| **Proyecto Frontend Angular (10)** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Criterios** | **Nulo**  AUSENTE  **0 PUNTOS** | **Bajo**  CONFUSO  **2 PUNTOS** | **En proceso**  FALTA DE PROFUNDIDAD  **3 PUNTOS** | **Correcto**  PROBLEMAS PUNTUALES  **4 PUNTO** | **Óptimo**  CLARO  **5 PUNTOS** | **NOTA** |
| Aspecto funcional | No se implementa un componente de login para acceder con cuenta de cliente o administrador.  El proyecto presentado no se adecúa a lo estipulado. | Se Implementa un componente de login para acceder con cuenta de cliente o administrador.  El menú lateral muestra las opciones correspondientes a cada perfil:   * admin: alumnos, cursos, inscripciones, usuarios, logout * usuario: alumnos, cursos, inscripciones, logout   Se trabaja con un backend en mockapi o similar pero su operación está incompleta. | Se Implementa un componente de login para acceder con cuenta de cliente o administrador.  El menú lateral muestra las opciones correspondientes a cada perfil:   * admin: alumnos, cursos, inscripciones, usuarios, logout * usuario: alumnos, cursos, inscripciones, logout   El toolbar superior muestra el nombre de la aplicación, el título del componente según la ruta y el nombre del usuario logueado  Las opciones de menú usuarios, cursos, y alumnos muestran los datos en una tabla cuando se seleccionan  Una vez logueado el usuario sólo se presenta la vista según su perfil.  El perfil cliente sólo lista las películas disponibles para su alquiler. No se presentan opciones en el carrito para borrar películas en forma individual o total.  Cada registro de la tabla tendrá una columna de acciones que el usuario podrá seleccionar sobre ese registro. Para todos serían la de modificar, eliminar, ver más detalle. Las opciones de modificar y eliminar sólo se deberán mostrar con un usuario con el rol de administrador | Se Implementa un componente de login para acceder con cuenta de cliente o administrador.  El menú lateral muestra las opciones correspondientes a cada perfil:   * admin: alumnos, cursos, inscripciones, usuarios, logout * usuario: alumnos, cursos, inscripciones, logout   El toolbar superior muestra el nombre de la aplicación, el título del componente según la ruta y el nombre del usuario logueado  Las opciones de menú usuarios, cursos, y alumnos muestran los datos en una tabla cuando se seleccionan  Cada registro de la tabla tendrá una columna de acciones que el usuario podrá seleccionar sobre ese registro. Para todos serían la de modificar, eliminar, ver más detalle. Las opciones de modificar y eliminar sólo se deberán mostrar con un usuario con el rol de administrador  La opción de ver detalle de un alumno se mostrará los datos del alumno más los cursos a los cuales está inscripto. Desde allí se podrá des inscribir de un curso a un alumno  La opción de logout deberá desologuear al usuario y llevarme a login | Se Implementa un componente de login para acceder con cuenta de cliente o administrador.  El menú lateral muestra las opciones correspondientes a cada perfil:   * admin: alumnos, cursos, inscripciones, usuarios, logout * usuario: alumnos, cursos, inscripciones, logout   El toolbar superior muestra el nombre de la aplicación, el título del componente según la ruta y el nombre del usuario logueado  Las opciones de menú usuarios, cursos, y alumnos muestran los datos en una tabla cuando se seleccionan  Cada registro de la tabla tendrá una columna de acciones que el usuario podrá seleccionar sobre ese registro. Para todos serían la de modificar, eliminar, ver más detalle. Las opciones de modificar y eliminar sólo se deberán mostrar con un usuario con el rol de administrador  La opción de ver detalle de un alumno se mostrará los datos del alumno más los cursos a los cuales está inscripto. Desde allí se podrá des inscribir de un curso a un alumno  La opción de ver detalle de un curso se mostrará los datos del curso más los alumnos a los cuales está inscripto. Desde allí se podrá des inscribir de un curso a un alumno  La opción de logout deberá desologuear al usuario y llevarme a login |  |
| Aspecto técnico | No presenta una definición y lógica en la estructura.  La aplicación no cuenta con un backend de datos o maneja los datos de forma local. | Se presentan inconsistencias en la estructura y la lógica del proyecto.  No se utiliza un servicio en la nube (ej: mockapi) para la implementación de la REST API del backend  No se puede comprobar el funcionamiento de las rutas del login y registro de usuarios.  No se implementa http interceptorpara manejo de peticiones http  No se implementan guardas en el router para las rutas que requieren autenticación.  Algunos los features modules se cargan en forma lazy loading, pero no todos  No implementa ningún test unitario  No implementa el módulo core  No implementa el guard | Se presenta en general con una estructura bien desarrollada. Se podrían realizar algunos ajustes en la lógica del proyecto.  Utiliza un servicio en la nube (ej: mockapi) para la implementación de la REST API del backend  Implementa todos los servicios de la aplicación utilizando REST API  Un feature módulo completo tiene sus componentes y servicios con unit test y estos ejecutan presentan errores  Implementa un módulo CORE con los servicios de autenticación  Algunos los features modules se cargan en forma lazy loading, pero no todos  Se presentan fallas en la implementación del guard o no están todas las implementaciones.  El guard no estn correctamente implementado en todas las rutas necesarias.  Implementa un feature module de Usuarios para el ABM de usuarios, similar al feature módulo de alumnos. Este módulo sólo se podrá acceder si el usuario logueado pertenece al role de admin. Esta validación se realiza con un gurad | La definición de la lógica utilizada y la estructura son correctas. Se podrían realizar mínimos ajustes para terminar de pulir.  Utiliza un servicio en la nube (ej: mockapi) para la implementación de la REST API del backend  Implementa todos los servicios de la aplicación utilizando REST API  Un feature módulo completo tiene sus componentes y servicios con unit test y estos ejecutan correctamente  Implementa un módulo CORE con los servicios de autenticación  Todos los features modules se cargan en forma lazy loading  Implementa un formulario de login que valida correctamente el ingreso  Implementa un guard que no permite acceder a las rutas de la aplicación si el usuario no está autenticado  Implementa un feature module de Usuarios para el ABM de usuarios, similar al feature módulo de alumnos. Este módulo sólo se podrá acceder si el usuario logueado pertenece al role de admin. Esta validación se realiza con un gurad | Contiene una excelente definición de la lógica utilizada y presenta una perfecta estructura.  Utiliza un servicio en la nube (ej: mockapi) para la implementación de la REST API del backend  Implementa todos los servicios de la aplicación utilizando REST API  Un feature módulo completo tiene sus componentes y servicios con unit test y estos ejecutan correctamente  Implementa un módulo CORE con los servicios de autenticación  Todos los features modules se cargan en forma lazy loading  Implementa un formulario de login que valida correctamente el ingreso  Implementa un guard que no permite acceder a las rutas de la aplicación si el usuario no está autenticado  Implementa un feature module de Usuarios para el ABM de usuarios, similar al feature módulo de alumnos. Este módulo sólo se podrá acceder si el usuario logueado pertenece al role de admin. Esta validación se realiza con un gurad |  |