

ASleR

Amalio

28 de octubre de 2022

Índice

1. Idea	2
2. Tecnologías	2
2.1. Next Js	2
2.2. Next-PWA	2
2.3. Web-Push	2
2.4. MongoDB	2
2.5. Mongoose	2
2.6. MUI	3
3. Explicación de funciones	3
3.1. Calendario	3
3.2. Tareas	3
3.3. Notificaciones	3
3.4. Administradores	3
3.5. Generales	3
4. Resultado	4
4.1. /	4
4.2. /app	6
4.2.1. sidebar	7
4.3. /app/curso	8
4.3.1. /app/curso/avisos	8
4.3.2. /app/curso/examenes	8
4.3.3. /app/curso/tareas	8
4.3.4. /app/curso/calendario	9
4.3.5. /app/curso/encuestas	10
4.4. /app/tarea/id	11
4.5. /app/mes/dia	12
4.6. /app/admin	13
4.6.1. /app/admin/encuestas	15
5. Otras funciones	16
5.1. Cookies	16
5.1.1. ¿Qué cookies usamos?	16
5.2. Notificaciones	18
5.2.1. ¿Como funcionan?	18

5.2.2.	¿Son seguras?	19
5.3.	Base de Datos	19
5.3.1.	Seguridad	19
5.3.2.	Conexión	20

1. Intraducción a la idea

La idea tecnológica de este proyecto es construir una web, que a la vez sea instalable en el móvil (PWA) con funciones de notificación, base de datos y usando la librería de componentes de Material UI, inspirada en la filosofía Material Design desarrollado por Google.

Aparte de esto, la idea general es construir una web/app totalmente útil para difundir la información relevante de clase, así como fechas de exámenes y tareas.

2. Tecnologías a usar

2.1. NextJs(React)

Esta web/app estará básicamente construida en NextJs (12) de Vercel, que trabaja sobre React (18) creado y mantenido por Meta (Facebook). Este framework nos permitirá ahorrar tiempo de carga gracias a su renderizado desde el servidor (SSR) y a su arquitectura de una sola página (SPA) lo que dará dinamismo.

2.2. next-pwa

Next-Pwa es un módulo npm de javascript específico para NextJs que nos permitirá crear la PWA de forma muy rápida, sencilla y sin mucha configuración extra.

2.3. web-push

Este también es un módulo de javascript enfocado a manejar las notificaciones emergentes o 'push' que envía la web al dispositivo. Gracias a este módulo, conseguiremos eso usando la api interna del navegador, sin tener que usar ningún servicio externo.

2.4. MongoDB

También hemos comentado que usaremos una base de datos, en este caso es MongoDB aunque no se descartaría usar una que sea SQL como Postgres (Supabase).

2.5. Mongoose

Otro módulo npm que sirve para manejar MongoDB con sus drivers sin tener que hacerlo nosotros mismos y así simplificarlo todo.

2.6. Materila UI

Esta librería de interfaces de usuario para React, basada en la idea del Material Design dará una apariencia similar a las apps de google y por tanto más familiar y conocida.

3. Explicación de funciones

3.1. Calendario con alertas

Para el calendario, tendrá en las fechas que haya examen, un icono arriba a la derecha y si hay una tarea habrá otro icono distinto para las tareas. Aparte, al clicar en la fecha nos listará las tareas y exámenes que hubiera. Aparte, si para el examen quedan menos de dos días, esa fecha saldrá en el calendario con un fondo rojo y en el caso de que quede menos de un día para una tarea su fondo será naranja.

3.2. Tareas

Las tareas irán con una descripción corta y un título, aparte indicando la asignatura a la que pertenecen y la fecha límite.

Aparte, a la hora de visualizar las tareas, se podrán añadir comentarios, que irán listados con el nombre del comentarista y el comentario como tal.

Función Adicional También las tareas podrían tener un botón para indicar que se ha hecho y así ver el número de personas que las ha hecho, a modo de presión social para hacerla.

3.3. Notificaciones

Las notificaciones llegarán con unas vibraciones más continuadas si es fecha de examen y más cortas si es una tarea. Aparte, dentro de la notificación se indicará en el título si es tarea o exámen y en el cuerpo o mensaje de la notificación se indicará cuando es dicha tarea o dicho exámen.

3.4. Administradores

Para los administradores habrá una página especial donde podrán mandar notificaciones o añadir tareas y exámenes a la app. Aparte, cada vez que añadan una tarea o un exámen se mandará una notificación a todos los suscritos.

3.5. Funciones generales

borrado dinámico De las tareas y exámenes que ya han pasado, así no se llena la base de datos de archivos innecesarios.

reminder Añadir una notificación automática cuando quede menos de dos días para un examen o un día para una tarea.

autenticación Para la autenticación general usaremos la base de datos de MongoDB, donde los usuarios tendrán un nombre y una contraseña específica con ese nombre. También, podría considerar usar Auth0, una librería específica para ello.

Aparte, el nombre se guardará en el almacenamiento local para no tener que llamar todo el rato a la base de datos y así evitar consumir recursos.

Una vez 'registrados', habrá un renderizado condicional, es decir, la página de inicio de sesión o registro solo se mostrará mientras no haya almacenado en memoria un nombre, en cuanto inicien sesión no volverá a aparecerles el login.

auth admins Para los administradores, debo usar un método distinto de autenticación, ya que sus credenciales si deben ser 100% secretas, por eso para ellos puede que si use Auth0 o les deje crear sus propios usuarios y contraseñas, pero siempre será un registro controlado por mi. Una idea sería crear una colección distinta solo para ellos y crear una forma de añadirlos rápidamente con un simple form de dos inputs de tipo password. Entonces ellos (en mi presencia) irán añadiendo su usuario y contraseña de administrador enviándolo a la base de datos y una vez eso, ya estaría añadida dicha información, con una contraseña y usuario que solo ellos saben.

Pupurrí de secciones Aquí dejo un listado de ideas que me parecen interesantes e implementables:

- Sección de asignaturas con videos explicativos (yt).
- Sección de 'resúmenes' para cada examen, por si alguien quiere aportar su resumen a la clase.
- Botón de 'estoy en Discord' para avisar cuando alguien está en el discord de la clase.
- Zona 'Encuestas' para votar cosas y funciones.

6 de noviembre de 2022

4. Resultado

Una vez acabado el core del proyecto, paso a explicar las funciones que finalmente tiene:

4.1. Login

He implementado una pantalla con login muy básica, con la función de solo iniciar sesión ya que esta web app cuenta con usuarios 'predefinidos'.

Al hacer login, deben estar los dos inputs (nombre y contraseña) rellenos, aparte, la contraseña debe cuadrar con la guardada en la variable de entorno, si no, da error y no autentica.

Una vez rellenados los inputs, al pulsar el botón este envía una petición a la api interna en la cual se comprueba que la contraseña sea válida, una vez pasado ese test, crea una cookie y autentica al usuario.

Esta autenticación es clave en toda la app ya que en caso de no estar verificado, no se podrá acceder a ningún recurso de la web.

ASIR

La web/app de ASIR

Log In

Nombre

Contraseña

Entrar

Figura 1: /

imagen

4.2. APP

EN esta parte está el core de la web/app ya que es a pantalla por defecto, aquí se visualiza tanto el calendario, con el mes actual y el siguiente, como las tareas, exámenes y avisos.

calenderio En el calendario estarán los días, en los que si clicas te llevarán al detalle de ese día concreto. Los días que haya exámen estarán resaltados en un rojo anaranjado, los días que haya tareas en naranja, los días que hay tarea y exámen en rojo fuerte con un 'x2' en un lateral, y los días que hay aviso habrá un . negro a la derecha del número que indica el día.

exámenes Debajo de este calendario, está un listado con exámenes, así como tareas y otros avisos, todos listados en orden de fecha ascendente. Si clicas en cualquiera de ellos te llevarán al detalle, donde podrás agregar comentarios.

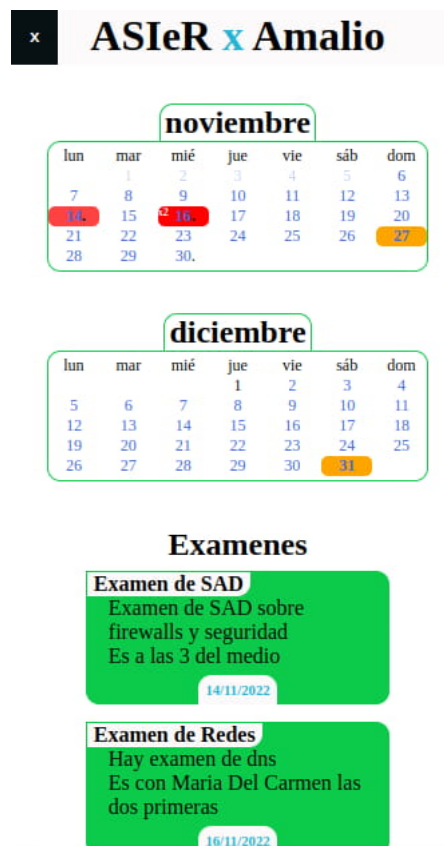


Figura 2: /app

imagen

4.2.1. sidebar

El sidebar estará presente en toda la interfaz gráfica de la app y variará según el rol de usuario (administrador o usuario). Este sidebar contiene un listado de todas las rutas disponibles para el usuario.

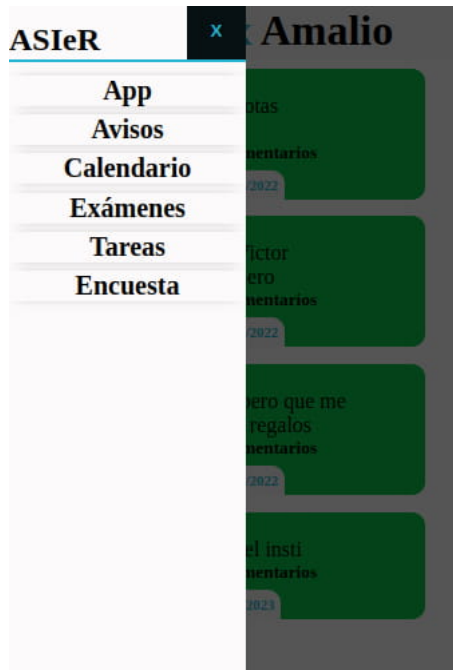


Figura 3: sidebar

imagen

4.3. Curso

Este es un subdirectorio del que cuelgan las siguientes secciones:

4.3.1. Avisos

En esta zona se encuentran listados todos los avisos en un orden ascendente por fecha (más próximos, antes). En cada aviso se muestra tanto su título, descripción y fecha como el número de comentarios que tiene. Pulsando en el aviso se va al detalle donde se puede comentar.

4.3.2. Exámenes

En esta zona se encuentran listados todos los exámenes en un orden ascendente por fecha (más próximos, antes). En cada examen listado se muestra tanto su título, descripción y fecha como el número de comentarios que tiene. Pulsando en el examen se va al detalle donde se puede comentar.

4.3.3. Tareas

En esta zona se encuentran listadas todas las tareas en un orden ascendente por fecha (más próximos, antes). En cada tarea se muestra tanto su título, descripción y fecha como el número de comentarios que tiene. Pulsando en una tarea se va al detalle de esta, donde se puede añadir comentarios.

imagen Al ser las tres secciones anteriores casi idénticas, una sola imagen basta para reflejar su aspecto:

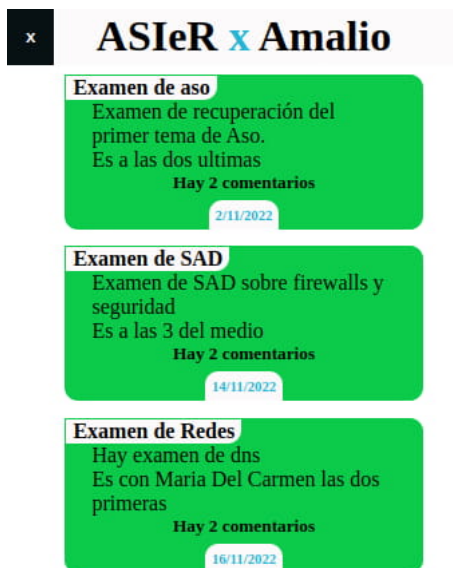


Figura 4: exámenes

4.3.4. Calendario

Aparte del calendario que ya viene en el /app, este es otro igual (mismas funcionalidades), solo que con los 6 meses siguientes que preceden al actual.

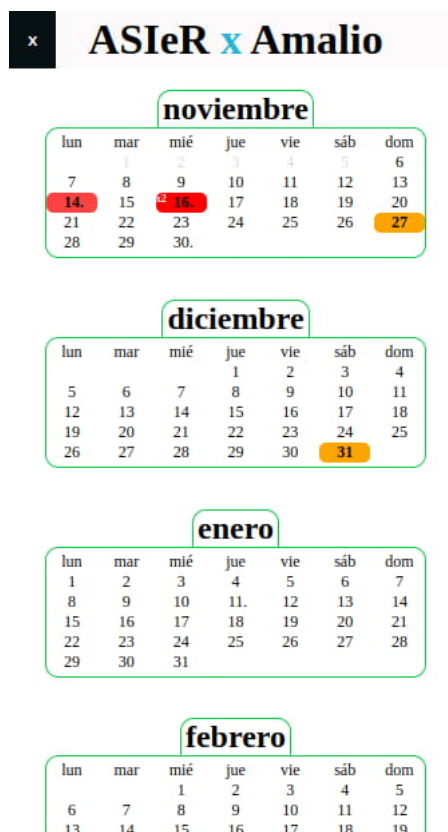


Figura 5: calendario

imagen

4.3.5. Encuestas

En esta zona están listadas las encuestas en orden descendente de adición (antes añadida, más abajo) donde, al pulsar en 'votar' en alguna de ellas abre el detalle y se podrá votar. En cada encuesta mostrada se observa el título y la descripción.

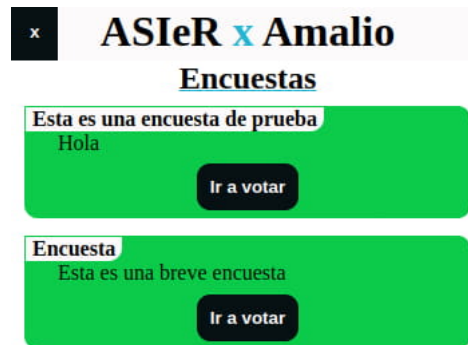


Figura 6: encuestas

imagen

4.4. Tarea:Detalle

Este es el detalle de alguna tarea, examen o aviso. En esta parte estará el título, la descripción, la fecha y también los comentarios así como la posibilidad de añadirlos.

x

ASiER x Amalio

Evaluación

2022-12-02

Ese día es la evaluación

Añadir comentario

tu comentario va aquí

Publicar

Amalio

Que ganas

Figura 7: detalle de tarea

imagen

4.5. Día:Detalle

El detalle del día es muy parecido al listado de tareas, avisos y exámenes. Muestran el título, descripción, número de comentarios y fecha de las tareas, exámenes o avisos que hay ese día. Al pulsar en alguna, lleva al detalle de la misma.

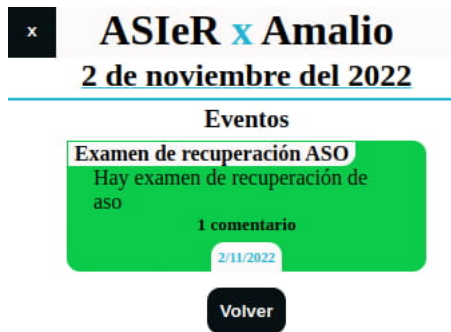


Figura 8: detalle del día

imagen

4.6. Admin

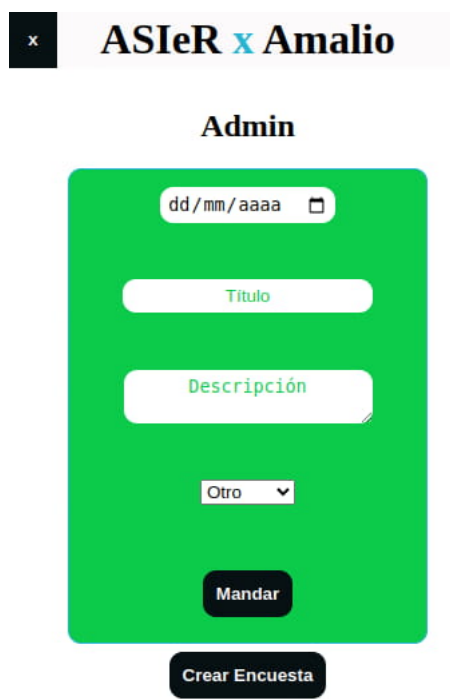
En esta sección se administra la adición de tareas, exámenes y avisos a la vez que las notificaciones, aparte, también se añaden encuestas. Esta es una ruta protegida puesto que solo los usuarios verificados pueden acceder a ella.

La verificación se hace en un enlace interno que no se muestra en el sidebar. Este enlace, en mi caso, es el `/app/control/create/one/administrator`. Una vez allí, se puede crear un nuevo usuario (si esa opción está cubierta dentro de las variables de entorno) o simplemente iniciar sesión. Una vez autenticado, aparecerá la sección admin dentro del sidebar:



Figura 9: admin sidebar

Dentro del admin, la interfaz en básica para realizar las acciones necesarias:



The image shows a web interface for 'ASIR x Amalio'. At the top, there is a header with a close button (x) and the text 'ASIR x Amalio'. Below the header, the word 'Admin' is centered. The main content area is a green rounded rectangle containing a form. The form has a date input field with the placeholder 'dd/mm/aaaa' and a calendar icon. Below the date field are two text input fields with the placeholders 'Título' and 'Descripción'. Under the description field is a dropdown menu with the option 'Otro' and a downward arrow. At the bottom of the green box is a black button labeled 'Mandar'. Below the green box is another black button labeled 'Crear Encuesta'.

Figura 10: admin

imagen

4.6.1. Admin Encuestas

En esta sección, los administradores pueden crear encuestas, con tantas opciones como deseen. Para seleccionar x opciones, ponen el número de opciones que necesitan en el input numérico, y, al dar a añadir opciones, se añadirán automáticamente al cuerpo de la encuesta. Una vez eso, solo hace falta rellenar la encuesta con el título, descripción y un título para cada opción. Si hiciera falta explicar las opciones, se deben explicar en la descripción.

El formulario se encuentra dentro de un modal con el título "ASIeR x Amalio" y un botón de cerrar "x".

En la parte superior, hay un input numérico con el valor "2" y un botón "Añadir Opciones".

Debajo de esto, el título "Encuestas" precede a un formulario principal con fondo verde.

Dentro del formulario verde:

- Un campo "Título" con el placeholder "título".
- Un campo "descripción" con el placeholder "Descripción breve".
- Una sección "Opciones" que contiene dos inputs con los placeholders "Opcion 1" y "Opcion 2".

En la parte inferior del formulario verde hay un botón "Crear".

Figura 11: añadir encuesta

imagen

5. Otras funciones

Ahora paso a listar y más adelante a desarrollar (con algunos ejemplos de código) ciertas funciones implementadas para aclararlas o simplemente resaltarlas.

- Cookies
- Notificaciones
- Conexión con la base de datos

5.1. Cookies

El uso de cookies en cualquier web debe ser notificado y es el usuario el que debe aceptar o denegar, al menos configurar.

Pero, si es así, ¿por qué no hay una ventanita o botón que te permita aceptar o denegar las cookies que nuestro sitio usa?

La respuesta es muy sencilla, ya que esta ley solo aplica a cookies 'de terceros' o cookies no necesarias.

Las cookies que usa este sitio son necesarias para el buen funcionamiento implícito así como para identificar al usuario y sus privilegios.

5.1.1. Nuestras Cookies

El número de cookies varía en función del navegador y rol del usuario, pero todas pertenecen a estas causas:

Service Workers Los Service Workers o sw son la clave de esta web/app ya que permiten a esta ser web ser una PWA. Los sw se instalan por defecto en el navegador para garantizar el correcto funcionamiento tanto de las notificaciones como de que si se llega a instalar la pwa, funcione como una app nativa. Para los sw se guardan entre 2 y 4 cookies referentes a los siguientes archivos:

1. SW principal: que es el que otorga la PWA
2. Sw secundarios: que pueden complementar al principal
3. Sw notificaciones: en el cual se registran las notificaciones entrantes y salientes
4. Web Manifest: que guarda el estado visual de la PWA: color de fondo, de la barra de navegación, etc...

Cookies de sesión Al identificar el usuario, se guardan cookies de sesión, tanto para el usuario como para los administradores.

Usuario Aquí dejo el código donde la llamada a la api hecha por el login, después de comprobar que los parámetros paados son verdaros devuelve la cookie de nombre:

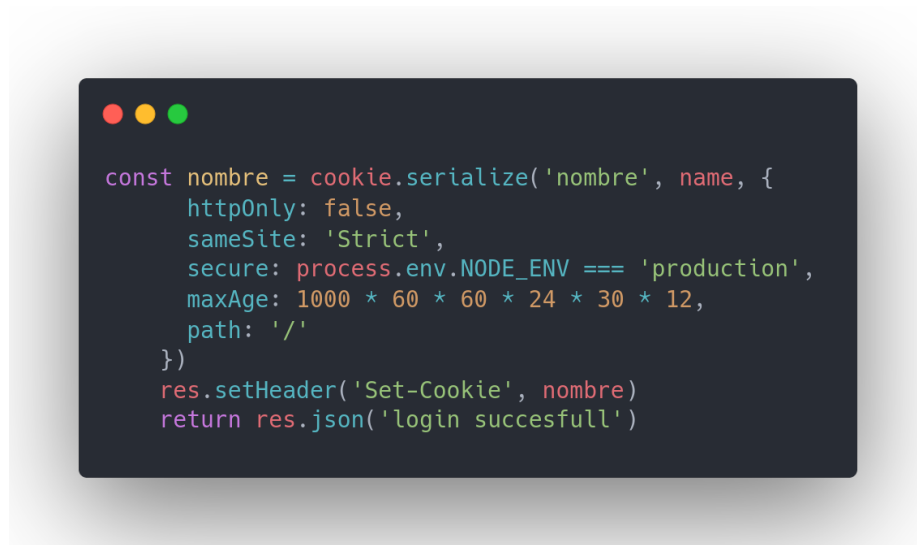


Figura 12: cookie usuario

Esta cookie se usa para persistir el nombre de usuario a lo largo de la aplicación, ya sea mostrandolo en la barra de navegación como al comentar. Aparte, caduca a los 6 meses, el máximo permitido, lo cual cubre el tiempo de clases.

Administrador LA cookie del administrador tiene la misma utilidad que la del usuario, sirve para persistir el estado de administrador a lo largo de la aplicación, para mostrarle el panel de admin en el sidebar y permitirle acceder a él:



Figura 13: cookie admin

Esta, al igual que la otra, se añade una vez comprobado que el usuario existe en la base de datos y también persiste durante el máximo, 6 meses. Aparte, usa una colección en la base de datos distinta a la que usa la autenticación del usuario.

5.2. Notificaciones

Ya hemos hablado de ellas antes, pero:¿Como funcionan?¿Son seguras?

5.2.1. ¿Como funcionan?

Cuando pulsas el botón de recibir notificaciones en el inicio de /app, el navegador hace una comprobación de que el sw esté activado y que además soporte las notificaciones (por eso el sw específico para ello). Una vez esto, cuando le damos a permitir, el navegador envía una respuesta en formato de objeto, con tres 'claves':

- endpoint
- p256dh
- auth

Todo este objeto es único para cada dispositivo y navegador, puedes suscribirte en el mismo móvil desde varios navegadores y estos datos serán totalmente distintos.

Una vez estás suscrito, el sw permanece 'a la escucha' de notificaciones, una vez se envía una petición desde el servidor, y las claves recibidas concuerda con las que el envió, muestra la notificación con los atributos (título,cuerpo,icono) que se le mandan.

5.2.2. ¿Son seguras?

EN la explicación de las notificaciones he omitido una cosa, y es como identifica quién manda las notificaciones. Bien, para esto se usa criptografía asimétrica, con una clave pública y una clave privada secreta, aparte, por seguridad, también es necesario un email.

Una vez tienes estas claves y el email, todos los 'endpoints' que nos devuelve cada navegador cuando alguien se suscribe irán cifrados con nuestras claves y correo.

Si alguien consiguiese de alguna manera algún endpoint, tendría papel mojado, ya que sin las claves, ambas, y el email, no sirven de nada.

5.3. Conexión con la BD

La base de datos que usa este proyecto es MongoDB, una bd no sql, distribuida y paso a explicar algunos aspectos a tener en cuenta.

5.3.1. Seguridad

La conexión a la base de datos gira entorno a una función, llamada mongoConn. Esta función importa la variable de entorno donde se guarda la url segura y única que provee mongo, en la cual van las credenciales del usuario con acceso a la bd. Este usuario tiene permisos de CRUD sobre la bd, y es creado por el propietario de la base de datos.



Figura 14: mongoConn

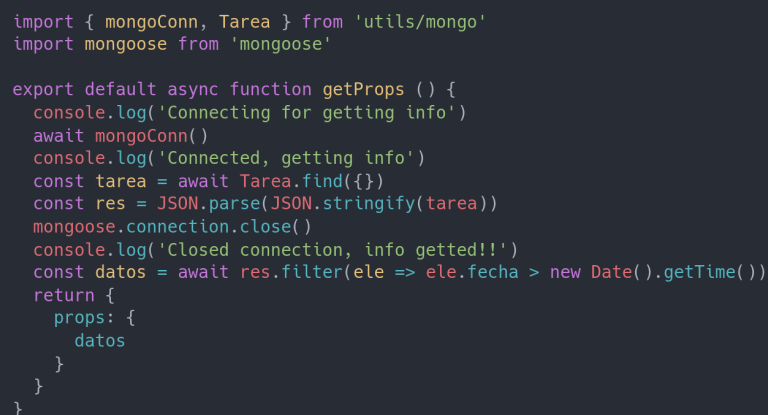
Aparte de esta url segura única para el usuario o usuarios con acceso a la bd, mongo nos permite decir desde que ips se permite la conexión a la base de datos, si esto los restringimos solo a la ip que tiene nuestra web, ninguna otra podrá acceder a la base de datos aún teniendo la url segura y la contraseña y nombre de el usuario propietario de dicha url.

Cuando hablo de usuarios de la base de datos, me refiero a administradores de dicha bd, no al rol de usuario o administrador de la web.

5.3.2. Conexión

La conexión a la base de datos es muy rápida y está implementada para no consumir recursos.

A lo largo de la app y también en los endpoints, se usa la misma dinámica para conectarse a la bd, y es así:



```
import { mongoConn, Tarea } from 'utils/mongo'
import mongoose from 'mongoose'

export default async function getProps () {
  console.log('Connecting for getting info')
  await mongoConn()
  console.log('Connected, getting info')
  const tarea = await Tarea.find({})
  const res = JSON.parse(JSON.stringify(tarea))
  mongoose.connection.close()
  console.log('Closed connection, info getted!!')
  const datos = await res.filter(ele => ele.fecha > new Date().getTime())
  return {
    props: {
      datos
    }
  }
}
```

Figura 15: conexión con mongo

¿Qué estoy viendo? Aquí lo más importante es que primero nos conectamos a la bd con `mongoConn`, una vez obtenida dicha conexión, conseguimos las Tareas que necesitamos y las guardamos en una constante. Una vez echo esto, desechamos la conexión cerrandola. Después seguimos manejando los datos que ya hemos guardado. Los `console.log` sirven para ver el estado de nuestra petición, si se ha cerrado correctamente la conexión y para ver el número de peticiones de datos que se hacen.