# DOCUMENTACIÓN INTERFAZ GRÁFICA DE CONSULTA PARA EL PROYECTO METAMODELO PARA VINCULACIÓN DE RECURSOS EDUCATIVOS ABIERTOS MEDIANTE ESPECIFICACIONES LOD Y BASADO EN PRINCIPIOS DE CONFIANZA

## TECNOLOGIAS UTILIZADAS

* REACT

Es una biblioteca de JavaScript para construir interfaces de usuario, React permite crear interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado en la aplicación, y React se encargará de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien. React crea componentes encapsulados que manejen su propio estado, y conviértelos en interfaces de usuario complejas. Ya que la lógica de los componentes está escrita en JavaScript y no en plantillas, se puede pasar datos de forma sencilla a través de la aplicación y mantener el estado fuera del DOM. En React no dejamos fuera al resto de tus herramientas tecnológicas, así que puede desarrollar nuevas características sin necesidad de volver a escribir el código existente.1

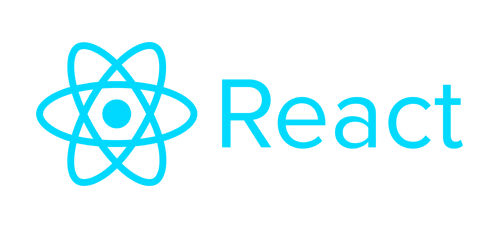


Figura 1

* TAILWIND CSS

Tailwind CSS es un framework CSS de bajo nivel y altamente personalizable que le brinda todos los bloques de construcción que necesita para crear diseños a medida sin ningún estilo de opinión molesto que tenga que luchar para anular.

En lugar de componentes prediseñados con opinión, Tailwind proporciona clases de utilidad de bajo nivel que le permiten crear diseños completamente personalizados sin tener que abandonar su HTML.

Cada utilidad Tailwind también viene con variantes receptivas, lo que hace que sea extremadamente fácil construir interfaces receptivas sin recurrir a CSS personalizado.

Si bien puede hacer mucho con solo clases de utilidad, a medida que crece un proyecto, puede ser útil codificar patrones comunes en abstracciones de nivel superior. Tailwind proporciona herramientas para extraer clases de componentes de patrones de utilidad repetidos, lo que facilita la actualización de múltiples instancias de un componente desde un solo lugar.2



Figura 2

### YARN

Yarn es un administrador de paquetes que también funciona como gerente de proyectos. Ya sea que trabaje en proyectos de una sola vez o grandes monorepos, como aficionado o usuario empresarial, lo tenemos cubierto.3



Figura 3

### NODE JS

Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables. Node.js es similar en diseño y está influenciado por sistemas como Event Machine de Ruby y Twisted de Python. Pero Node.js lleva el modelo de eventos un poco más allá. Incluye un bucle de eventos como runtime de ejecución en lugar de una biblioteca. En otros sistemas siempre existe una llamada de bloqueo para iniciar el bucle de eventos. Por lo general, el comportamiento se define mediante devoluciones callbacks de llamada al iniciarse un script y al final se inicia un servidor a través de una llamada de bloqueo como EventMachine::run(). En Node.js, no existe como tal la llamada de inicio del evento de bucle o start-the-event-loop. Node.js simplemente entra en el bucle de eventos después de ejecutar el script de entrada y sale cuando no hay más devoluciones callbacks de llamada para realizar. Se comporta de una forma similar a JavaScript en el navegador - el bucle de eventos está oculto al usuario.

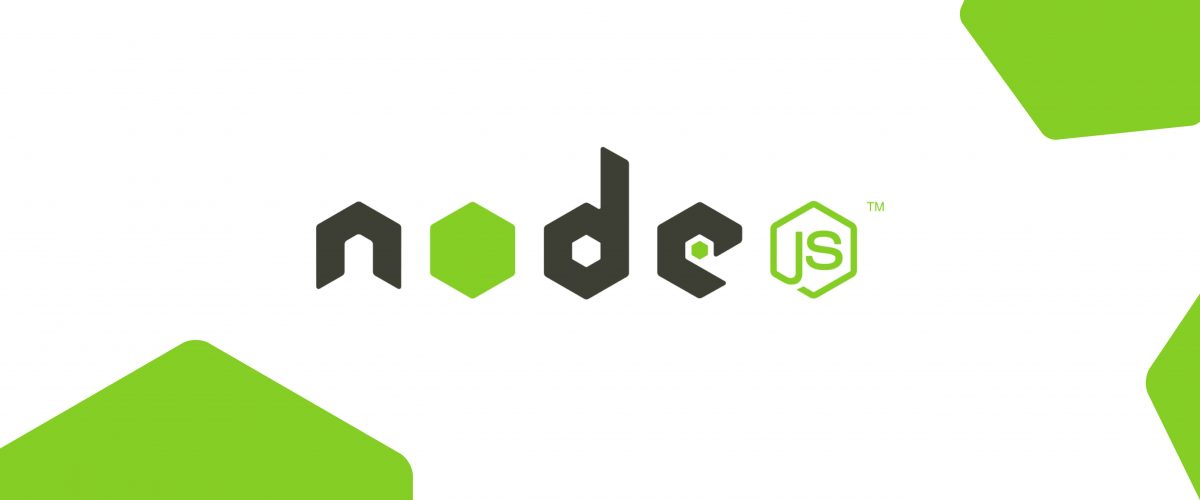


Figura 4

## DISEÑO

### CAJAS DE CONTENIDO

Cajas de contenido con sombras y márgenes de separación con otras cajas, se dividen en Header, caja de consulta y caja de tablas. La caja Header contiene el título de la interfaz y el proyecto tal y como se ve en la figura 5.

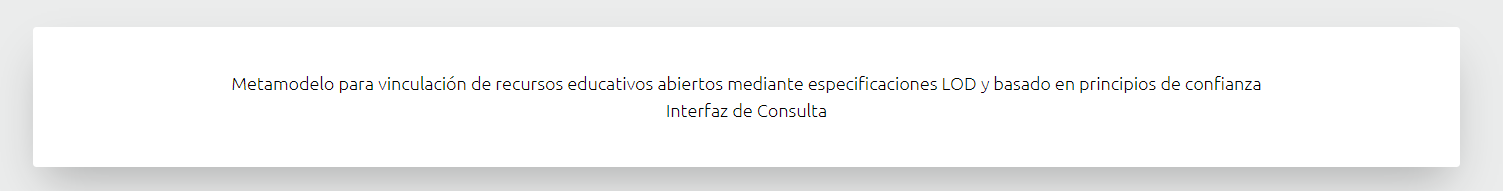


Figura 5

La caja de consulta contiene un menú de selección con valor por defecto ninguno, un recuadro para ingresar el término a buscar, y el botón de búsqueda.

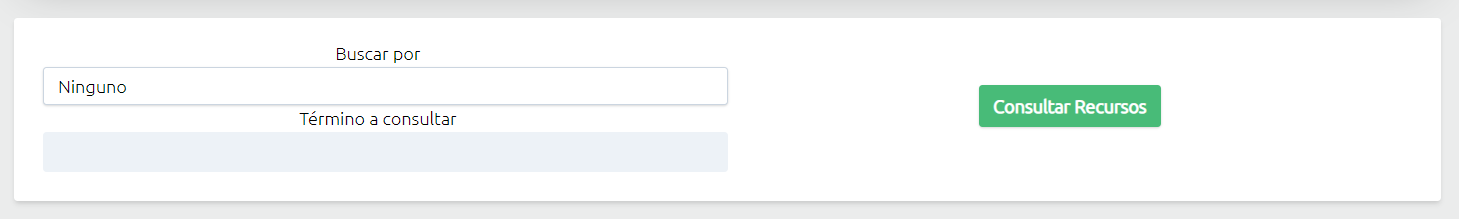


Figura 6

La caja de tablas es el lugar donde se muestran los resultados de las consultas, si no hay consultas ejecutadas se mostrara el mensaje “No hay datos que mostrar”, en caso contrario se cargara la tabla de resultados. En la parte superior derecha hay un recuadro para ingresar la palabra por la cual se filtrarán las tablas.



Figura 7

Una vez se carguen los datos en la división donde se genera la tabla también se generarán barras de desplazamiento en caso de que la misma sobrepase los límites de la división.



Figura 8

Las columnas son de altura y anchor fijo, si los datos sobrepasan estos límites se generarán barras de desplazamiento en las celdas de las tablas. (Los datos en las tablas son meramente de prueba).



FIGURA 9

## FORMATO DE PETICIONES FETCH REACT-FUSEKI

Fuseki tiene la capacidad de funcionar como una API a la cual le podemos realizar peticiones POST, para ello utilizamos Fetch de javascript para el cual debemos definir las características de la petición entre ellos el método, el header, el cuerpo como lo muestra el código en la figura 10

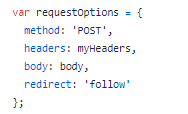


Figura 10

Como puede observarse en la figura 10 se aplica el método POST, en el header se define el tipo de contenido de la petición como se muestra en la figura 11, en este caso el tipo de contenido es “x-www-form-urlencoded”.



Figura 11

Se define el cuerpo de la petición que consiste en la consulta y su codificación utilizando encodeURIComponent(), el cual codifica un componente URI (Identificador Uniforme de Recursos) al reemplazar cada instancia de ciertos caracteres por una, dos, tres o cuatro secuencias de escape que representan la codificación UTF-8 del carácter. Dentro de la cadena existe una variable “filtroSparql” que se reemplazara por un valor de consulta en caso que se haya decidido filtrar la consulta (Desde la caja de consultas), vea la figura 12.

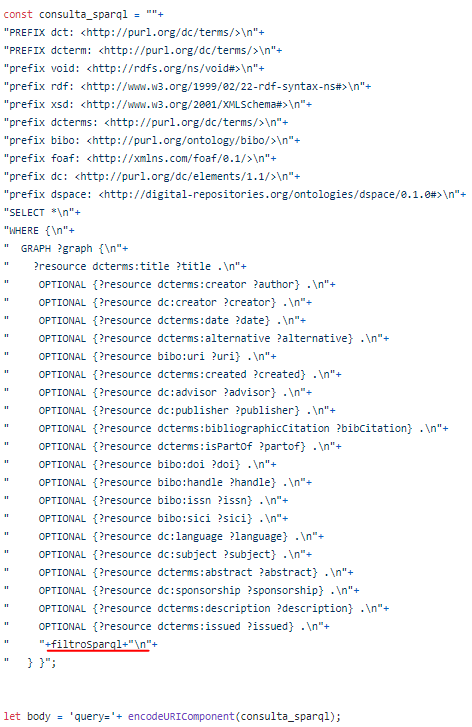


Figura 12

Finalmente, la petición es enviada al servidor utilizando la función fetch, los datos (que llegan en formato JSON) se almacenan en las variables de la interfaz

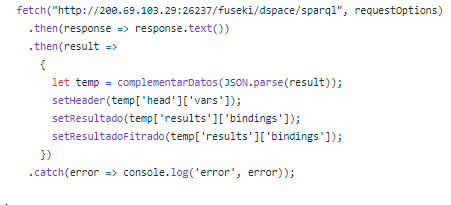


Figura 13

## CONSULTAS

Para realizar una consulta diríjase a la caja de consultas, si desea ver todos los resultados elija en el menú de opciones el valor “Ninguno” y ejecute la consulta utilizando el botón “Consultar Recursos” como en la Figura 14.

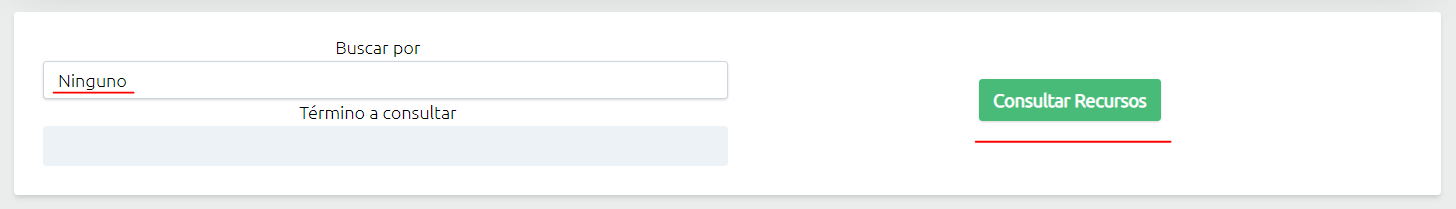


Figura 14

Si por el contrario desea filtrar por algún ítem del menú de selección, elija el ítem y escriba el valor a buscar (El valor debe ser claro y de acuerdo al formato de resultados de fuseki, sin no los conoce puede ejecutar una consulta sin filtro y verificarlo).

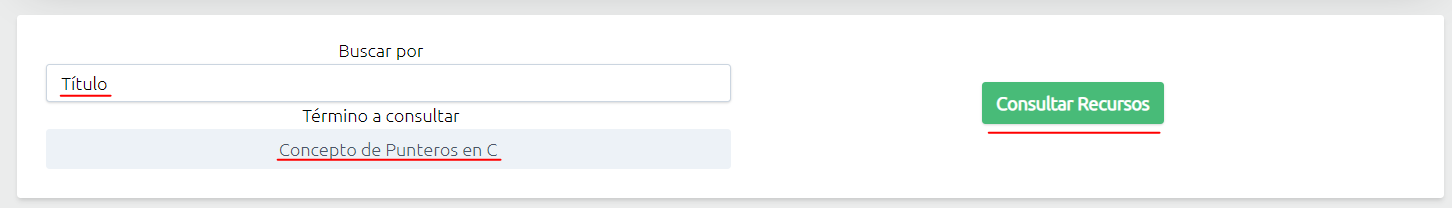


Figura 15

## TABLA DINAMICA DE CONSULTA

Una vez se consulten los datos utilizando la caja de consulta, los resultados pueden filtrarse nuevamente en la caja de tabla utilizando en recuadro en la parte superior derecha, solo debe escribirse el término a buscar. Cada vez que se escriba o borre un carácter del recuadro automáticamente se buscaran todas las celdas que incluyan el valor del mismo. Un ejemplo en la figura 14. No requiere que tenga en cuenta mayúsculas o minúsculas.

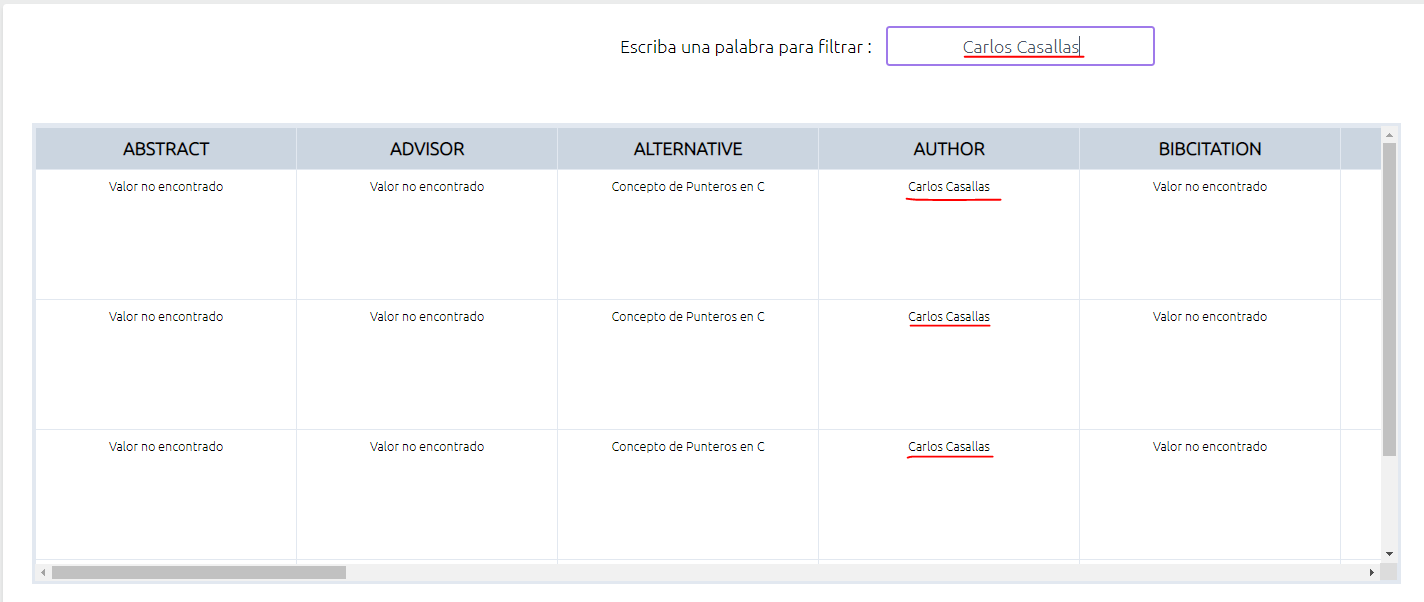


Figura 16

## DESPLIEGUE

* En puTTy ingrese las siguientes credenciales del servidor:
  + IP: 200.69.103.29
  + Puerto: 23380
  + Tipo de conexión SSH

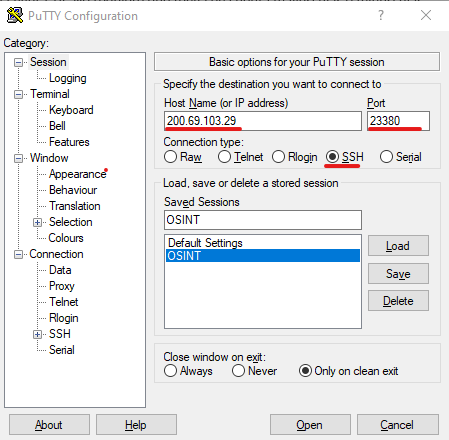


Figura 17

* Diríjase a Connection>SSH>Auth y busque el archive de conexión tipo \*.ppk entregado por el CECAD de la universidad distrital

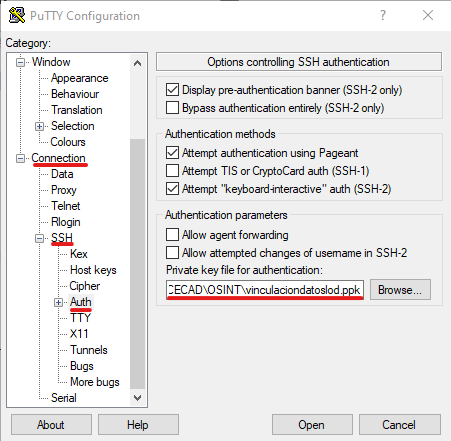


Figura 18

* Una vez iniciado verá la pantalla a continuación

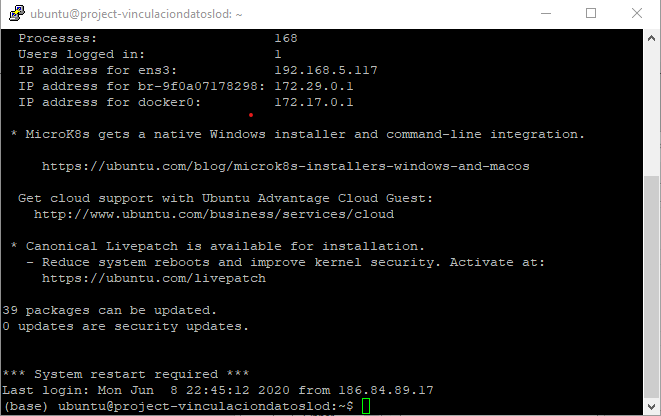


Figura 19

* El despliegue de esta aplicación funciona por medio de un proceso generado por un paquete de **node** llamado **serve** por lo cual debemos detener el proceso antes de continuar para ello utilizamos el comando **lsof -i :5000** el cual nos indica cual es el proceso corriendo en el puerto 5000 el cual es usado en el proceso de despliegue.

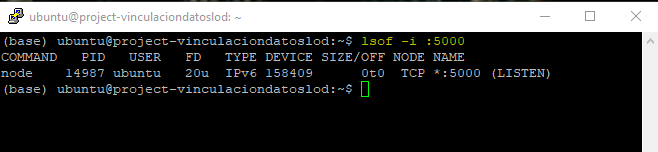


Figura 20

* Para detenerlo utilizamos el comando **kill** **PID** en el cual el parámetro PID es el número de identificación del proceso que encontramos en la columna **PID**, luego use nuevamente lsof y verifique que el proceso haya sido terminado.



Figura 21

* El proyecto fue desarrollo en React el cual trabaja sobre Node JS, para instalarlo se deben seguir los siguientes pasos
  + Instalar **NVM** o **Node version manager**, este además de administrar las versiones de Node en el sistema fascilita su instalación, en la terminal escriba el comando **wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash**, una vez termine ejecute el comando **nvm**, debe mostrarle las propiedades del comando, si por el contrario muestra que no fue encontrado ejecute **source ~/-bashrc**, y luego nuevamente **nvm**, en la Figura 22 muestra la ejecución del comando que en este caso muestra que ya está instalado

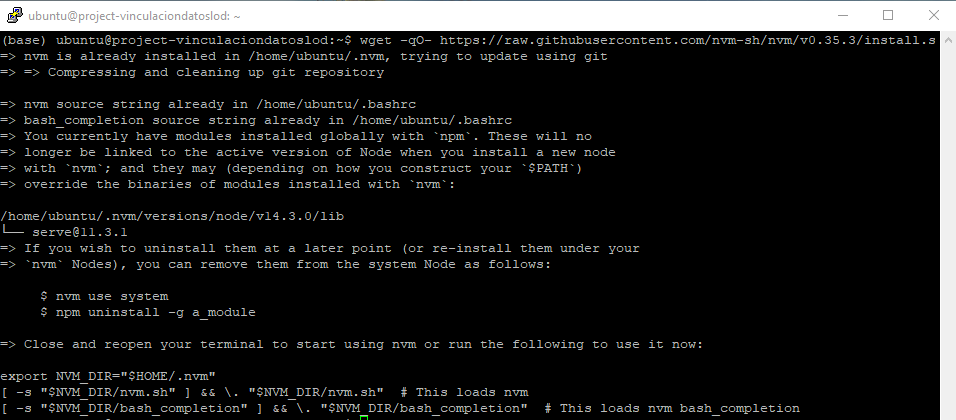


Figura 22

* + Ejecute el comando **nvm install node** este comando instala la última versión de node en el sistema

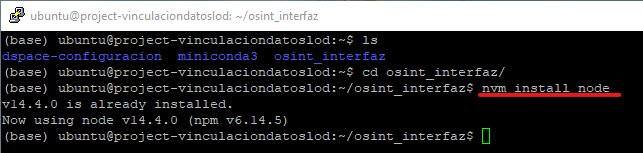


Figura 23

* + Una vez finalizado se deberá instalar YARN que se encargará de administrar los paquetes de node. Ejecute los comandos **curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add –** luego **echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list**

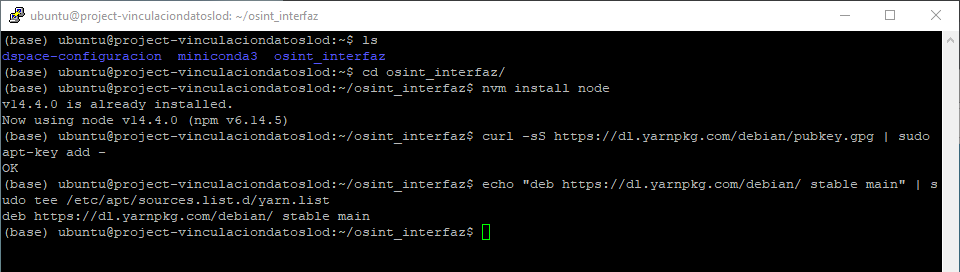


Figura 24

* + Por ultimo instale YARN ejecutando el comando **sudo apt update && sudo apt install yarn**

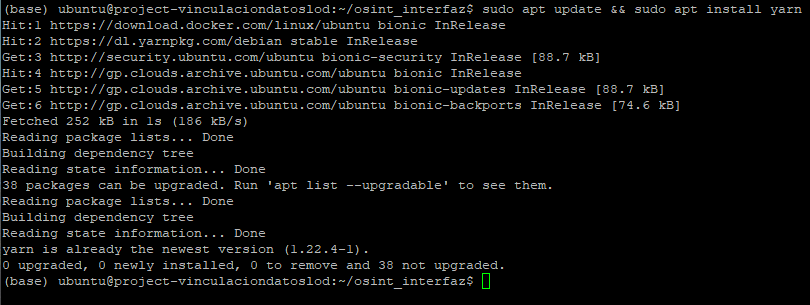


Figura 25

* + Clone el repositorio de la interfaz gráfica del proyecto con el comando **git clone** [**https://github.com/AlienX456/osint\_interfaz.git**](https://github.com/AlienX456/osint_interfaz.git)esto creara una carpeta en el directorio actual llamada **osint\_interfaz**, ingrese a ella con el comando **cd osint\_interfaz** y finalmente ejecute el comando **ls** para ver su contenido

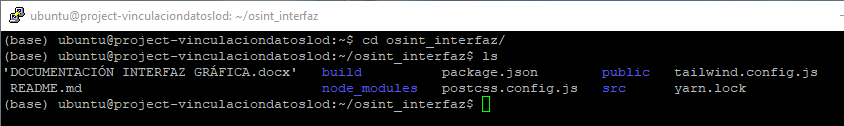


Figura 26

* + Ya en la carpeta ejecute **yarn** esto instalara automáticamente todas las dependencias del proyecto

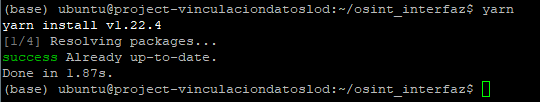


Figura 27

* + Luego ejecute **yarn build** para construir el proyecto y generar los archivos de producción, esto puede llevar algunos segundos en completarse dependiendo del poder de la máquina, al final se generará la carpeta **build**, compruébelo usando el comando **ls**

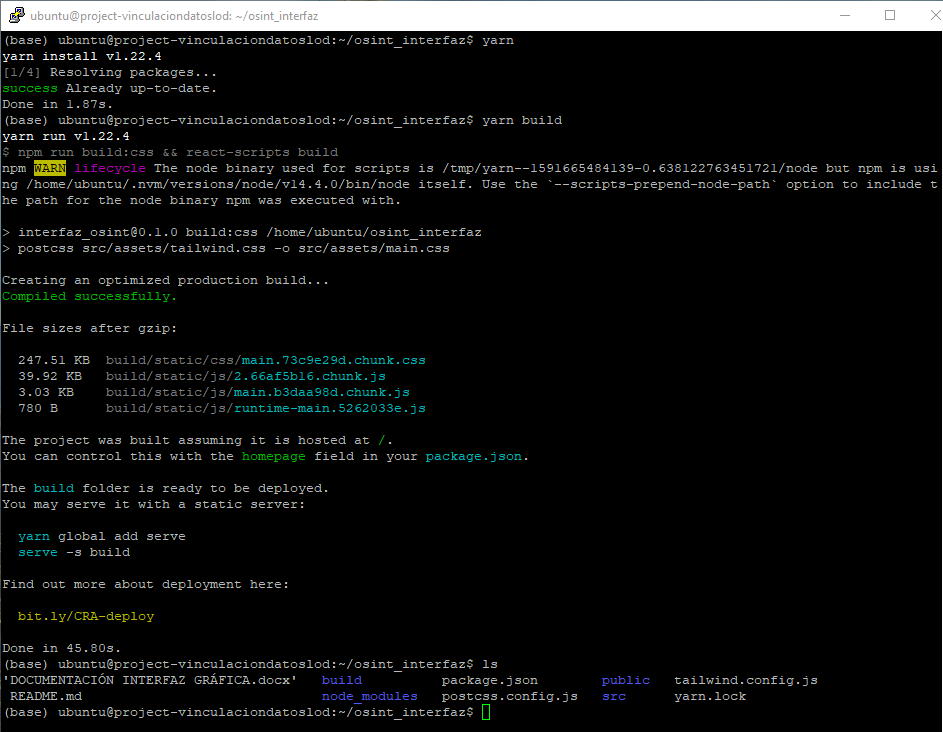


Figura 28

* + Para iniciar el proceso de despliegue instale **serve** utilizando **npm install -g serve**

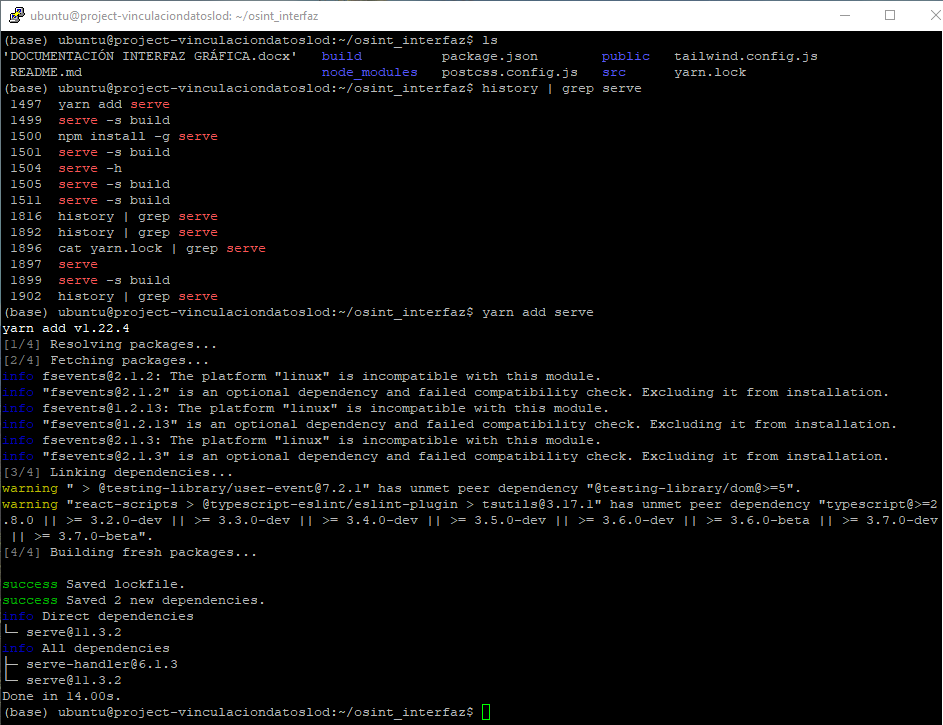


Figura 29

* + Una vez instalado ejecute el comando **serve –s build** esto desplegara la aplicación que en este caso debería poder visualizarse en la url [**http://200.69.103.29:26236/**](http://200.69.103.29:26236/)perteneciente al CECAD.

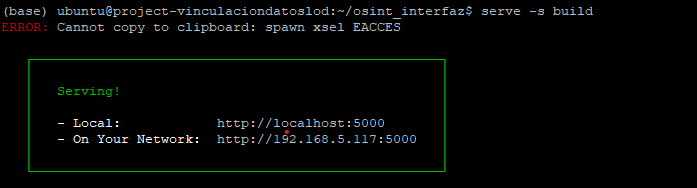


Figura 30

* + Si se cierra la terminal en este momento el proceso con el que se despliega la página terminara por lo cual debe oprimir **Ctrl + z** lo que llevara la aplicación a segundo plano, luego ejecute **bg** y **disown** para evitar que el proceso se cierre cuando deje la terminal

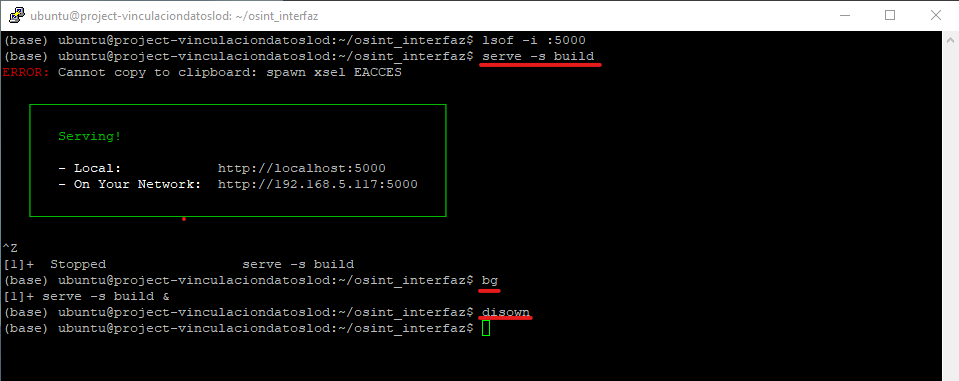


Figura 31

## REFERENCIAS

1. <https://es.reactjs.org/>
2. <https://tailwindcss.com/>
3. <https://yarnpkg.com/>