

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Software Avanzado

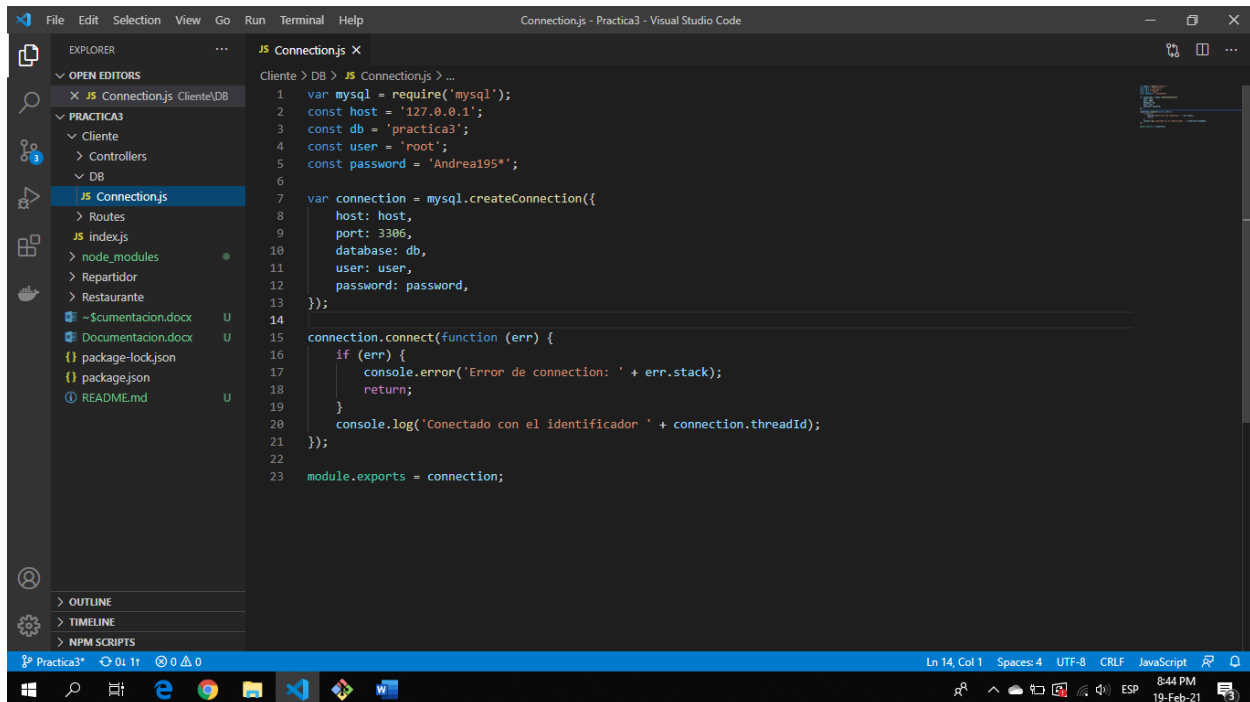
Documentación – Práctica 3

Andrea María López Flores

201404134

Guatemala, 19 de febrero de 2021

Conexión con la base de datos

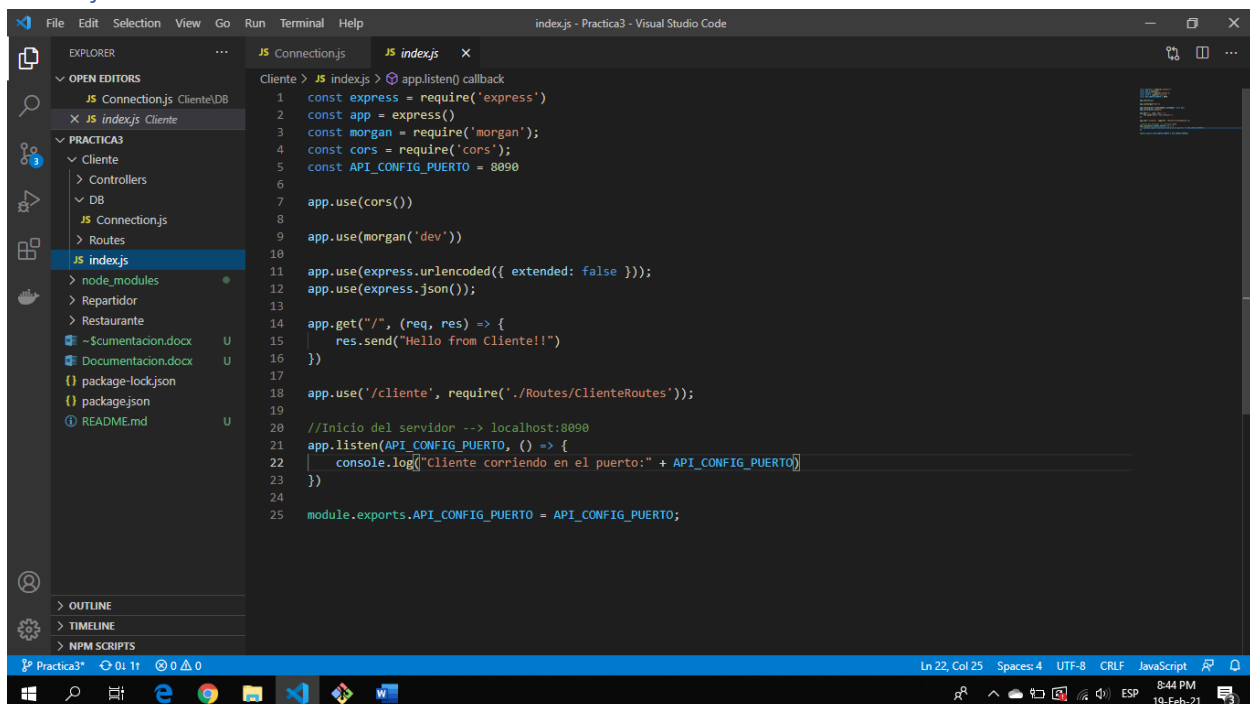


```
1 var mysql = require('mysql');
2 const host = '127.0.0.1';
3 const db = 'practica3';
4 const user = 'root';
5 const password = 'Andrea195*';
6
7 var connection = mysql.createConnection({
8   host: host,
9   port: 3306,
10  database: db,
11  user: user,
12  password: password,
13 });
14
15 connection.connect(function (err) {
16   if (err) {
17     console.error('Error de conexión: ' + err.stack);
18     return;
19   }
20   console.log('Conectado con el identificador ' + connection.threadId);
21 });
22
23 module.exports = connection;
```

Existe un archivo de conexión para el cliente, restaurante y repartidor. Todos contienen la misma información ya que todos se conectan a la misma base de datos.

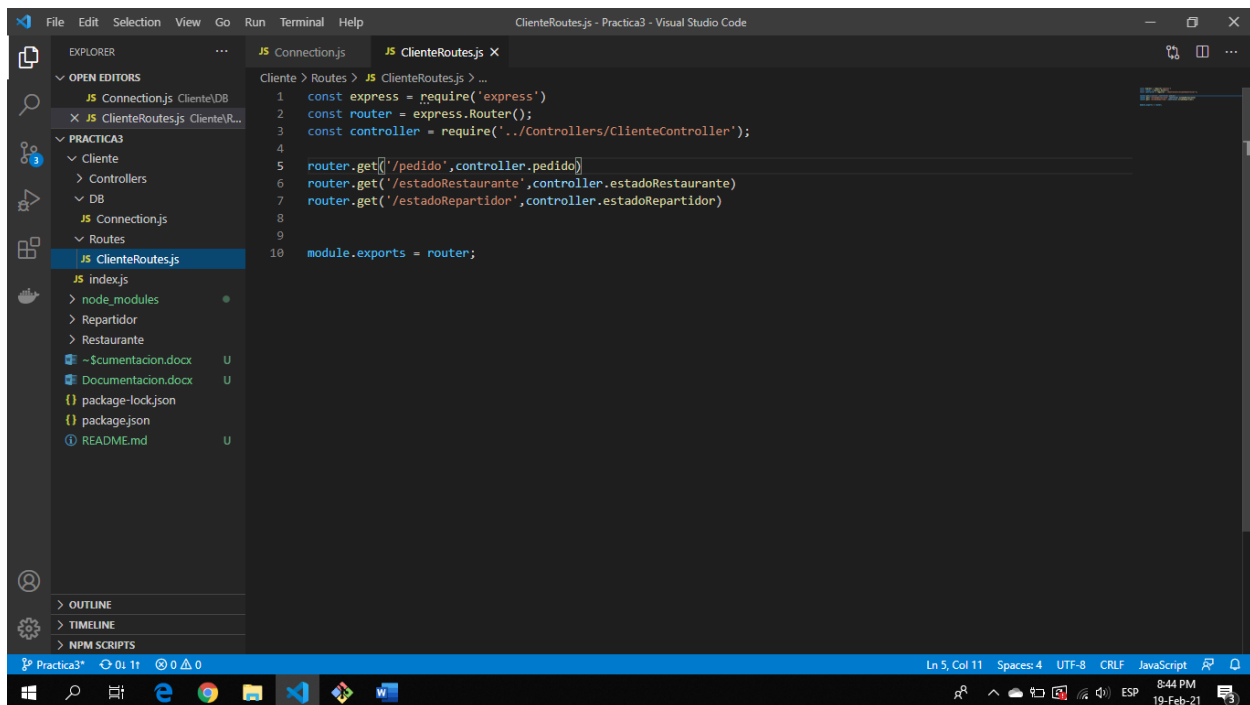
Cliente

Index.js



```
1 const express = require('express')
2 const app = express()
3 const morgan = require('morgan');
4 const cors = require('cors');
5 const API_CONFIG_PUERTO = 8090
6
7 app.use(cors())
8
9 app.use(morgan('dev'))
10
11 app.use(express.urlencoded({ extended: false }));
12 app.use(express.json());
13
14 app.get('/', (req, res) => {
15   res.send("Hello from Cliente!!")
16 })
17
18 app.use('/cliente', require('./Routes/ClienteRoutes'));
19
20 //Inicio del servidor --> localhost:8090
21 app.listen(API_CONFIG_PUERTO, () => {
22   console.log(`Cliente corriendo en el puerto: " + API_CONFIG_PUERTO`)
23 })
24
25 module.exports.API_CONFIG_PUERTO = API_CONFIG_PUERTO;
```

Rutas

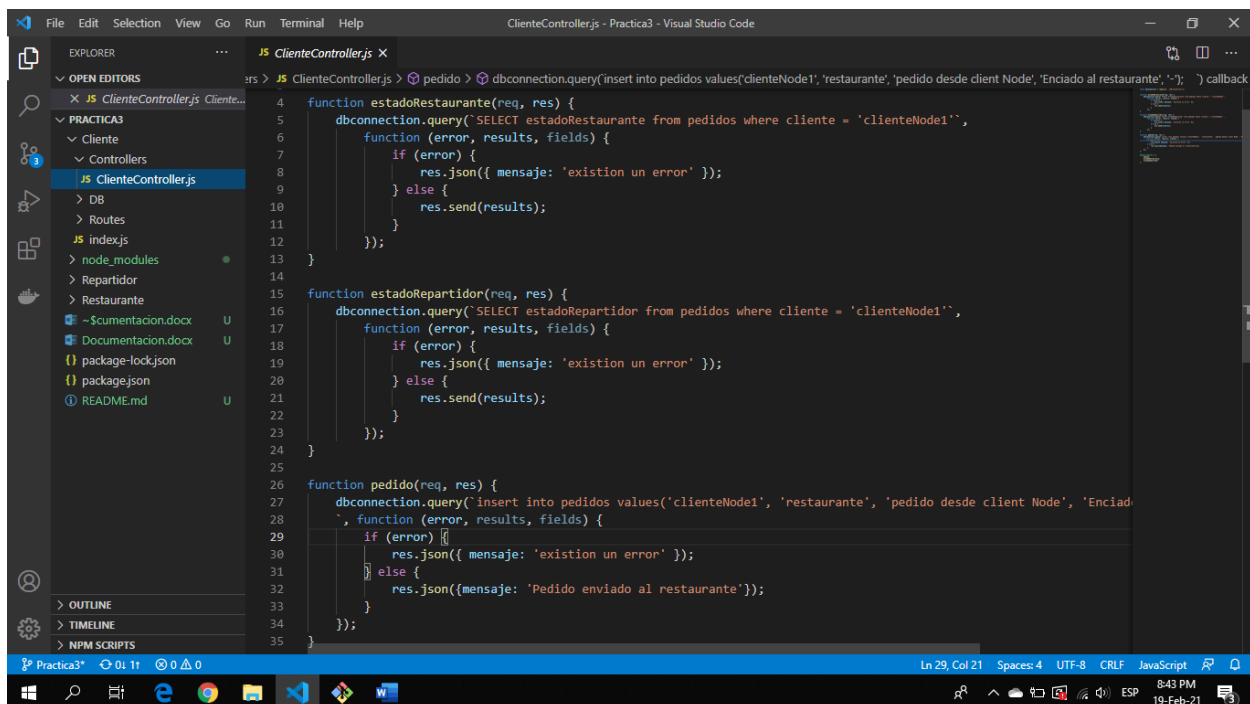


The screenshot shows the Visual Studio Code interface with the 'Rutas' file open. The Explorer sidebar on the left shows the project structure: 'Practica3' > 'Cliente' > 'Routes' > 'ClienteRoutes.js'. The main editor displays the code for 'ClienteRoutes.js'.

```
1 const express = require('express')
2 const router = express.Router();
3 const controller = require('../Controllers/ClienteController');
4
5 router.get('/pedido', controller.pedido)
6 router.get('/estadoRestaurante', controller.estadoRestaurante)
7 router.get('/estadoRepartidor', controller.estadoRepartidor)
8
9
10 module.exports = router;
```

The status bar at the bottom indicates the file is at line 5, column 11, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript syntax.

Controller



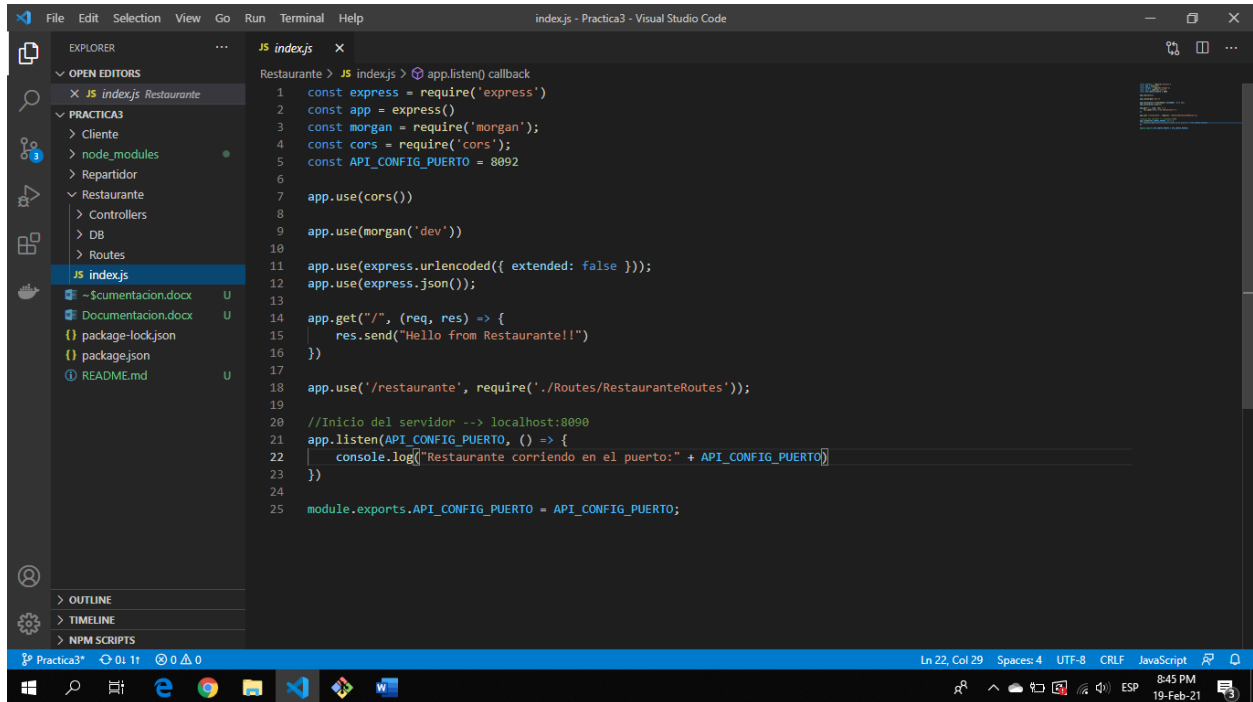
The screenshot shows the Visual Studio Code interface with the 'Controller' file open. The Explorer sidebar on the left shows the project structure: 'Practica3' > 'Cliente' > 'Controllers' > 'ClienteController.js'. The main editor displays the code for 'ClienteController.js'.

```
4 function estadoRestaurante(req, res) {
5     dbconnection.query('SELECT estadoRestaurante from pedidos where cliente = 'clienteNode1'',
6     function (error, results, fields) {
7         if (error) {
8             res.json({ mensaje: 'existance un error' });
9         } else {
10             res.send(results);
11         }
12     });
13 }
14
15 function estadoRepartidor(req, res) {
16     dbconnection.query('SELECT estadoRepartidor from pedidos where cliente = 'clienteNode1'',
17     function (error, results, fields) {
18         if (error) {
19             res.json({ mensaje: 'existance un error' });
20         } else {
21             res.send(results);
22         }
23     });
24 }
25
26 function pedido(req, res) {
27     dbconnection.query('insert into pedidos values('clienteNode1', 'restaurante', 'pedido desde client Node', 'Encliad
28     ', function (error, results, fields) {
29         if (error) {
30             res.json({ mensaje: 'existance un error' });
31         } else {
32             res.json({mensaje: 'Pedido enviado al restaurante'});
33         }
34     });
35 }
```

The status bar at the bottom indicates the file is at line 29, column 21, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript syntax.

Restaurante

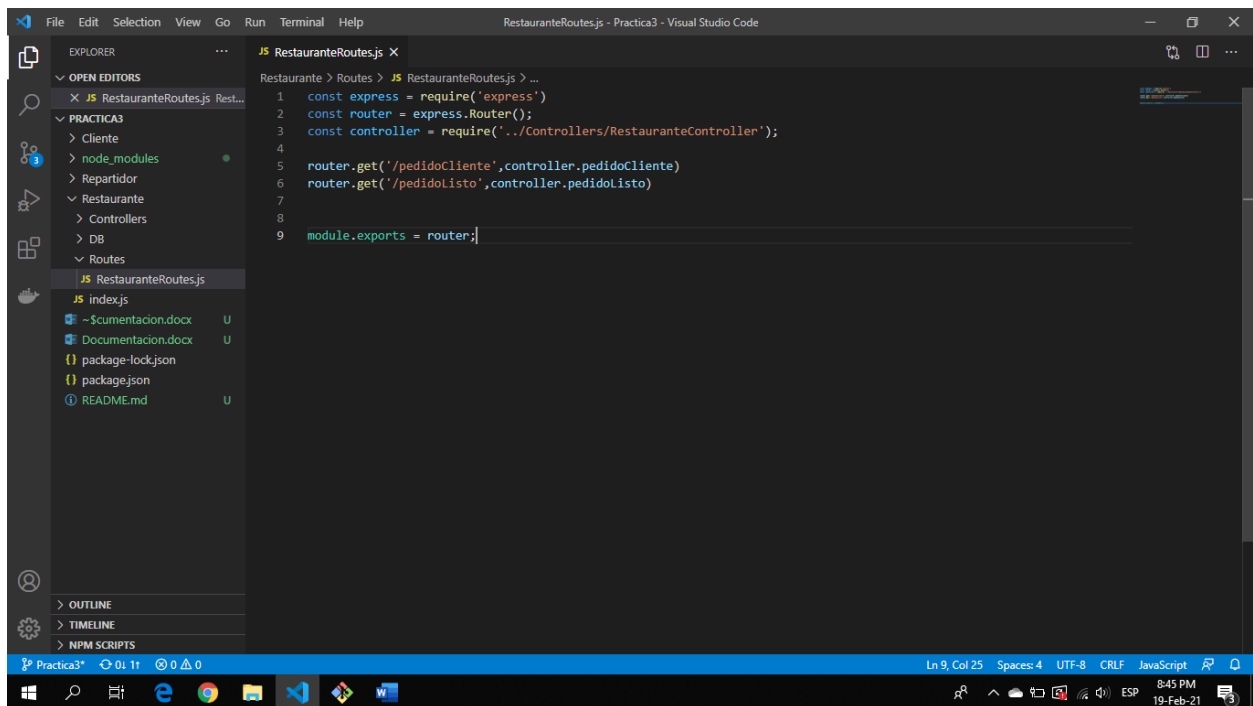
Index.js



The screenshot shows the Visual Studio Code interface with the 'index.js' file open. The Explorer sidebar on the left shows the project structure: 'Practica3' (root) contains 'Cliente', 'node_modules', 'Repartidor', 'Restaurante' (containing 'Controllers', 'DB', 'Routes'), and 'index.js'. The 'index.js' file is selected and its content is displayed in the editor. The code sets up an Express application with CORS, Morgan, and Express.json middleware. It defines a GET route for '/' that sends a 'Hello from Restaurante!' response. The application is configured to listen on port 8090. The status bar at the bottom indicates the file is at line 22, column 29, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript syntax.

```
Restaurante > JS indexjs > app.listen() callback
1  const express = require('express')
2  const app = express()
3  const morgan = require('morgan');
4  const cors = require('cors');
5  const API_CONFIG_PUERTO = 8092
6
7  app.use(cors())
8
9  app.use(morgan('dev'))
10
11 app.use(express.urlencoded({ extended: false }));
12 app.use(express.json());
13
14 app.get('/', (req, res) => {
15   res.send("Hello from Restaurante!")
16 })
17
18 app.use('/restaurante', require('./Routes/RestauranteRoutes'));
19
20 //Inicio del servidor -> localhost:8090
21 app.listen(API_CONFIG_PUERTO, () => {
22   console.log("Restaurante corriendo en el puerto:" + API_CONFIG_PUERTO)
23 })
24
25 module.exports.API_CONFIG_PUERTO = API_CONFIG_PUERTO;
```

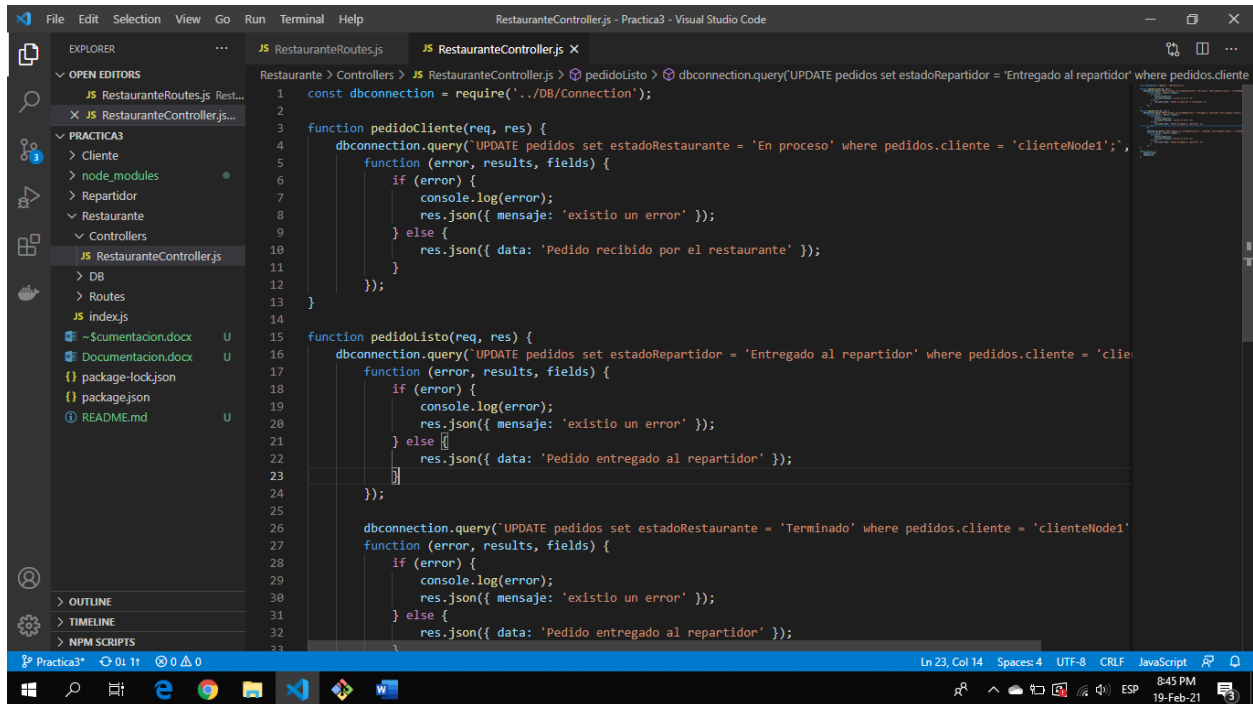
Rutas



The screenshot shows the Visual Studio Code interface with the 'RestauranteRoutes.js' file open. The Explorer sidebar on the left shows the project structure: 'Practica3' (root) contains 'Cliente', 'node_modules', 'Repartidor', 'Restaurante' (containing 'Controllers', 'DB', 'Routes'), and 'index.js'. The 'RestauranteRoutes.js' file is selected and its content is displayed in the editor. The code sets up an Express Router with a controller. It defines two GET routes: '/pedidoCliente' and '/pedidoListo', both using the 'pedidoCliente' controller. The router is exported as 'module.exports = router;'. The status bar at the bottom indicates the file is at line 9, column 25, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript syntax.

```
Restaurante > Routes > JS RestauranteRoutes.js > ...
1  const express = require('express')
2  const router = express.Router();
3  const controller = require('../Controllers/RestauranteController');
4
5  router.get('/pedidoCliente',controller.pedidoCliente)
6  router.get('/pedidoListo',controller.pedidoListo)
7
8
9  module.exports = router;
```

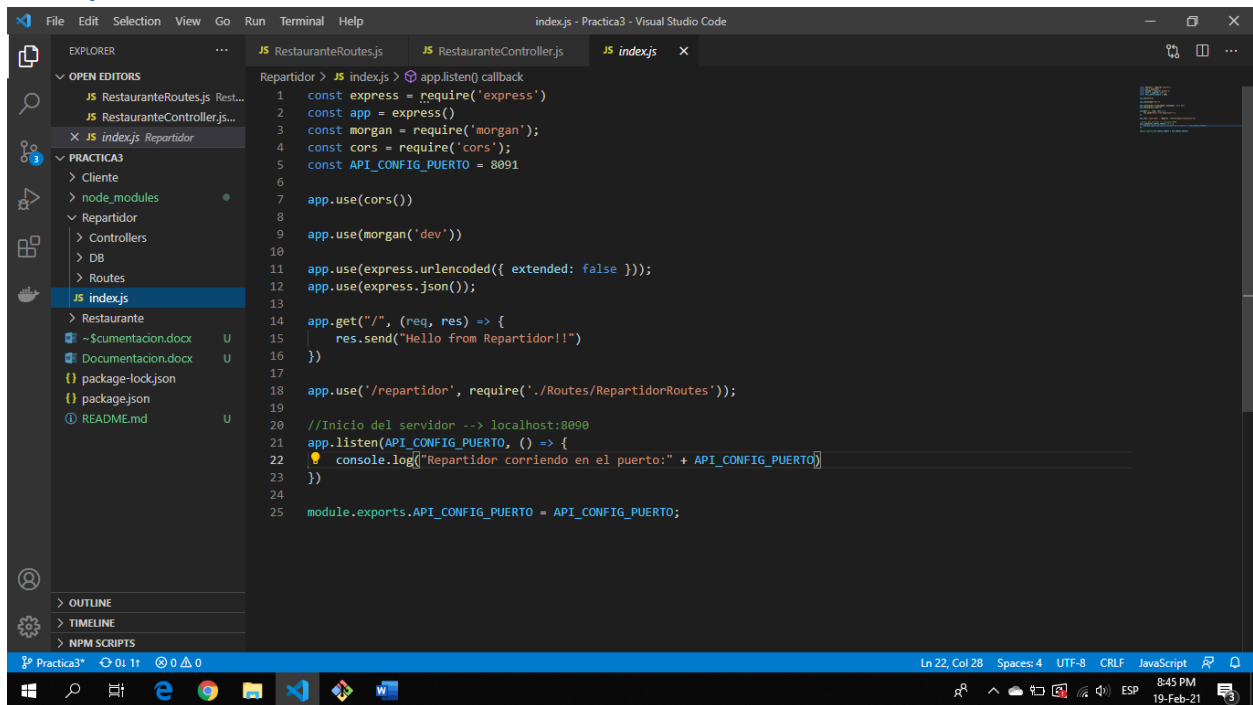
Controller



```
1 const dbconnection = require('../DB/Connection');
2
3 function pedidoCliente(req, res) {
4   dbconnection.query('UPDATE pedidos set estadoRestaurante = 'En proceso' where pedidos.cliente = 'clienteNode1';',
5     function (error, results, fields) {
6       if (error) {
7         console.log(error);
8         res.json({ mensaje: 'existio un error' });
9       } else {
10        res.json({ data: 'Pedido recibido por el restaurante' });
11      }
12    });
13 }
14
15 function pedidoListo(req, res) {
16   dbconnection.query('UPDATE pedidos set estadoRepartidor = 'Entregado al repartidor' where pedidos.cliente = 'clienteNode1';',
17     function (error, results, fields) {
18       if (error) {
19         console.log(error);
20         res.json({ mensaje: 'existio un error' });
21       } else {
22        res.json({ data: 'Pedido entregado al repartidor' });
23      }
24    });
25 }
26
27 dbconnection.query('UPDATE pedidos set estadoRestaurante = 'Terminado' where pedidos.cliente = 'clienteNode1';',
28   function (error, results, fields) {
29     if (error) {
30       console.log(error);
31       res.json({ mensaje: 'existio un error' });
32     } else {
33       res.json({ data: 'Pedido entregado al repartidor' });
34     }
35   });
36 }
```

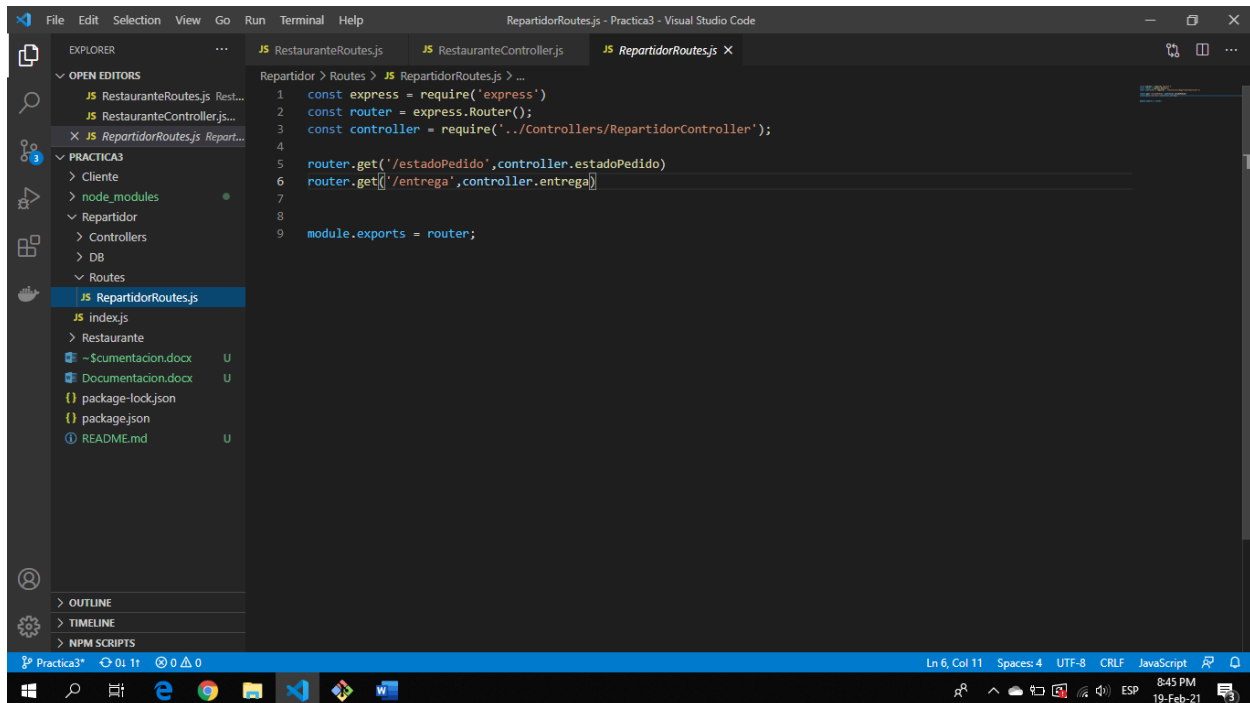
Repartidor

Index.js



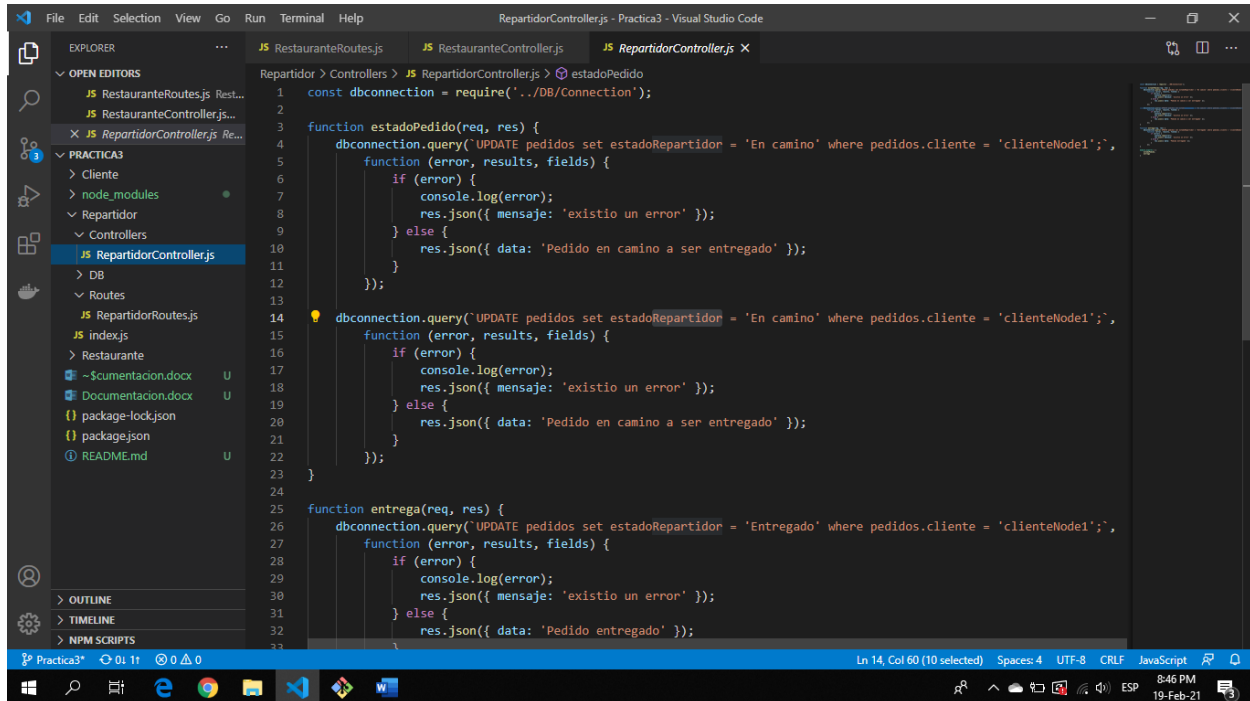
```
1 const express = require('express')
2 const app = express()
3 const morgan = require('morgan');
4 const cors = require('cors');
5 const API_CONFIG_PUERTO = 8091
6
7 app.use(cors())
8
9 app.use(morgan('dev'))
10
11 app.use(express.urlencoded({ extended: false }));
12 app.use(express.json());
13
14 app.get("/", (req, res) => {
15   res.send("Hello from Repartidor!!")
16 })
17
18 app.use('/repartidor', require('./Routes/RepartidorRoutes'));
19
20 //Inicio del servidor --> localhost:8090
21 app.listen(API_CONFIG_PUERTO, () => {
22   console.log("Repartidor corriendo en el puerto:" + API_CONFIG_PUERTO)
23 })
24
25 module.exports.API_CONFIG_PUERTO = API_CONFIG_PUERTO;
```

Rutas



```
1 const express = require('express')
2 const router = express.Router()
3 const controller = require('../Controllers/RepartidorController');
4
5 router.get('/estadoPedido', controller.estadoPedido)
6 router.get('/:entrega', controller.entrega)
7
8
9 module.exports = router;
```

Controller



```
1 const dbconnection = require('../DB/Connection');
2
3 function estadoPedido(req, res) {
4   dbconnection.query('UPDATE pedidos set estadoRepartidor = 'En camino' where pedidos.cliente = 'clienteNode1');
5   function (error, results, fields) {
6     if (error) {
7       console.log(error);
8       res.json({ mensaje: 'existio un error' });
9     } else {
10      res.json({ data: 'Pedido en camino a ser entregado' });
11    }
12  }
13 }
14
15 dbconnection.query('UPDATE pedidos set estadoRepartidor = 'En camino' where pedidos.cliente = 'clienteNode1');
16 function (error, results, fields) {
17   if (error) {
18     console.log(error);
19     res.json({ mensaje: 'existio un error' });
20   } else {
21     res.json({ data: 'Pedido en camino a ser entregado' });
22   }
23 }
24
25 function entrega(req, res) {
26   dbconnection.query('UPDATE pedidos set estadoRepartidor = 'Entregado' where pedidos.cliente = 'clienteNode1');
27   function (error, results, fields) {
28     if (error) {
29       console.log(error);
30       res.json({ mensaje: 'existio un error' });
31     } else {
32       res.json({ data: 'Pedido entregado' });
33     }
34   }
35 }
```

Información

- El cliente corre en el puerto 8090
- El repartidor corre en el puerto 8091
- El restaurante corre en el puerto 8092