

I solved the above problem in NodeJs with Express framework and MYSQL database.

**index.js:**

```
const express = require('express');

const bodyParser = require('body-parser');

const db = require('./DB_Sequelize/connection');

const Item = require('./DB_Sequelize/Item_model')(db);

let router = express.Router();

let app = express();

app.use(router)

app.use(bodyParser.json())

app.listen(3000, () => {

  console.log('server listening');

});

// POST request

app.post('/boards', (req, res)=> {

  let insertObject = {

    title: req.body.title,

    stage: 1 // setting stage as 1 for every newly creating item

  };

});
```

```

Item.create(insertObject).then(data => {

  Item.findByPk(data.id).then(itemData => {

    // sending 201 status code with newly created item details

    res.status(201).send(itemData.dataValues);

  });

});

});

// PUT request

app.put('/boards/:id', (req, res) => {

  // checking the 'stage' value is between 1 to 3

  if(req.body.stage >= 1 && req.body.stage <= 3) {

    Item.update(req.body, {

      where: {id: req.params.id}

    }).then(data => {

      Item.findByPk(req.params.id).then(itemData => {

        // sending 200 status code with updated item details

        res.status(200).send(itemData.dataValues);

      });

    })

  } else {

    // sending 400 status code with empty response

    res.status(400).send();

  }

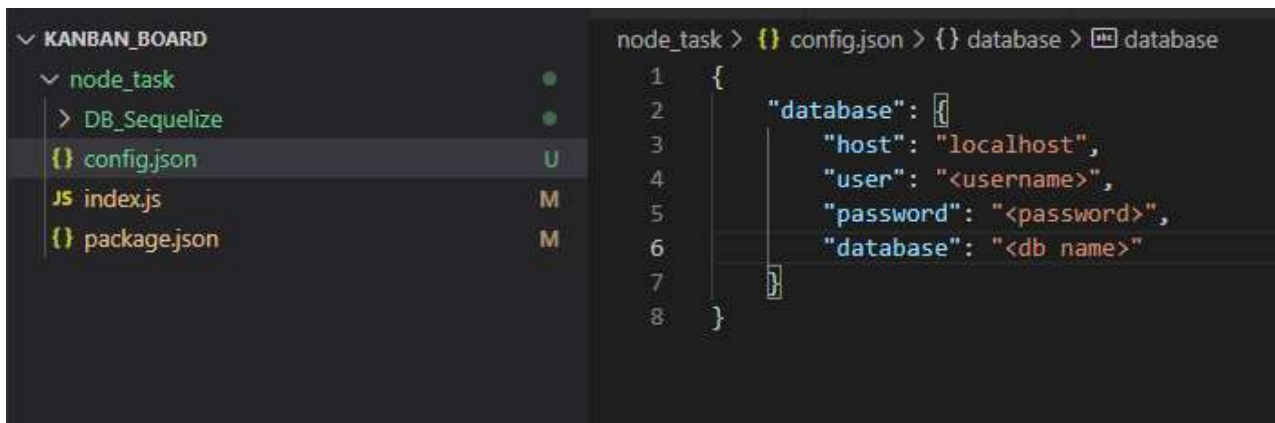
});

```

```
node_task > .js index.js > ...
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const db = require('./DB_Sequelize/connection');
4  const Item = require('./DB_Sequelize/Item_model')(db);
5  let router = express.Router();
6  let app = express();
7  app.use(router)
8
9
10 app.use(bodyParser.json())
11 app.listen(3000, () => {
12   console.log('server listening');
13 });
14
15
16 // POST request
17 app.post('/boards', (req, res)=> {
18   let insertObject = {
19     title: req.body.title,
20     stage: 1 // setting stage as 1 for every newly creating item
21   };
22
23   Item.create(insertObject).then(data => {
24     Item.findByIdPk(data.id).then(itemData => {
25       // sending 201 status code with newly created item details
26       res.status(201).send(itemData.dataValues);
27     });
28   });
29 });
30
31 // PUT request
32 app.put('/boards/:id', (req, res) => {
33   // checking the 'stage' value is between 1 to 3
34   if(req.body.stage >= 1 && req.body.stage <= 3) {
35     Item.update(req.body, {
36       where: {id: req.params.id}
37     }).then(data => {
38       Item.findByIdPk(req.params.id).then(itemData => {
39         // sending 200 status code with updated item details
40         res.status(200).send(itemData.dataValues);
41       });
42     })
43   } else {
44     // sending 400 status code with empty response
45     res.status(400).send();
46   }
47 });
48
```

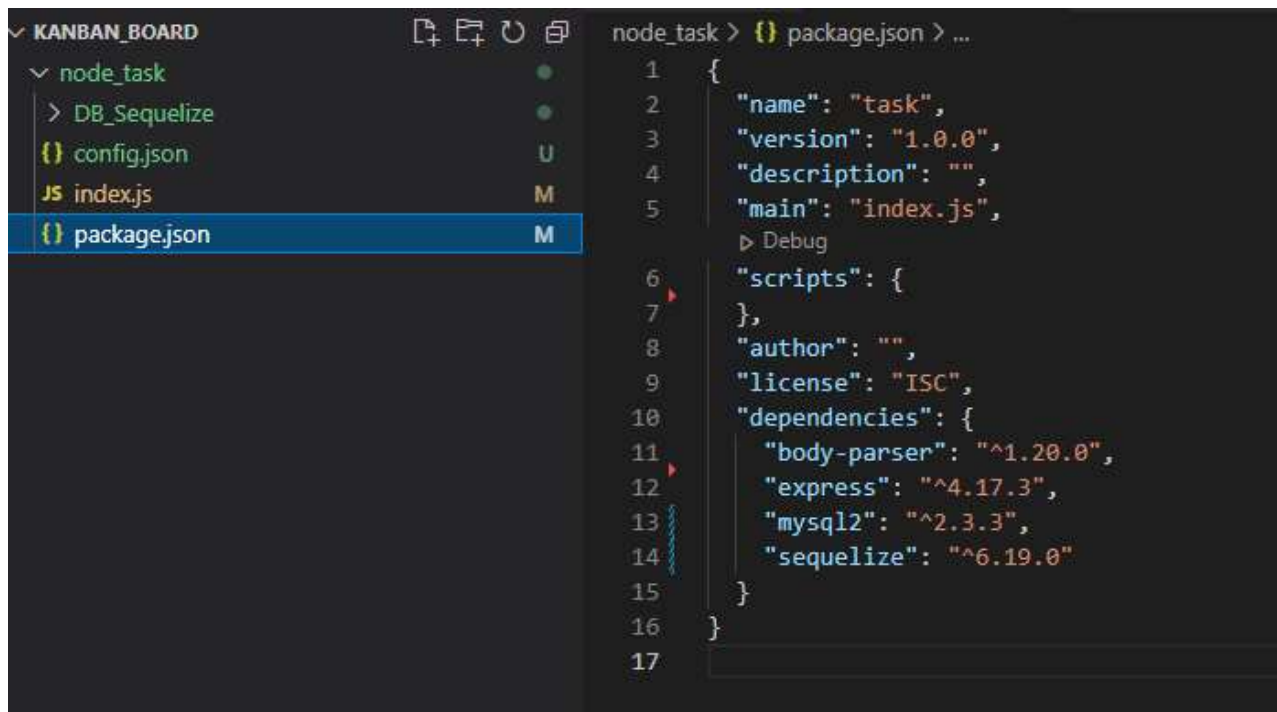
## config.json:

```
{
  "database": {
    "host": "localhost",
    "user": "<username>",
    "password": "<password>",
    "database": "<db name>"
  }
}
```



### package.json:

```
{
  "name": "task",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.0",
    "express": "^4.17.3",
    "mysql2": "^2.3.3",
    "sequelize": "^6.19.0"
  }
}
```



## DB\_Sequelize:

### connection.js:

```
const config = require('../config.json');
```

```
const Sequelize = require('sequelize');
```

```
// you can configure your DB credentials in 'config.json'
```

```
const host = config.database.host;
```

```
const database = config.database.database;
```

```
const userName = config.database.user;
```

```
const password = config.database.password;
```

```
// local DB is connecting using 'Sequelize'
```

```
const sequelize = new Sequelize(database, userName, password, {
```

```
  host: host,
```

```
  dialect: 'mysql',
```

```
  logging: false,
```

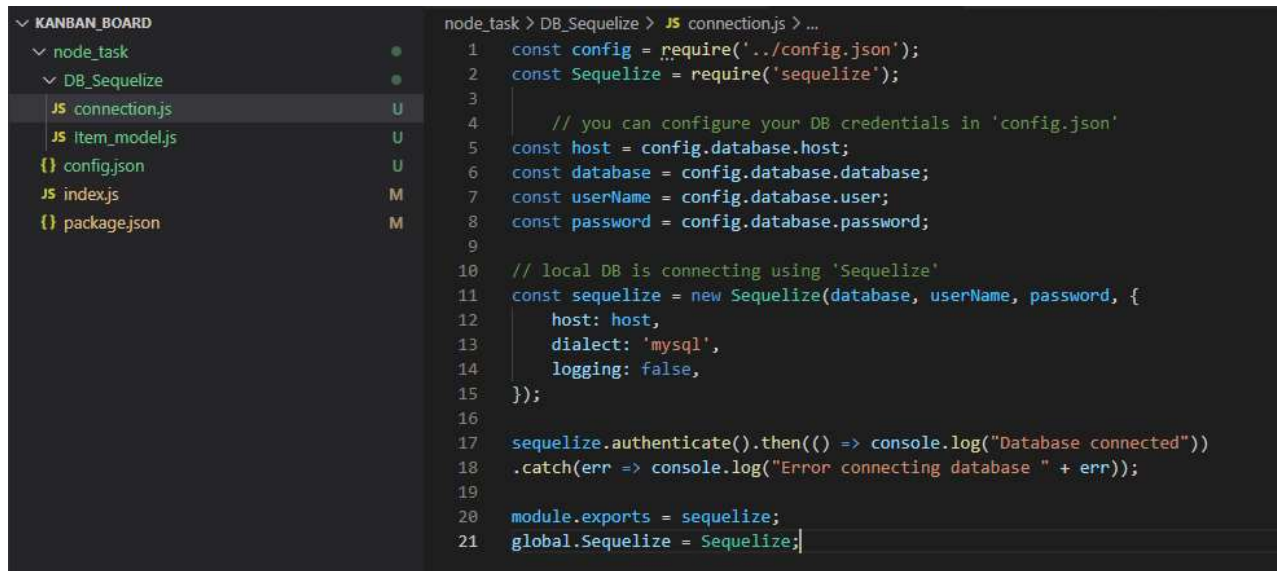
```
});
```

```
sequelize.authenticate().then(() => console.log("Database connected"))
```

```
.catch(err => console.log("Error connecting database " + err));
```

```
module.exports = sequelize;
```

```
global.Sequelize = Sequelize;
```



```
node_task > DB_Sequelize > JS connection.js > ...
1  const config = require('../config.json');
2  const Sequelize = require('sequelize');
3
4  // you can configure your DB credentials in 'config.json'
5  const host = config.database.host;
6  const database = config.database.database;
7  const userName = config.database.user;
8  const password = config.database.password;
9
10 // local DB is connecting using 'Sequelize'
11 const sequelize = new Sequelize(database, userName, password, {
12   host: host,
13   dialect: 'mysql',
14   logging: false,
15 });
16
17 sequelize.authenticate().then(() => console.log("Database connected"))
18 .catch(err => console.log("Error connecting database " + err));
19
20 module.exports = sequelize;
21 global.Sequelize = Sequelize;
```

## Item\_model.js:

```
module.exports = (db) => db.define(
```

```
  "items",
```

```
  {
```

```
    id: {
```

```
      type: Sequelize.INTEGER,
```

```
      allowNull: false,
```

```
      primaryKey: true,
```

```
      autoIncrement: true, // id is automatically incrementing with unique integers
```

```
    },
```

```
    title: {
```

```
      type: Sequelize.STRING,
```

```
      allowNull: false,
```

```
    },
```

```

stage: {

  type: Sequelize.INTEGER,

  allowNull: false,

},

},

{

  timestamps: false,

  freezeTableName: true,

}

);

```

```

node_task > DB_Sequelize > JS_item_model.js > <unknown> > exports > id
1  module.exports = (db) => db.define(
2    "items",
3    {
4      id: {
5        type: Sequelize.INTEGER,
6        allowNull: false,
7        primaryKey: true,
8        autoIncrement: true, // id is automatically incrementing with unique integers
9      },
10     title: {
11       type: Sequelize.STRING,
12       allowNull: false,
13     },
14     stage: {
15       type: Sequelize.INTEGER,
16       allowNull: false,
17     },
18   },
19   {
20     timestamps: false,
21     freezeTableName: true,
22   }
23 );
24

```

**Database schema and table data:**

kanban\_board\_db

Tables

Items

Views

Stored Procs

Functions

Triggers

Events

1 Messages

2 Table Data

3 Info

Format: ☒ HTML ☐ Text/Detailed

Refresh

Table: items

Columns (3)

Calculate Optimal Datatypes

Find the optimal datatypes for this table by reading existing data. [Read more](#)

	Field	Type	Comment
	id	int(11) NOT NULL	
	title	varchar(100) NOT NULL	
	stage	int(11) NOT NULL	

Indexes (1)

Find Redundant Indexes

Find the redundant indexes of this table. [Read more](#)

	Indexes	Columns	Index Type
	PRIMARY	id	Unique

1 Messages

2 Table Data

3 Info

id

title

stage

## Postman request screenshots:

*POST request:*

http://localhost:3000/boards

Save

POST

http://localhost:3000/boards

Send

Params

Auth

Headers (8)

Body

Pre-req.

Tests

Settings

raw

JSON

1

2

3

1

2

3

4

5

201 Created

38 ms

289 B

Save Response

*PUT request:*



http://localhost:3000/boards/1

PUT http://localhost:3000/boards/1

Params Auth Headers (8) Body Pre-req. Tests Settings

raw JSON

```
1 [
2   "stage": 2
3 ]
```

Body

200 OK 24 ms 284 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "title": "Create a new project",
4   "stage": 2
5 }
```

http://localhost:3000/boards/1

PUT http://localhost:3000/boards/1

Params Auth Headers (8) Body Pre-req. Tests Settings

raw JSON

```
1 [
2   "stage": 25
3 ]
```

Body

400 Bad Request 17 ms 154 B Save Response

Pretty Raw Preview Visualize Text

```
1
```

Thank you.