD, especifica la autentificación de estos datos por medio de la consulta y actúa como un condicional que retorna valores booleanos, es decir que si es verdadero se puede dar acceso al usuario, de lo contrario no tendrá acceso al sistema y puede corresponder a su bloqueo inmediato.

Era importante el ligar este módulo con MONGO DB, sin embargo nos pareció más práctico englobar todas las funcionalidades en Webstrorm y administrarlas desde Angular.

La funcionalidad dio resultado y nos permitió agregar, administrar y eliminar usuarios. Así como asignar permisos y protección a las rutas solo para usuarios registrados en el sistema. Inicialmente teníamos una idea distinta en la formación de la base de datos mediante la creación de una tabla de bloqueo, pero para ello era necesario contar con los módulos correspondientes y complicaba más el desarrollo de la implementación así que decidimos trabajar específicamente en el GUARD y la autentificación.