

16/03/2022



DESARROLLO EN ENTORNO SERVIDOR

Asier González y Mikel Goicoechea

Proyecto 4 Node-MongoDB

TABLA DE CONTENIDOS

1.- PROYECTO	3
1.1 - Node	3
1.2 - Mongo Atlas	6
1.3 - Heroku	8
2.- FUENTES	9

1.- PROYECTO

1.1 - Node



El proyecto cuenta como inicio una página donde podemos observar la lista de las reservas hechas. Además, podemos ordenar estas reservas con el selector en el que pone -- Ordenar por --.

```
getAll(request, response) {
  let filtro = "";
  var orden = request.params.orden;
  switch (orden) {
    case "pre":
      filtro = {$sort: {_id: 1}}
      break;
    case "usu":
      filtro = {$sort: {usuario: 1}}
      break;
    case "fec":
      filtro = {$sort: {fecha: 1}}
      break;
    case "aul":
      filtro = {$sort: {aula: 1}}
      break;
  }

  Reservas.aggregate([filtro, {
    $project: {
      _id: 1,
      usuario: 1,
      aula: 1,
      fecha: 1,
      horaDesde: 1,
      horaHasta: 1
    }
  }]).then(res => {
    response.json(res)
  }).catch(err => {
    response.end("Ha ocurrido un error")
  })
}
```

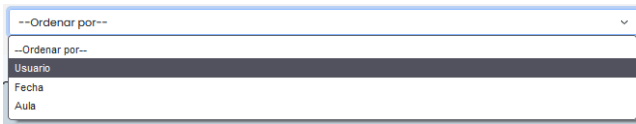
Para listar todas las reservas

```
app.get("/listarReservas/:orden", (request, response) => {
  cn.getAll(request, response);
})
```

Por defecto cuando abres la página se envía "pre" que ordena las reservas por id.

Dependiendo lo que se seleccione en el select el filtro se cambia y se mostrarán de diferente forma.

El menú desplegable que ve el usuario es el siguiente. Es donde tiene la opción de elegir cómo desea ordenar la tabla de reservas y su información.

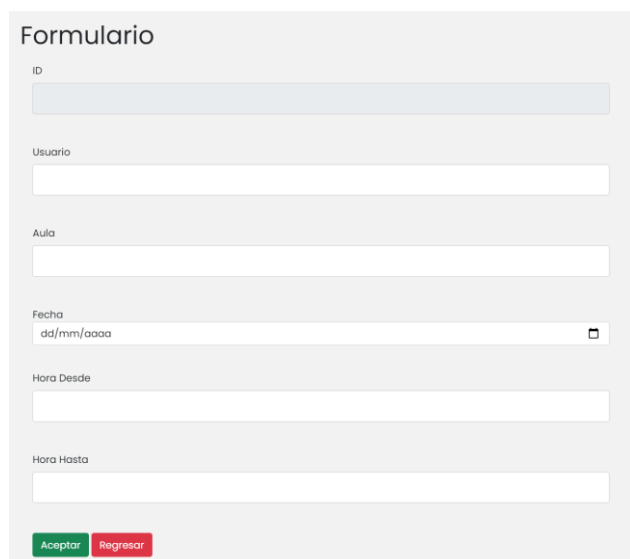
A screenshot of a web application's dropdown menu. The menu is open, showing three options: "--Ordenar por--", "Usuario", "Fecha", and "Aula". The "Usuario" option is currently selected and highlighted.

Cuando el usuario entra en la página principal la primera opción que ve es el botón de agregar una reserva en la parte superior.



Al *clickar* aparecerá un formulario con los campos a rellenar para realizar e insertar una nueva reserva. Los campos a rellenar son los diferentes datos de la reserva, el usuario deberá rellenar los campos **obligatorios** para poder realizar la inserción.

El usuario dispone de dos botones: **Aceptar** y **Regresar**. El botón de aceptar insertará la reserva con los datos introducidos por el usuario en la base de datos, si no ocurre ningún error. El botón de regresar regresa al usuario a la página principal.

A screenshot of a web form titled "Formulario". It contains several input fields: "ID" (a light blue field), "Usuario" (a white field), "Aula" (a white field), "Fecha" (a white field with a date picker icon and the format "dd/mm/aaaa"), "Hora Desde" (a white field), and "Hora Hasta" (a white field). At the bottom of the form are two buttons: a green "Aceptar" button and a red "Regresar" button.

En el caso de que haya un error porque las validaciones lo indiquen, se le hará saber al usuario mediante un cuadro emergente de **errores** informando sobre estos. La reserva no se podrá realizar hasta que no se presente ningún error.

- Debe ingresar un usuario
- Debe ingresar un aula
- Debe ingresar un fecha
- Debe ingresar una hora de inicio
- Debe ingresar una hora final de reserva

Una vez se ha realizado la reserva con éxito se le devolverá al usuario a la página principal con la tabla de reservas **actualizada** con el nuevo registro.

En la tabla de reservas el usuario tiene las opciones de **Editar** y **Eliminar** las reservas.



La opción de eliminar un registro tiene una **confirmación por parte del usuario** antes de ser realizada. Esta confirmación es una alerta en la que se le pregunta la confirmación de dicha acción al usuario. En el caso de que el usuario confirme la acción y se haya podido realizar la eliminación de reserva saltará un mensaje que confirma que se eliminó correctamente. La tabla de reservas **se actualizará automáticamente** con el campo ya eliminado.



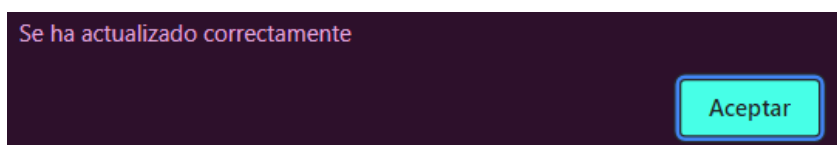
La opción de editar permite al usuario poder cambiar los campos que **no sean únicos** de cualquier reserva. Al hacer *click* sobre esta opción se abrirá un **formulario con los campos y valores de la reserva** seleccionada dándole la opción al usuario de poder editarlos.

El usuario podrá editar los campos habilitados y mediante el botón de **Aceptar** se actualizarán los datos.

En el caso de que el usuario desee regresar a la tabla de reservas podrá hacerlo mediante el botón de **Regresar**.

Una imagen de un formulario web con el título 'Formulario' en la parte superior. El formulario contiene varios campos de entrada: 'ID' con el valor '623059791c157b542b66e67', 'Usuario' con 'mgolcoeoca', 'Aula' con 'A09', 'Fecha' con '27/03/2022' y un icono de calendario, 'Hora Desde' con '12:00' y 'Hora Hasta' con '12:12'. En la parte inferior del formulario hay dos botones: 'Aceptar' en verde y 'Regresar' en rojo.

Una vez el usuario haya realizado todos los cambios que desee y haya los haya aceptado, se le notificará de que se han realizado cambios correctamente mediante una alerta. Se devolverá al usuario a la página principal con la tabla y datos actualizados.

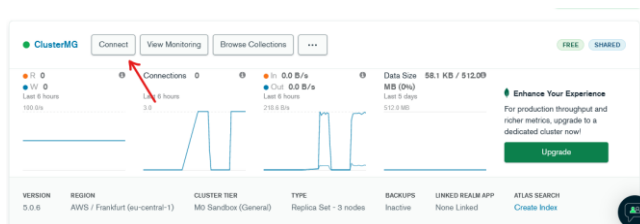


1.2 - Mongo Atlas

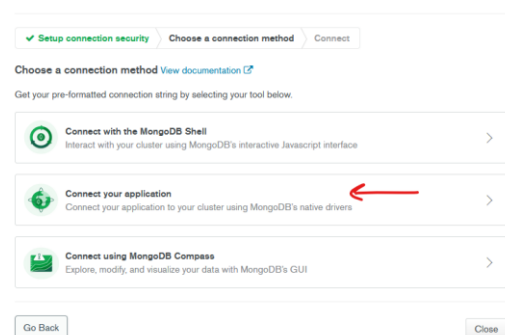
Para conectar Mongo Atlas hacemos el código con un `mongoose.connect`.

```
//Línea para conectarlo con Mongo Atlas
var cadenaConexion = "mongodb+srv://mgoicoeca:wfk7hX3TvfZ6wEF@clustermg.pz7eh.mongodb.net/Reservas?retryWrites=true&w=majority";
mongoose.connect(cadenaConexion, (err, res) => {
  if (err) {
    console.log('Ocurrió un error');
  } else {
    console.log('Se conectó correctamente');
  }
});
```

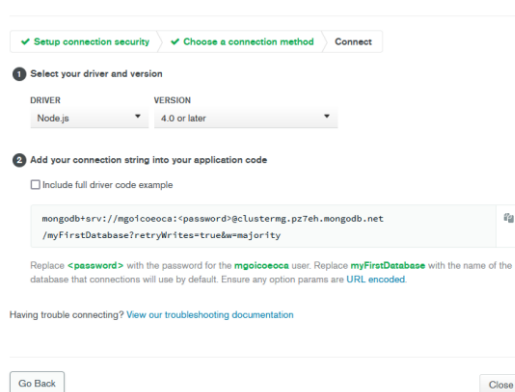
Para sacar esta **cadena de conexión** y permitir que se pueda acceder desde cualquier sitio hay que seguir estos pasos.



Connect to ClusterMG

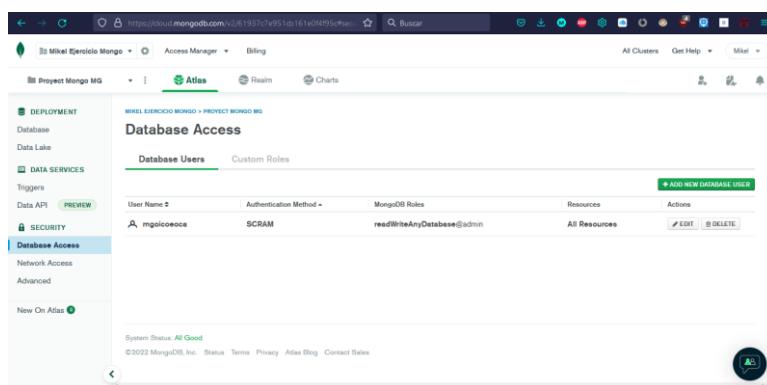


Connect to ClusterMG

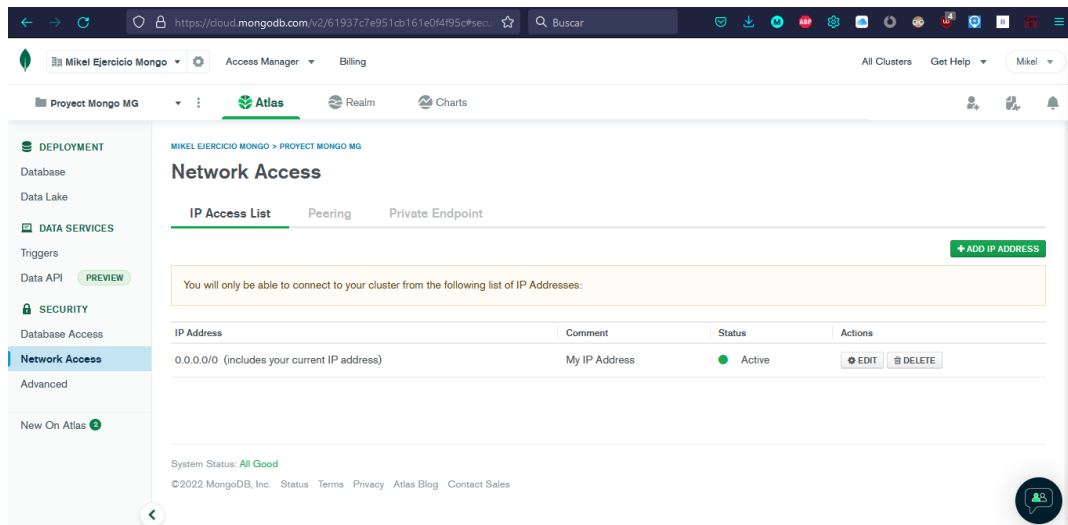


Le damos a la opción de Connect y elegimos conectar la aplicación. Nos dará el string en el que tendremos que añadir la nuestra contraseña y la base de datos.

Para la contraseña entramos en la página de Database Access que está a la izquierda de la página. Añadimos un usuario que pueda acceder a la base de datos y generamos una contraseña.



Además, tenemos que permitir que se acceda a la base de datos desde, en este caso, cualquier IP, para ello definimos que la IP accesible sea la 0.0.0.0/0.



Una vez terminado estos pasos podemos ya hemos conectado nuestra base de datos a la aplicación. Por último, nos toca subirlo a Heroku.

1.3 - Heroku

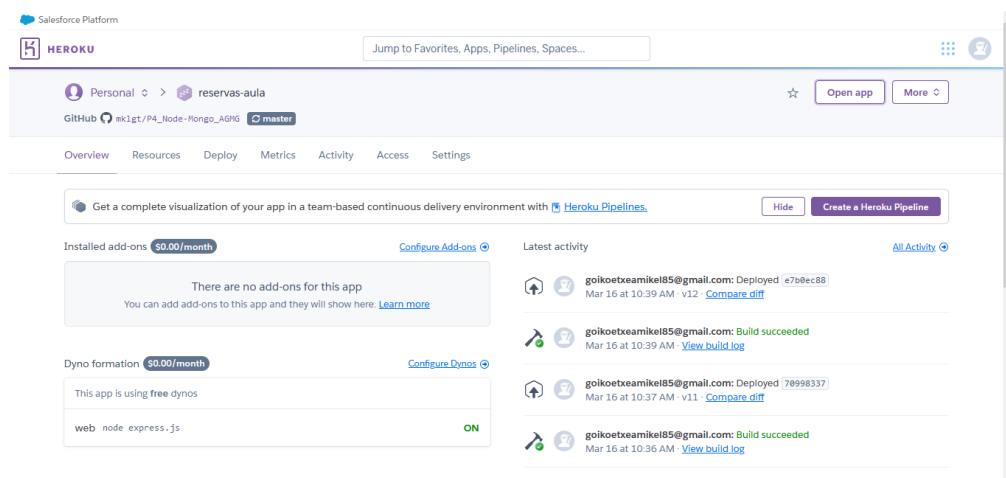
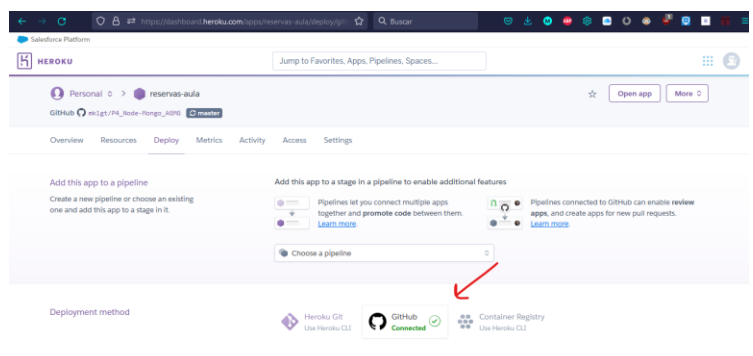
```
Procfile
Procfile
1 web: node express.js

let port = process.env.PORT;
if (port == null || port == "") {
  port = 8000;
}
app.listen(port);
```

Para que nuestra aplicación funcione en Heroku tenemos que añadir un archivo Procfile.

Y cambiar el puerto en el que se ejecuta ya que si no daría error al abrir la aplicación.

Ya hechos los cambios podemos conectar el proyecto con GitHub. Eso hará un Deploy.



Ya se puede acceder al proyecto: [Enlace](#)

2.- FUENTES

Para el proyecto hemos seguidos la guía detallada y bases del proyecto de nuestra profesora Karnele: [Enlace](#)

Para la opción de ordenar por hemos consultado la documentación oficial que ofrece MongoDB: [Enlace](#)

Al final para la opción de ordenar hemos consultado la documentación oficial que ofrece MongoDB al respecto: [Enlace](#)

Para la búsqueda por valores de campos que pensamos en utilizar estuvimos consultando la documentación que ofrece la página de SQL Server Guides: [Enlace](#)