

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Software Avanzado

Documentación – Práctica 4

Andrea María López Flores

201404134

Guatemala, 23 de febrero de 2021

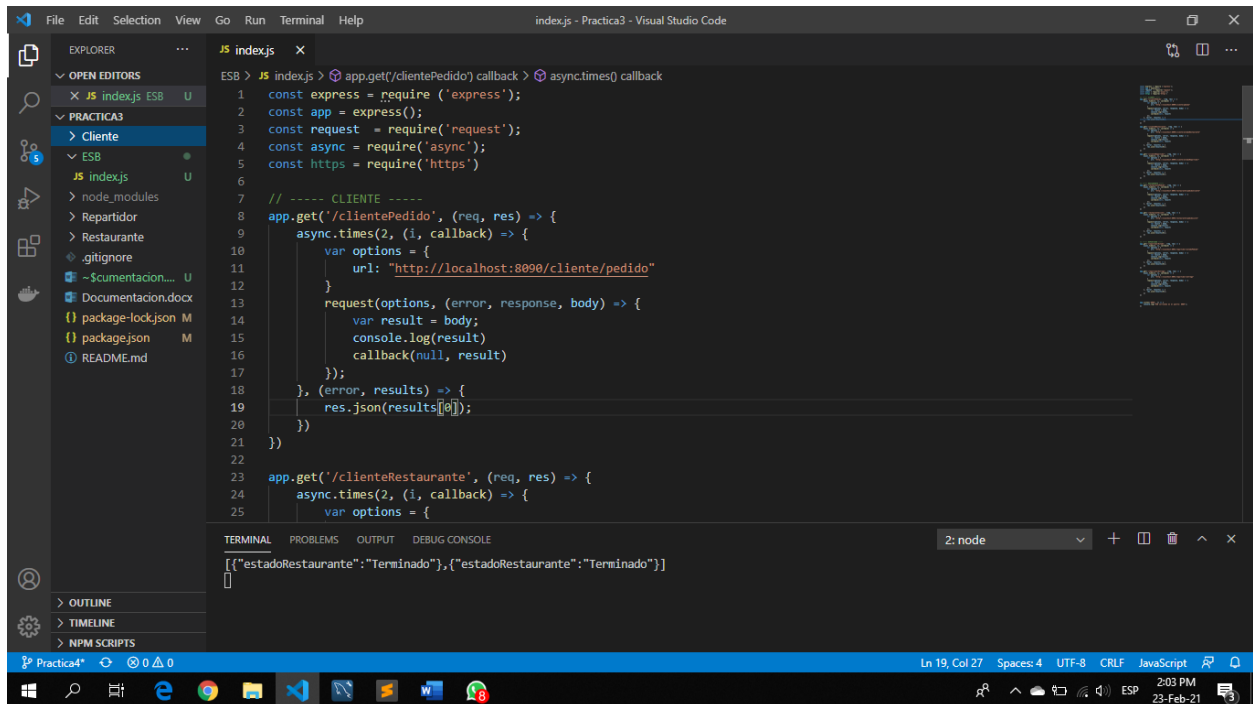
ESB

El servicio encargado de redireccionar las solicitudes es u nuevo servicio que corre en el puerto 8089.

Todo el código se encuentra en index.js de la carpeta ESB.

Index.js

Con comentarios se separan las llamadas al cliente, repartido y restaurante.



```
File Edit Selection View Go Run Terminal Help
index.js - Practica3 - Visual Studio Code

EXPLORER
  OPEN EDITORS
    JS index.js ESB U
  PRACTICA3
    Cliente
    ESB
      JS index.js U
    node_modules
    Repartidor
    Restaurante
    .gitignore
    ~$documentacion... U
    Documentacion.docx
    package-lock.json M
    package.json M
    README.md

  OUTLINE
  TIMELINE
  NPM SCRIPTS

JS index.js
1 const express = require('express');
2 const app = express();
3 const request = require('request');
4 const async = require('async');
5 const https = require('https');
6
7 // ---- CLIENTE ----
8 app.get('/clientePedido', (req, res) => {
9   async.times(2, (i, callback) => {
10     var options = {
11       url: "http://localhost:8090/cliente/pedido"
12     }
13     request(options, (error, response, body) => {
14       var result = body;
15       console.log(result);
16       callback(null, result);
17     });
18   }, (error, results) => {
19     res.json(results[0]);
20   })
21 })
22
23 app.get('/clienteRestaurante', (req, res) => {
24   async.times(2, (i, callback) => {
25     var options = {
26
27     }
28   }, (error, results) => {
29     res.json(results[0]);
30   })
31 })
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

TERMINAL
2: node
[{"estadoRestaurante": "Terminado"}, {"estadoRestaurante": "Terminado"}]
```

```
ESB > JS index.js > app.get('/clientePedido') callback > async.times() callback
54 // ----- RESTAURANTE -----
55 app.get('/restaurantePedido', (req, res) => {
56   async.times(2, (i, callback) => {
57     var options = {
58       url: "http://localhost:8092/restaurante/pedidoCliente"
59     }
60     request(options, (error, response, body) => {
61       var result = body;
62       console.log(result)
63       callback(null, result)
64     });
65   }, (error, results) => {
66     res.json(results[0]);
67   })
68 })
69
70 app.get('/restauranteListo', (req, res) => {
71   async.times(2, (i, callback) => {
72     var options = {
73       url: "http://localhost:8092/restaurante/pedidoListo"
74     }
75     request(options, (error, response, body) => {
76       var result = body;
77       console.log(result)
78       callback(null, result)
79     })
80   })
81 })
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

2: node

```
[{"estadoRestaurante": "Terminado"}, {"estadoRestaurante": "Terminado"}]
```

```
ESB > JS index.js > app.get('/clientePedido') callback > async.times() callback
86 // ----- REPARTIDOR -----
87 app.get('/repartidorEstado', (req, res) => {
88   async.times(2, (i, callback) => {
89     var options = {
90       url: "http://localhost:8091/repartidor/estadoPedido"
91     }
92     request(options, (error, response, body) => {
93       var result = body;
94       console.log(result)
95       callback(null, result)
96     });
97   }, (error, results) => {
98     res.json(results[0]);
99   })
100 })
101
102 app.get('/repartidorEntrega', (req, res) => {
103   async.times(2, (i, callback) => {
104     var options = {
105       url: "http://localhost:8091/repartidor/entrega"
106     }
107     request(options, (error, response, body) => {
108       var result = body;
109       console.log(result)
110       callback(null, result)
111     })
112   })
113 })
```

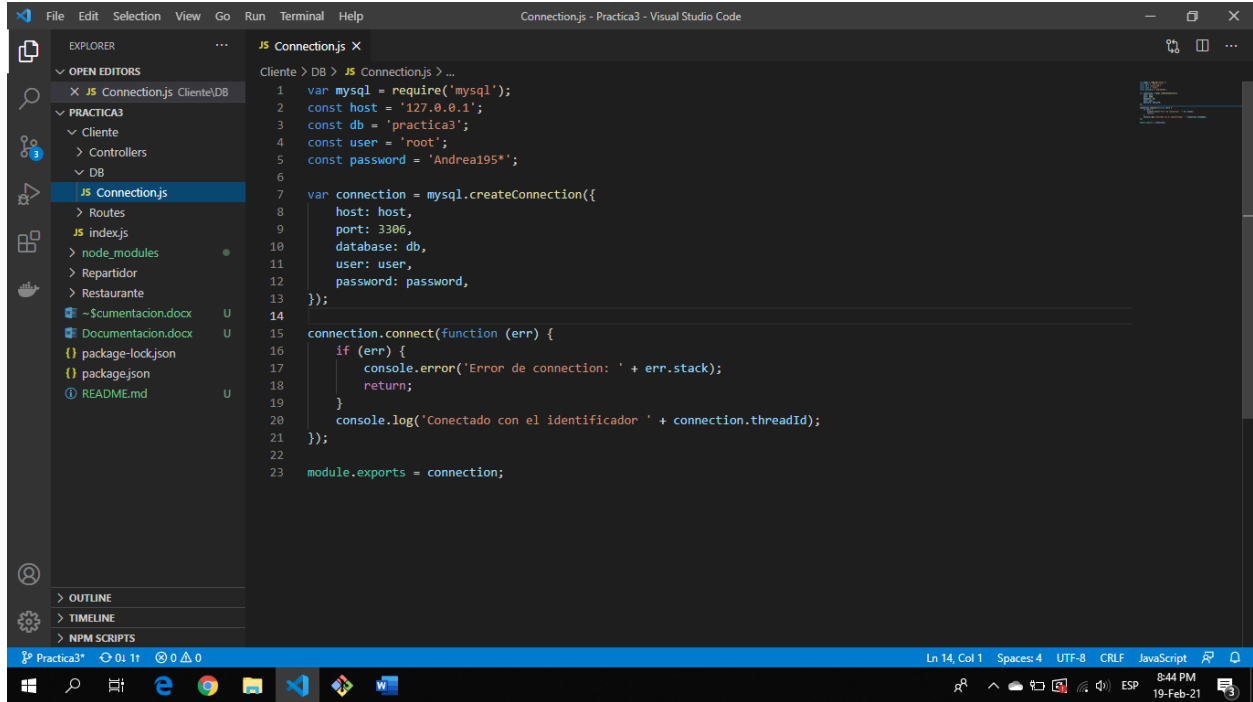
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

2: node

```
[{"estadoRestaurante": "Terminado"}, {"estadoRestaurante": "Terminado"}]
```

Servicios de la Práctica 3

Conexión con la base de datos



The screenshot shows the Visual Studio Code editor with a file named 'Connection.js' open. The file contains JavaScript code for connecting to a MySQL database. The code is as follows:

```
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 connection: ' + err.stack);
18     return;
19   }
20   console.log('Conectado con el identificador ' + connection.threadId);
21 });
22
23 module.exports = connection;
```

The Explorer sidebar on the left shows the project structure, including folders for 'Cliente', 'Routes', 'index.js', 'node_modules', 'Repartidor', and 'Restaurante'. The status bar at the bottom indicates the file is 'Ln 14, Col 1' and the editor is using 'UTF-8' encoding and 'CRLF' line endings.

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

```
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;
```

Controller

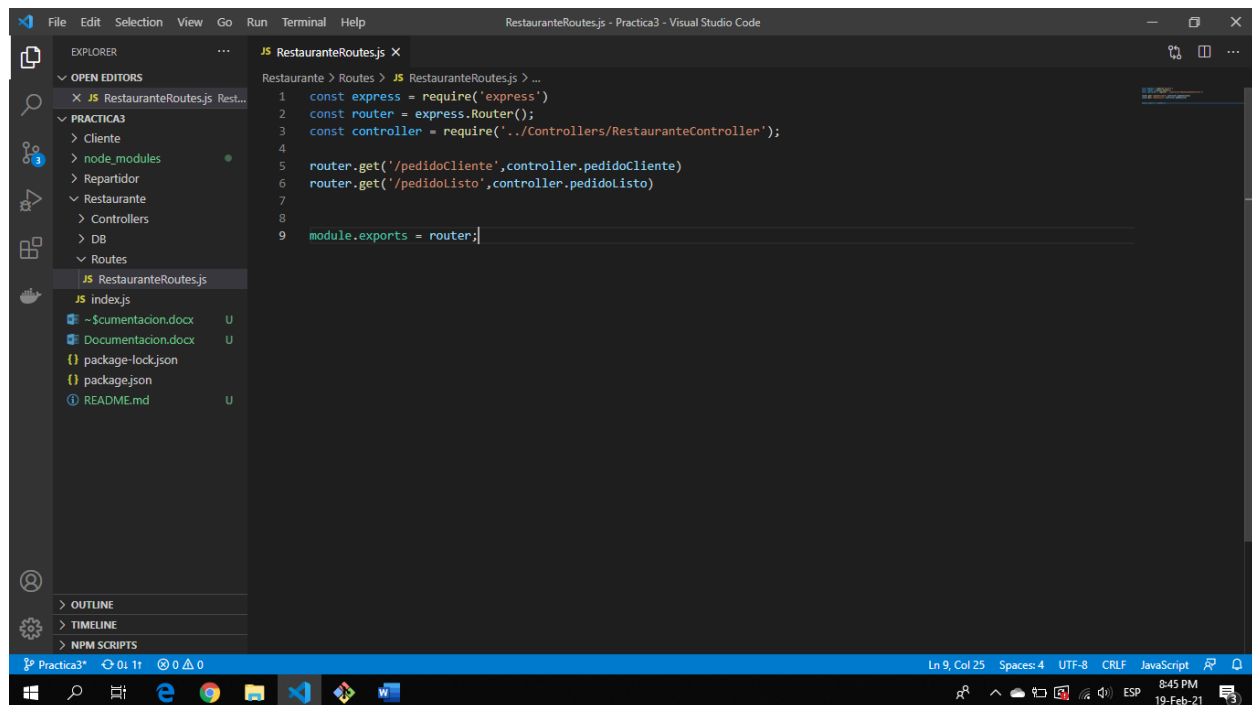
```
1  function estadoRestaurante(req, res) {
2      dbconnection.query("SELECT estadoRestaurante from pedidos where cliente = 'clienteModel'",
3      function (error, results, fields) {
4          if (error) {
5              res.json({ mensaje: 'existance un error' });
6          } else {
7        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

Restaurante

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 = 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;
```

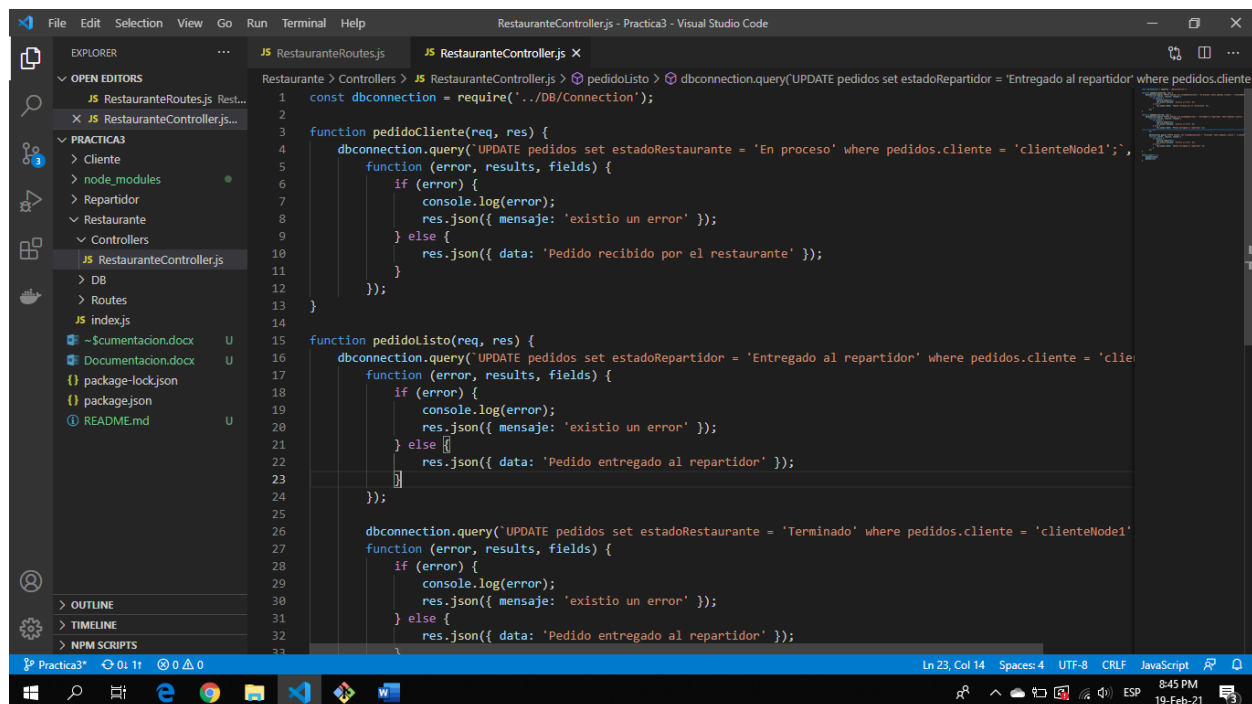
Rutas



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'RestauranteRoutes.js' file is open in the editor. The code defines an Express.js router with two GET routes: one for '/pedidoCliente' and another for '/pedidoListo'. The router is exported as the module's exports.

```
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

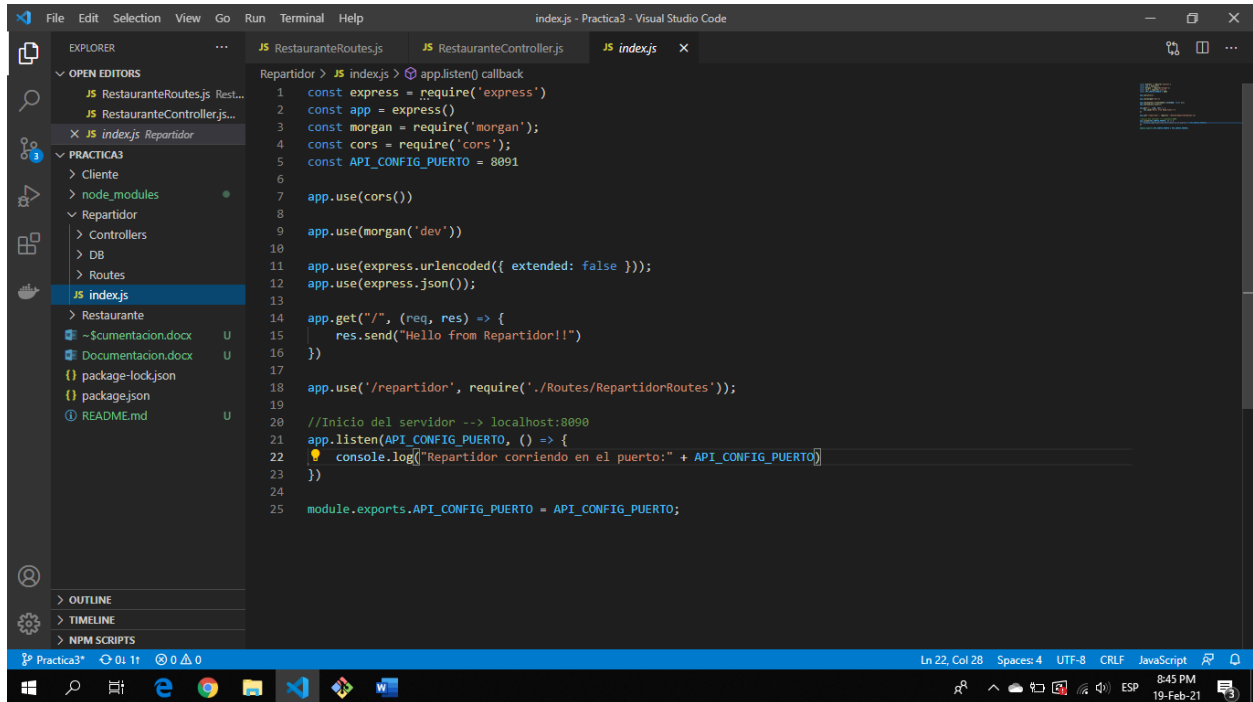


The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'RestauranteController.js' file is open in the editor. The code defines three functions: 'pedidoCliente', 'pedidoListo', and a third function (partially visible) that handles database queries for updating the status of orders. The 'pedidoCliente' function updates the status to 'En proceso', 'pedidoListo' updates it to 'Entregado al repartidor', and the third function updates it to 'Terminado'.

```
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

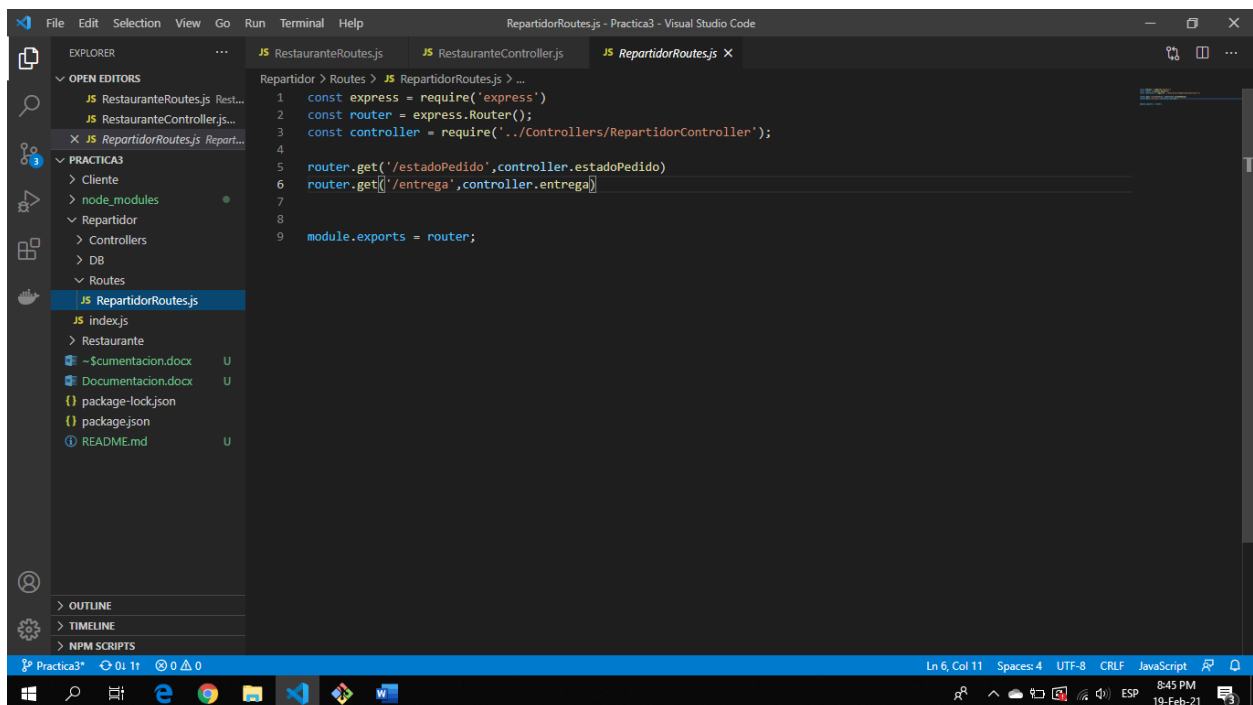


The screenshot shows the Visual Studio Code editor with the 'index.js' file open. The Explorer sidebar on the left shows the project structure: 'PRACTICA3' > 'Repartidor' > 'Routes' > 'index.js'. The main editor area displays the following JavaScript code:

```
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;
```

The status bar at the bottom indicates the file is at line 22, column 28, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript syntax.

Rutas

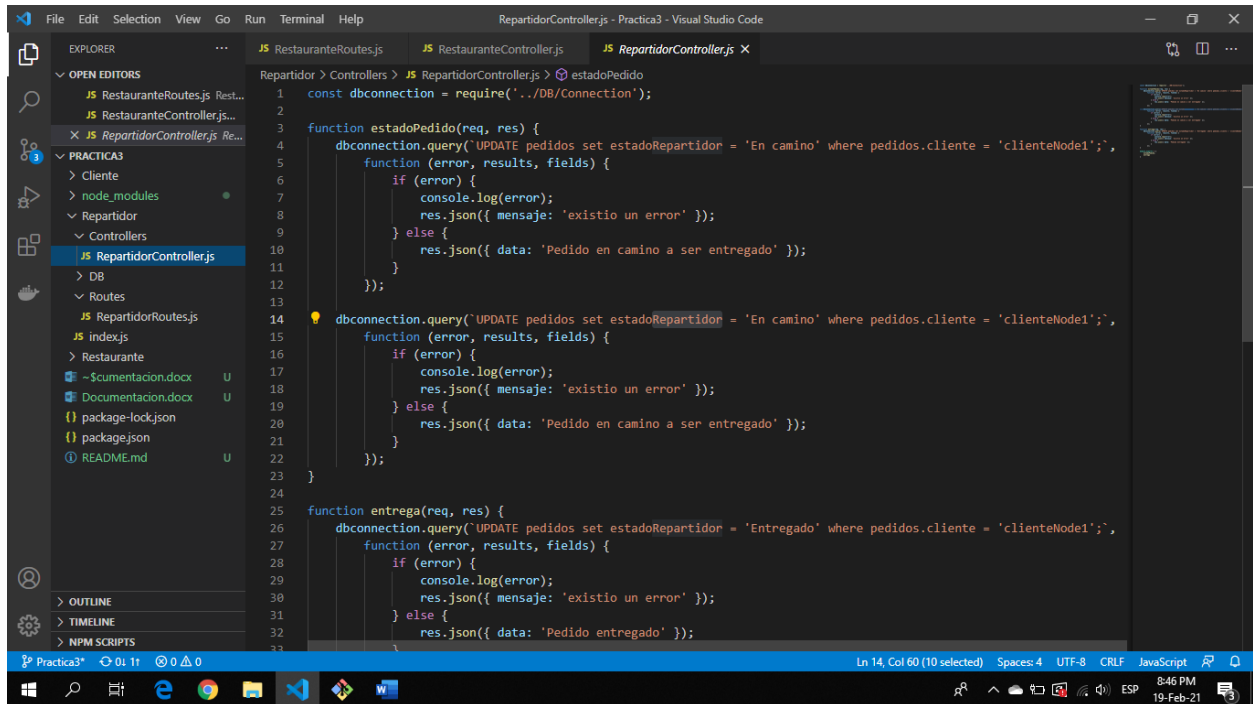


The screenshot shows the Visual Studio Code editor with the 'RepartidorRoutes.js' file open. The Explorer sidebar on the left shows the project structure: 'PRACTICA3' > 'Repartidor' > 'Routes' > 'RepartidorRoutes.js'. The main editor area displays the following JavaScript code:

```
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;
```

The status bar at the bottom indicates the file is at line 6, column 11, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript syntax.

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
26 function entrega(req, res) {
27   dbconnection.query('UPDATE pedidos set estadoRepartidor = 'Entregado' 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' });
34      }
35    });
36  }
37
38 function (req, res) {
39   dbconnection.query('UPDATE pedidos set estadoRepartidor = 'En camino' where pedidos.cliente = 'clienteNode1';',
40     function (error, results, fields) {
41       if (error) {
42         console.log(error);
43         res.json({ mensaje: 'existio un error' });
44       } else {
45        res.json({ data: 'Pedido en camino a ser entregado' });
46      }
47    });
48  }
49 }
```

Información

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