

En la clase anterior comenzamos a instalar el servidor **node** y los módulos que utilizaremos en nuestra aplicación web para hacer una **agenda**. Hemos creado la carpeta **src** y dentro de ella el archivo **app.js** que contiene las declaraciones e instancias de nuestro **framework**. En esta clase vamos a continuar preparando el entorno de desarrollo y vamos a **crear en Xampp la base de datos** y la tabla con la que vamos a trabajar. Luego, nos conectaremos.

- 1) Ejecutamos **Xampp** Control Panel, arrancamos **MySQL** y **Apache**, vamos al navegador web y escribimos **localhost**.
- 2) Entramos en **phpMyAdmin** y vamos a crear una **nueva base de datos** con el nombre: **crudnodejs**.
- 3) Dentro de esa base de datos, vamos a crear la tabla **agenda** de 4 columnas, que tendrá la siguiente estructura de campos:

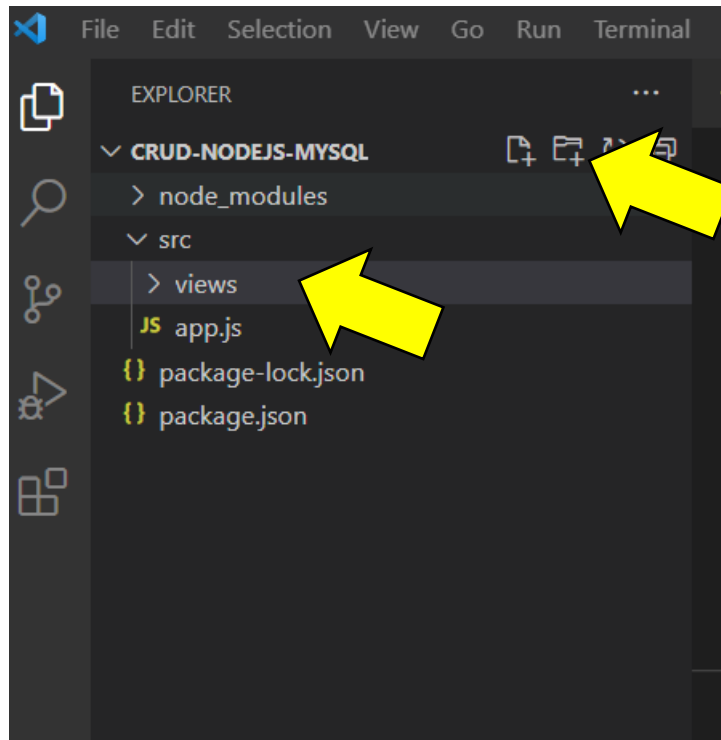
Nombre de Campo	Tipo	Longitud	A..I	Nulo
id_agenda	INT	14	<input checked="" type="checkbox"/>	
ag_nombre	VARCHAR	40		
ag_domicilio	VARCHAR	50		<input checked="" type="checkbox"/>
ag_telefono	VARCHAR	30		<input checked="" type="checkbox"/>

En el campo **id\_agenda**, recordá marcar el **AUTOINCREMENTAL (A..I)** para indicar que ese campo de la tabla se incrementará automáticamente, generando un ID único para cada registro. Vamos a indicar en los campos **ag\_domicilio** y **ag\_telefono** que pueden aceptar **valores nulos**.

Tiene que quedar así:

Servidor: 127.0.0.1 » Base de datos: crudnodejs » Tabla: agenda										
Examinar	Estructura	SQL	Buscar	Insertar	Exportar	Importar	Privilegios	Operaciones	Dispa	
Estructura de tabla		Vista de relaciones								
#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción	
<input type="checkbox"/> 1	id_agenda	int(14)			No	Ninguna		AUTO_INCREMENT	Cambiar	Eliminar
<input type="checkbox"/> 2	ag_nombre	varchar(40)	utf8mb4_general_ci		No	Ninguna			Cambiar	Eliminar
<input type="checkbox"/> 3	ag_domicilio	varchar(50)	utf8mb4_general_ci		Sí	NULL			Cambiar	Eliminar
<input type="checkbox"/> 4	ag_telefono	varchar(30)	utf8mb4_general_ci		Sí	NULL			Cambiar	Eliminar

- 4) Entramos en **Visual Studio Code**.
- 5) Iremos a la opción **File** y luego elegimos **Open Folder**. Buscamos y seleccionamos la carpeta **crud-nodejs-mysql** que creamos durante la clase anterior.
- 6) **Seleccionamos la carpeta src** y **dentro de ella** vamos a crear la carpeta **views** que nos servirá para almacenar todas las plantillas que correspondan a la vista.

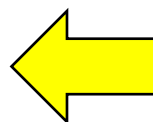


- 7) Abrimos la carpeta **src** y vamos a editar el archivo **app.js** haciendo doble click en él. Vamos a setear algunos aspectos del framework Express para nuestra aplicación. Debajo de las definiciones Const, vamos a agregar un comentario que diga **//Settings** y debajo de él lo siguiente para indicarle a Express que el **motor de plantillas** será **ejs**. Debe quedar así:

```
const express = require("express");
//requerimos el módulo
const app = express();
//inicializamos express

//Settings
app.set('view engine', 'ejs');

//routes
```




- 8) Vamos a agregar otras constantes. Una será **path** y la usaremos para indicar la ruta a la carpeta **views**. Otra será **morgan** y la usaremos para instanciar el módulo del mismo nombre. Además, agregaremos un comando **set** para indicar dónde debe buscar la carpeta **views**. Vamos a agregar un comentario de middleware para indicar el lugar donde vamos a instanciar a **morgan**. Todo debe quedar como muestra la siguiente imagen:

```
const express = require("express");
//requerimos el módulo
const app = express();
//inicializamos express
const path = require('path');
//directorio de la app
const morgan = require('morgan');
//inicializamos morgan

//Settings
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

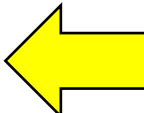
//Middleware
app.use(morgan('dev'));
```



- 9) Vamos a configurar nuestro servidor para indicarle que vamos a trabajar con MySQL. Para ello vamos a definir dos **const**: una para el módulo **mysql** y otra para **mysql-myconnection**. Debe quedar así:

```
const express = require("express");
//requerimos el módulo
const app = express();
//inicializamos express
const path = require('path');
//directorio de la app
const morgan = require('morgan');
//inicializamos morgan

//MySQL
const mysql = require('mysql');
const myconnection = require('express-myconnection');
```

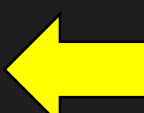


- 10) Vamos a agregar en los middlewares la conexión a MySQL. Debe quedar así:

```
//Settings
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

//Middleware
app.use(morgan('dev'));
app.use(myconnection(mysql, {
  host: 'localhost',
  user: 'root',
  password: '',
  port: 3306,
  database: 'crudnodejs'
}, 'single'));

//routes
//inicializamos el servidor
```



Si todo se ha agregado correctamente, tu archivo `app.js` completo debería verse así (controlá que todo esté correctamente escrito):

```
const express = require('express');
//requerimos el módulo
const app = express();
//inicializamos express
const path = require('path');
//directorio de la app
const morgan = require('morgan');
//inicializamos morgan

//MySQL
const mysql = require('mysql');
const myconnection = require('express-myconnection');

//Settings
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

//Middleware
app.use(morgan('dev'));
app.use(myconnection(mysql, {
  host: 'localhost',
  user: 'root',
  password: '',
  port: 3306,
  database: 'crudnodejs'
}, 'single'));

//routes
//inicializamos el servidor
//El servidor escuchará en el puerto 3000.
//Mostramos un mensaje indicando que está activo.
app.listen(3000, () => {
  console.log('Server en puerto 3000') })
;
```

11) En la terminal escribimos el comando:

**node src/app.js**

12) Si todo está correcto, debemos ver el mensaje: **“Server en puerto 3000”**

Es importante que lleguemos todos juntos a este punto para poder avanzar. Revisá que hayas completado correctamente todos los pasos de estas dos clases.