hiberus
TECNOLOGIA

Xerach Casanova Cabrera

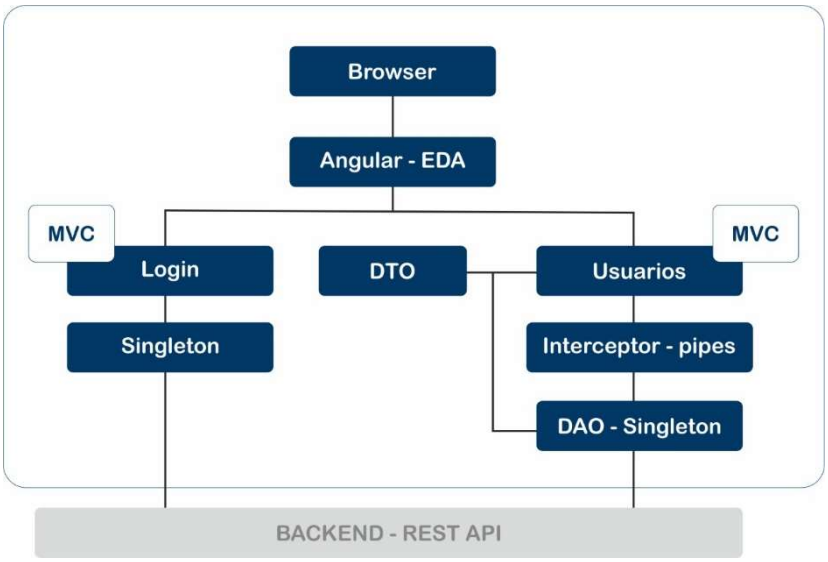
Práctica para Hiberus

Javascript

Crecemos contigo

Contenido

1. ARQUITECTURA



Para este proyecto las decisiones sobre la arquitectura del mismo están tomadas en base a que sea un proyecto totalmente escalable, con vistas a que la aplicación pueda ir creciendo con el tiempo de una manera ordenada y fácil de modificar y ampliar.

Para ello, se ha decidido seguir una arquitectura de tipo MVC (modelo vista controlador) para cada uno de los componentes que conforman la aplicación. A su vez, la interfaz de usuario se implementará con una arquitectura orientada a Eventos. Ambas arquitecturas vienen implícitas en la metodología de trabajo que dispone Angular, el cual será el framework elegido para realizar este proyecto, en conjunto con la librería Material para el diseño de web.

Este proyecto cuenta con dos módulos principales:

LOGIN MODULE

Este módulo utilizará un servicio singleton para despachar las peticiones http hacia el backend: tanto el componente de login, como el de registro, así como un método en el propio servicio para comprobar si el usuario está autenticado. A pesar de poder utilizar un interceptor de autenticación para el módulo, en este caso, debido a su simplicidad, nos decantamos por un servicio que se encargue de resolverlo.

USERS MODULE

Este módulo cuenta con todos los componentes que tendrá la aplicación para la gestión de usuarios. En caso de que la aplicación pudiese crecer en un futuro, este sería el módulo más propenso a recibir ampliaciones y modificaciones, en base a esto, se han tomado las siguientes decisiones:

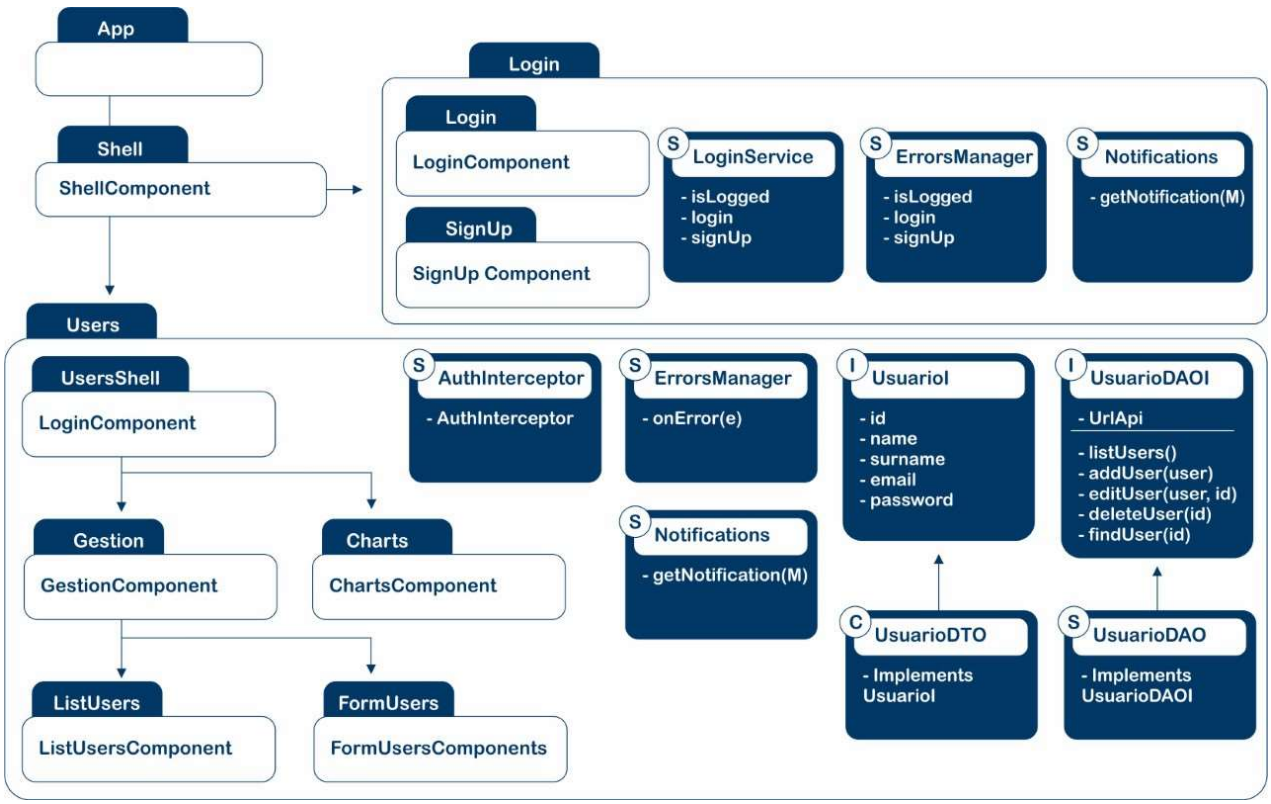
Creación de una interface de usuario que a su vez sea implementada por la clase Usuario. De esta manera, conforme vaya creciendo la aplicación, nos aseguraremos que se cumpla con el contrato que la propia

interface Usuario nos ofrece. A su vez, la clase Usuario podrá ampliar su funcionalidad en un futuro si fuese necesario, sin tener que modificar la propia interface.

De la misma manera, se ha decidido utilizar un servicio DAO de usuario que a su vez implemente su propia interface. Este servicio será el encargado de realizar toda la lógica de negocio para conectar la vista con el modelo. Al implementar su propia interface, nos aseguramos que cualquier cambio en el servicio (por ejemplo, si se decidiese cambiar el backend en un futuro o si el mismo recibiese modificaciones), no afectará al funcionamiento de la vista, logrando así un bajo acoplamiento de la aplicación al completo.

Por último, cabe mencionar la utilización de un servicio interceptor que utilizará el servicio DAO del módulo para realizar las peticiones http asegurando la autenticación de usuario. De esta manera, nos aseguraremos que todos los componentes del módulo utilicen dicho servicio sin tener que implementarlo en cada uno de ellos.

2. DIAGRAMA DE CLASES



APP

Contiene un único componente que se encarga de cargar la Shell que contiene toda la estructura de la aplicación así como su enrutador. No contendrá ningún tipo de lógica.

SHELL

Contiene un único componente con la estructura de la navegación y barra de herramientas, dentro de la cual se cargarán los distintos componentes de la aplicación definidos en su propio enrutador.

LOGIN

Módulo que contiene a su vez dos componentes: Login y Regiter.

Servicios

- LoginService: Se encarga de realizar las peticiones http al servidor, tanto para autenticación, como para realizar un nuevo registro.
- ErrorsManager: gestión de los distintos errores de las peticiones http al servidor.
- Notifications: Servicio encargado de recibir las distintas notificaciones que se mostrarán al usuario a partir de su interacción con la app.

USERS

Módulo que contiene su propio enrutador para los siguientes componentes:

- Shell: contiene el menú navegación específico del módulo completo, incluyendo el botón Logout, encargado de eliminar el token de la sesión de usuario para desconectarlo de la aplicación.
- Gestion: se encarga de comunicar el componente Form-Users con List-Users y viceversa.
- Form-Users: contiene el formulario encargado de la edición y creación de nuevos usuarios.
- List-Users: contiene el listado completo de los usuarios.
- Charts: contiene estadísticas variadas de la gestión de usuarios.

Interfaces

- UsuarioI: Interface que será implementado por la clase UsuarioDTO.
- UsuarioDAOI: Interface que será implementado por el servicio UsuarioDAO.

Servicios

- UsuarioDAO: Servicio encargado de realizar las peticiones http al backend para gestionar los registros de la base de datos con los métodos de listar, añadir, modificar, eliminar y buscar, los cuales están designados en la interface UsuarioDAOI y que este servicio se encarga de implementar.
- ErrorsManager: gestión de los distintos errores de las peticiones http al servidor.
- Notifications: Servicio encargado de recibir las distintas notificaciones que se mostrarán al usuario a partir de su interacción con la app.

Clases

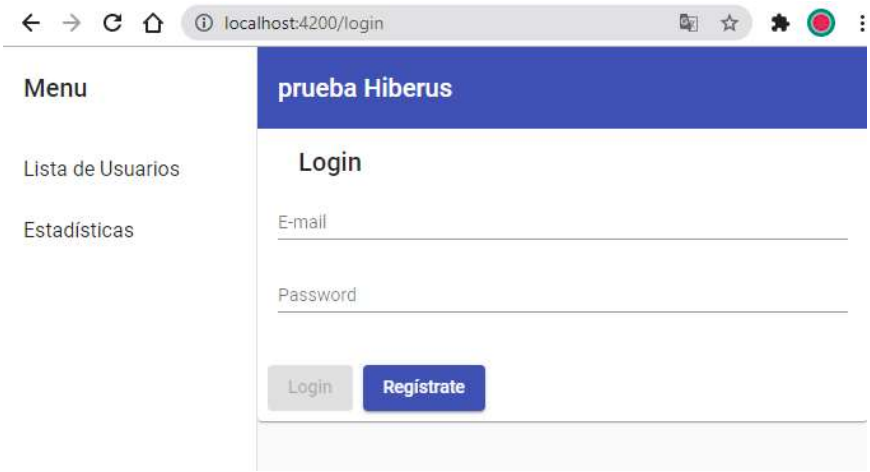
UsuarioDAO: implementa la interface UsuarioI, la cual contiene las propiedades y métodos que debe tener el objeto usuario dentro del uso de los distintos componentes del módulo Users.

3. Pantallas

LOGIN

Contiene un formulario para autenticarse en la app con los siguientes elementos:

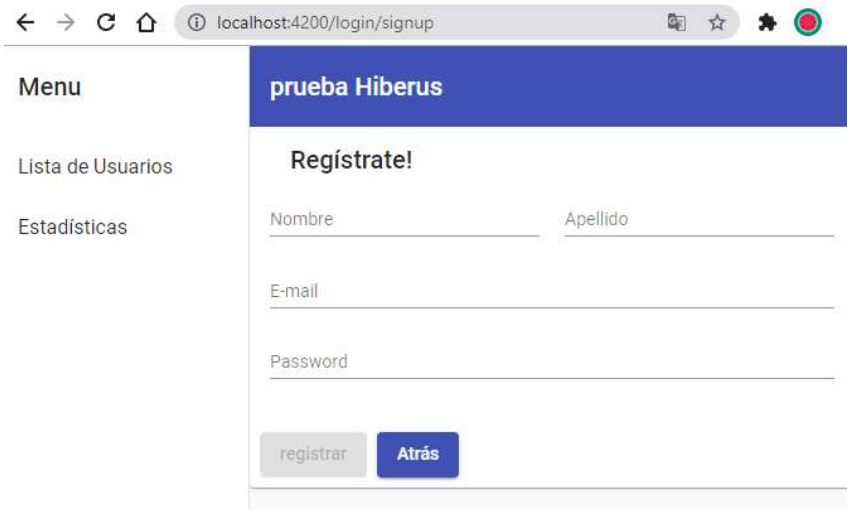
- Campo email (requerido)
- Campo password (requerido)
- Botón enviar.
-



SIGNUP

Contiene un formulario para registrarse en la app con los siguientes elementos:

- Campo email (requerido)
- Campo password (requerido)
- Campo name (requerido)
- Campo surname (requerido, min. Length: 4 caracteres)
- Botón registrar
- Botón atrás (volver a pantalla login)

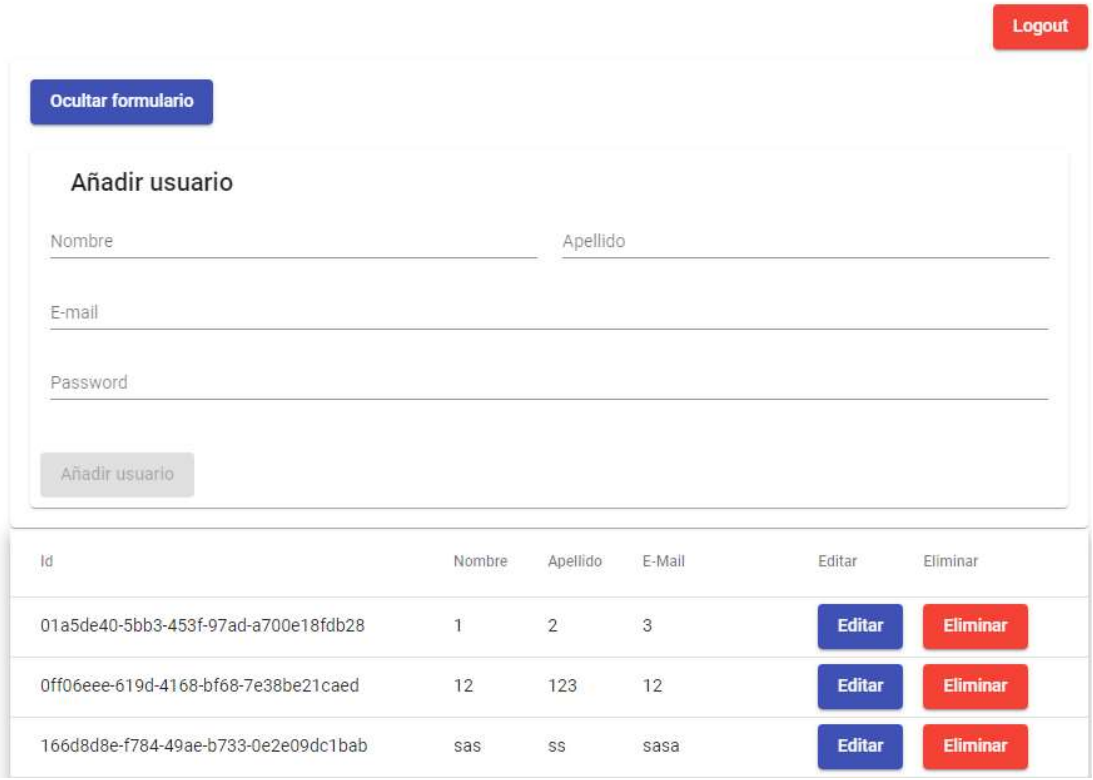


USUARIOS

Contiene dos componentes:

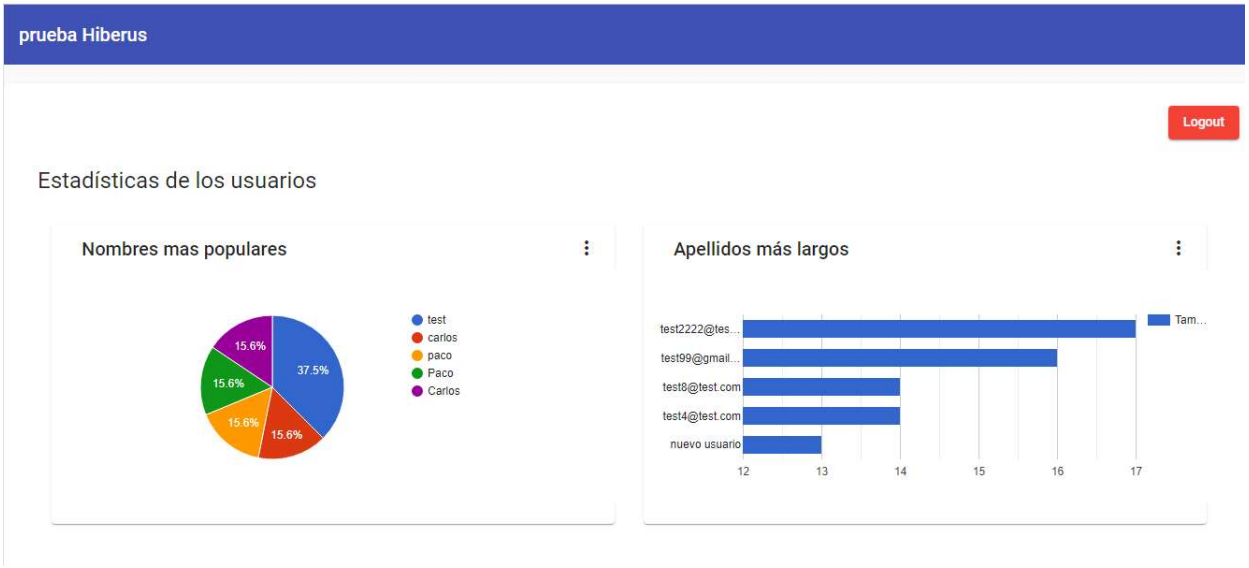
- users-form para dar en la app con los siguientes elementos:
  - Botón añadir usuario / ocultar formulario (desplegar u ocultar el formulario)
  - Campo email (requerido)
  - Campo password (requerido)
  - Campo name (requerido)
  - Campo surname (requerido, min. Length: 4 caracteres)
  - Botón añadir/modificar usuario
  - Botón cancelar edición (solo disponible en modo edición de usuario)

- users-list, diseñado con un schematic de tipo table que proporciona Material Angular para realizar tablas, la cual contiene los datos de los usuarios de la base de datos. Cada uno de los usuarios tiene su propio botón para editar y otro para eliminar.



CHARTS

El componente charts con un schematic de tipo dashboard que proporciona Material Angular para la gestión de cards. Cada una de esas cards mostrará una estadística distinta a partir de los datos de los usuarios proporcionados por la base de datos. Para la gestión de las estadísticas utilizaremos Google Charts.



4. ASPECTOS A TENER EN CUENTA

- La convención de nomenclaturas elegido para este proyecto es el estándar de codificación de java.
- La decantación por la librería de Angular Material para este proyecto se debe a que es la recomendada por Google para trabajar con proyectos basados en el framework de Angular.