

**Curso:**

# **Programador Web Inicial-Front End Developer**



Módulo 5:

# CMS y despliegue en Heroku

Unidad 2:

## CRUD (parte 2)



## Presentación

En esta unidad continuamos trabajando con el CRUD de novedades. En este caso sumamos las funcionalidades necesarias para agregar y modificar novedades desde nuestro administrador.



## Objetivos

Que los participantes logren...

- Implementar el código necesario para dar de alta novedades.
- Comprender el proceso de modificación de una novedad.



## Bloques temáticos

1. Alta admin novedades..
2. Modificar admin novedades..



# 1. Alta admin novedades

## Paso 1

En el archivo **models/novedades Model.js** agregaremos la función necesaria para dar de alta una novedad.

```
async function insertNovedad(obj) {  
  try {  
    var query = "insert into novedades set ? ";  
    var rows = await pool.query(query, [obj]);  
    return rows;  
  } catch (error) {  
    console.log(error);  
    throw error;  
  }  
}  
  
module.exports = { getNovedades, deleteNovedadById,  
  insertNovedad }
```

## Paso 2

Modificamos nuestro archivo **views/admin/novedades.hbs** creamos un link a una nueva ruta que utilizaremos para mostrar el formulario donde ingresaremos las novedades.

```
<div class="col-2 text-right">
  <a href="/admin/novedades/agregar" class="btn btn-success">
    <i class="fa fa-plus"></i>Nuevo</a>
</div>
```

## Paso 3

En el archivo **routes/admin/novedades.js** creamos el nuevo manejador de ruta que mostrará el formulario de alta.

```
router.get('/agregar', (req, res, next) => {
  res.render('admin/agregar', {
    layout: 'admin/layout'
  });
});
```



## Paso 4

Creamos el archivo **views/admin/agregar.hbs** con el código necesario para el formulario. Dirigimos el action a /admin/novedades/agregar.

```
<div class="container" style="margin:100px auto">
  <div class="row">
    <div class="col-6 offset-3">
      <h4>Agregar una nueva novedad</h4>
      <form action="/admin/novedades/agregar" method="post">
        <div class="form-group">
          <input type="text" class="form-control"
            placeholder="Titulo" name="titulo">
        </div>
        <div class="form-group">
          <textarea type="text" class="form-control"
            placeholder="Subtitulo"
            name="subtitulo"></textarea>
        </div>
        <div class="form-group">
          <textarea type="text" class="form-control"
            placeholder="Cuerpo" name="cuerpo">
          </textarea>
        </div>
        <button type="submit" class="btn btn-
          primary">Guardar</button>
      </form>
    </div>
  </div>
</div>
```



## Paso 5

De vuelta en el archivo **routes/admin/novedades.js**, agregaremos el controlador necesario para capturar los datos enviados por el formulario. Una vez recibidos, validamos que tengamos todos los datos necesarios y se los pasamos a la función del model para guardarlos en la base de datos.

En caso de no tener todos los datos necesarios o que se produzca un error en el grabado de la información volveremos a mostrar el mismo template pasándole una variable que indique que hubo un error y una descripción del mismo.

```
router.post('/agregar', async (req, res, next) => {
  try {
    if (req.body.titulo !== "" && req.body.subtitulo !== "" &&
        req.body.cuerpo !== "") {
      await novedadesModel.insertNovedad(req.body);
      res.redirect('/admin/novedades')
    } else {
      res.render('admin/agregar', {
        layout: 'admin/layout',
        error: true, message: 'Todos los campos son requeridos'
      })
    }
  } catch (error) {
    console.log(error)
    res.render('admin/agregar', {
      layout: 'admin/layout',
      error: true, message: 'No se cargo la novedad'
    });
  }
});
```

## Paso 6

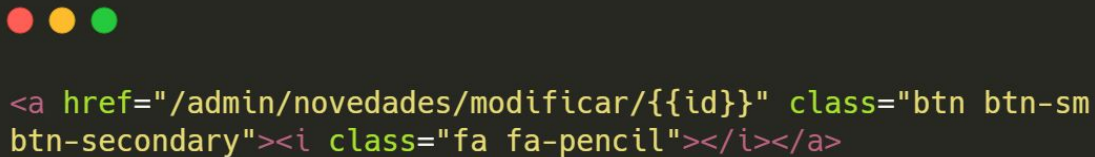
En el archivo **view/admin/agregar.hbs** agregamos el `if` para que imprima el error en caso de haber uno.

```
{{#if error}}  
  <h5>{{message}}</h5>  
{{/if}}
```

## 2. Modificar admin novedades

### Paso 1

Modificamos nuestro archivo **views/admin/novedades.hbs** e incluimos un link a la ruta de modificar.



```
<a href="/admin/novedades/modificar/{{id}}" class="btn btn-sm  
btn-secondary"><i class="fa fa-pencil"></i></a>
```

### Paso 2

En el archivo **models/novedades Model.js** agregaremos dos funciones nuevas. Una nos permitirá obtener una noticia única de la base de datos utilizando el id de la misma para seleccionarla. La otra será la encargada de modificar los campos de la novedad que seleccionemos por id y que reciba como parámetro.



```
async function getNovedadById(id) {
    var query = "select * from novedades where id = ? ";
    var rows = await pool.query(query, [id]);
    return rows[0];
}

async function modificarNovedadById(obj, id) {
    try {
        var query = "update novedades set ? where id=?";
        var rows = await pool.query(query, [obj, id]);
        return rows;
    } catch (error) {
        throw error;
    }
}

module.exports = { getNovedades, insertNovedad,
deleteNovedadById, getNovedadById, modificarNovedadById }
```

## Paso 3

En el archivo **routes/admin/novedades.js** creamos el controlador de ruta necesario para imprimir el formulario de modificación. Esta ruta tiene la particularidad, al igual que la de eliminar, de recibir como parámetro el id de la novedad. Este id se utilizara para llamar a la función previamente creada y pasar la novedad seleccionada al template.



```
router.get('/modificar/:id', async (req, res, next) => {  
  
  let id = req.params.id;  
  let novedad = await novedadesModel.getNovedadById(id);  
  res.render('admin/modificar', {  
    layout: 'admin/layout',  
    novedad  
  });  
});
```

## Paso 4

Creemos la vista **views/admin/modificar.hbs** donde armaremos el formulario para modificar. Este es muy similar al de agregar, con la salvedad de que los valores de los campos vienen predefinidos por los campos de la novedad que trajimos de la base de datos.

Además agregamos un campo oculto que utilizaremos en el envío del formulario para saber qué novedad debemos modificar.



```
<div class="contanier" style="margin:50px auto">
  <div class="row">
    <div class="col-6 offset-3">
      <h4>Modificar la novedad</h4>
      <form method="post" action="/admin/novedades/modificar">
        <input type="hidden" value="{{novedad.id}}" name="id">
        <div class="form-group">
          <input type="text" placeholder="Titulo"
            value="{{novedad.titulo}}" class="form-control"
            name="titulo">
        </div>
        <div class="form-group">
          <textarea placeholder="Subtitulo" class="form-control"
            name="subtitulo">{{novedad.subtitulo}}</textarea>
        </div>
        <div class="form-group">
          <textarea placeholder="Cuerpo" class="form-control"
            name="cuerpo">{{novedad.cuerpo}}</textarea>
        </div>
        <button type="submit" class="btn btn-primary">Modificar</button>
      </form>
    </div>
  </div>
</div>
```

## Paso 5

En el archivo **routes/admin/novedades.js** creamos el controlador encargado de recibir los datos del formulario y pasarlos a la función de model para efectuar la modificación de la novedad en la base de datos.

En caso de éxito redirigimos al usuario al listado de novedades, de lo contrario enviamos una variable de error y el mensaje describiendo el mismo.



```
router.post('/modificar', async (req, res, next) => {
  try {
    let obj = {
      titulo: req.body.titulo,
      subtítulo: req.body.subtítulo,
      cuerpo: req.body.cuerpo
    }

    await novedadesModel.modificarNovedadById(obj, req.body.id);
    res.redirect('/admin/novedades');
  }
  catch (error) {
    console.log(error)
    res.render('admin/modificar', {
      layout: 'admin/layout',
      error: true, message: 'No se modifico la novedad'
    })
  }
})
```

## Paso 6

En el archivo **views/admin/modificar.hbs** agregamos el `if` para que imprima el error en caso de haber uno.

```
{{#if error}}
  <h5>{{message}}</h5>
{{/if}}
```



## Bibliografía utilizada y sugerida

### Artículos de revista en formato electrónico:

**Handlebars.** Disponible desde la URL: <https://handlebarsjs.com/>

**npmjs.** Disponible desde la URL: <https://www.npmjs.com/>