

# Отчет

## Инициализируем проект

```
DIS {} package-lock.json > {} packages > {} > {} dependencies
> node_modules
{} package-lock.json
{} package.json
JS server.js
2 "name": "logreg",
3 "version": "1.0.0",
4 "lockfileVersion": 3,
5 "requires": true,
6 "packages": {
7   "": {
8     "name": "logreg",
9     "version": "1.0.0",
10    "license": "ISC",
11    "dependencies": {
12      "bcrypt": "^5.1.1",
13      "bcryptjs": "^2.4.3",
14      "body-parser": "^1.20.3",
15      "mssql": "^11.0.1",
16      "ejs": "^3.1.10",
17      "cors": "^2.8.5",
18      "express": "^4.21.2",
19      "express-session": "^1.18.1",
20      "pg": "^8.13.1",
21      "nodemon": "^3.1.9",
22      "pg-hstore": "^2.3.4",
23      "sequelize": "^6.37.5",
24      "sqlite3": "^5.1.7"
25    }
26  },
27  ...
28 }
```

## Базовые настройки сервера

```
EXPLORER ... JS server.js {} package-lock.json
DIS JS server.js > ...
> node_modules
{} package-lock.json
{} package.json
JS server.js
3 const path = require('path');
4 const app = express();
5 const PORT = 3000;
6
7 app.set('view engine', 'ejs');
8 app.set('views', path.join(__dirname, 'views'));
9
10 const sequelize = new Sequelize({
11   dialect: 'sqlite',
12   storage: 'database.sqlite',
13 });
14
15 |
```

## Создаем модель данных

```
15 const Product = sequelize.define('Product', {
16   name: { type: DataTypes.STRING, allowNull: false },
17   price: { type: DataTypes.FLOAT, allowNull: false },
18 });
19 const Category = sequelize.define('Category', {
20   name: { type: DataTypes.STRING, allowNull: false },
21 });
22 const Supplier = sequelize.define('Supplier', {
23   name: { type: DataTypes.STRING, allowNull: false },
24   contact: { type: DataTypes.STRING },
25 });
26 Product.belongsTo(Category);
27 Product.belongsTo(Supplier);
28 Category.hasMany(Product);
29 Supplier.hasMany(Product);
```

## Подключаем статические файлы и прописываем маршруты

```
31 app.get('/', async (req, res) => {
32   const products = await Product.findAll({
33     include: [Category, Supplier],
34   });
35   res.render('index', { products });
36 });
37 app.get('/add-category', (req, res) => {
38   res.render('add-category');
39 });
40 app.get('/add-supplier', (req, res) => {
41   res.render('add-supplier');
42 });
43 app.get('/add-product', async (req, res) => {
44   try {
45     const categories = await Category.findAll();
46     const suppliers = await Supplier.findAll();
47     res.render('add-product', { categories, suppliers });
48   } catch (error) {
49     console.error('Error fetching categories or suppliers:', error);
50     res.status(500).send('Internal Server Error');
51   }
52 });
```

```
54 app.get ('/edit-product/:id', async (req, res) => {
55   try {
56     const productId = req.params.id;
57     const product = await Product.findByPk(productId, {
58       include: [Category, Supplier],
59     });
60     const categories = await Category.findAll();
61     const suppliers = await Supplier.findAll();
62     res.render ('edit-product', { product, categories, suppliers });
63   } catch (error) {
64     console.error('Error fetching product for edit:', error);
65     res.status(500).send('Internal Server Error');
66   }
67 });
```

## Пост запросы

```
69 app.post('/add-category', express.urlencoded({ extended: true }), async (req, res) => {
70   const { name } = req.body;
71   if (name) {
72     await Category.create({ name });
73   }
74   res.redirect('/');
75 });
76 app.post('/add-supplier', express.urlencoded({ extended: true }), async (req, res) => {
77   const { name, contact } = req.body;
78   if (name) {
79     await Supplier.create({ name, contact });
80   }
81   res.redirect('/');
82 });
83
```

```

84 app.post('/add-product', express.urlencoded({ extended: true })), async (req, res) => {
85   const { name, price, categoryId, supplierId } = req.body;
86   if (name && price && categoryId && supplierId) {
87     try {
88       await Product.create({
89         name,
90         price,
91         CategoryId: categoryId,
92         SupplierId: supplierId,
93       });
94       res.redirect('/');
95     } catch (error) {
96       console.error('Error adding product:', error);
97       res.status(500).send('Internal Server Error');
98     }
99   } else {
100     res.status(400).send('All fields are required');
101   }
102 });

```

```

104 app.post('/delete-product/:id', async (req, res) => {
105   try {
106     const productId = req.params.id;
107     await Product.destroy({
108       where: { id: productId },
109     });
110     res.redirect('/');
111   } catch (error) {
112     console.error('Error deleting product:', error);
113     res.status(500).send('Internal Server Error');
114   }
115 });
116
117 app.post('/edit-product/:id', express.urlencoded({ extended: true })), async (req, res) => {
118   try {
119     const productId = req.params.id;
120     const { name, price, categoryId, supplierId } = req.body;
121
122     await Product.update(
123       { name, price, CategoryId: categoryId, SupplierId: supplierId },
124       { where: { id: productId } }
125     );
126
127     res.redirect('/');
128   } catch (error) {
129     console.error('Error updating product:', error);
130     res.status(500).send('Internal Server Error');
131   }
132 });

```

## Начальный набор данных и подключение к серверу

```

134 (async () => {
135   try {
136     await sequelize.sync({ force: true });
137
138     const category1 = await Category.create({ name: 'Electronics' });
139     const category2 = await Category.create({ name: 'Books' });
140
141     const supplier1 = await Supplier.create({ name: 'TechCorp', contact: 'techcorp@example.com' });
142     const supplier2 = await Supplier.create({ name: 'BookStore', contact: 'contact@bookstore.com' });
143
144     await Product.create({ name: 'Laptop', price: 1209.99, CategoryId: category1.id, SupplierId: supplier1.id });
145     await Product.create({ name: 'Smartphone', price: 799.49, CategoryId: category1.id, SupplierId: supplier1.id });
146     await Product.create({ name: 'Science Fiction Book', price: 19.99, CategoryId: category2.id, SupplierId: supplier2.id });
147
148     app.listen(PORT, () => console.log(`Server is running on http://localhost:${PORT}`));
149   } catch (error) {
150     console.error('Error initializing the application:', error);
151   }
152 })();
153

```

## Index.ejs

```
10 <body>
11   <table>
12     <thead>
13       <th>Name</th>
14       <th>Price</th>
15       <th>Category</th>
16       <th>Supplier</th>
17       <th>Actions</th>
18     </tr>
19   </thead>
20   <tbody>
21     <%products.forEach (product=> { %>
22       <tr>
23         <td><%= product.name %></td>
24         <td><%= product.price.toFixed(2) %></td>
25         <td><%= product.Category.name %></td>
26         <td><%= product.Supplier.name %></td>
27         <td>
28           <form action="/delete-product/<%= product.id %>" method="POST" style="display:inline;">
29             <button type="submit">Удалить</button>
30           </form>
31           <button onclick="window.location.href='/edit-product/<%= product.id %>'">Изменить</button>
32         </td>
33       </tr>
34     <% }) %>
35   </tbody>
36 </table>
37
38 </body>
```

```
37   <div class="buttons">
38     <button onclick="window. location.href='/add-product'">Добавить продукт </button>
39     <button onclick="window. location.href='/add-category'">Добавить категорию</button>
40     <button onclick="window. location.href='/add-supplier'">Добавить поставщика</button>
41   </div>
```

## Edit-product.ejs

```
8   <body>
9     <h1>Изменить продукт</h1>
10    <form action="/edit-product/<%= product.id %>" method="POST" >
11
12      <label For="name">Название продукта: </label>
13      <input type="text" id="name" name="name" value="<%= product.name %>" required>
14
15      <label for="price">Цена:</label>
16      <input type="number" id="price" name="price" value="<%= product.price %>" step="0.01" required>
17
18      <label for="category">Выберите категорию:</label>
19      <select id="category" name="categoryId" required>
20        <% categories.forEach(category => { %>
21        <option value="<%= category.id %>" <%= category.id === product.CategoryId ? 'selected' : '' %>><%= category.name %></option>
22        <% }) %>
23      </select>
24
25      <label for="supplier">Выберите поставщика:</label>
26      <select id="supplier" name="supplierId" required>
27        <% suppliers.forEach(supplier => { %>
28        <option value="<%= supplier.id %>" <%= supplier.id === product.SupplierId ? 'selected' : '' %>><%= supplier.name %></option>
29        <% }) %>
30      </select>
31
32      <button type="submit">Сохранить</button>
33    </form>
```

## Add-supplier.ejs

```
9
10 <body>
11   <h1>Добавить поставщика</h1>
12   <form action="/add-supplier" method="POST">
13     <label for="name">Название поставщика: </label>
14     <input type="text" id="name" name="name" required>
15     <label for="contact">Контактная информация: </label>
16     <input type="text" id="contact" name="contact">
17     <button type="submit">Сохранить</button>
18   </form>
19 </body>
20
21 </html>
```

## Add-category.ejs

```
11   <h1>Добавить категорию</h1>
12   <form action="/add-category" method="POST">
13     <label for="name">Название категории:</label>
14     <input type="text" id="name" name="name" required>
15     <button type="submit">Сохранить</button>
16   </form>
17 </body>
```

## Add-product.ejs

```
10 <body>
11   <h1>Добавить продукт</h1>
12   <form action="/add-product" method="POST">
13     <label for="name">Название продукта:</label>
14     <input type="text" id="name" name="name" required>
15
16     <label for="price">Цена: </label>
17     <input type="number" id="price" name="price" step="0.81" required>
18
19     <label for="category">Выберите категорию:</label>
20     <select id="category" name="categoryId" required>
21       <% categories.forEach (category => { %> c
22         <option value="<%= category.id %>"><%= category.name %></option>
23       <% }) %>
24     </select>
25
26     <label for="supplier">Выберите поставщика: </label>
27     <select id="supplier" name="supplierId" required>
28       <% suppliers.forEach(supplier => { %>
29         <option value="<%= supplier.id %>"><%= supplier.name %></option>
30       <% }) %>
31     </select>
32
33     <button type="submit">Сохранить</button>
34   </form>
35 </body>
```

Результаты работы

SQL ▾

< 1 / 1 > 1 - 2 of 2

id	name	contact	createdAt
1	TechCorp	techcorp@example.com	2024-12-16 21:29:30.123 +00:00
2	BookStore	contact@bookstore.com	2024-12-16 21:29:30.137 +00:00

SQL ▾

< 1 / 1 > 1 - 3 of 3

id	name	price	createdAt	updatedAt	CategoryId	Supplier
1	Laptop	1209.99	2024-12-16 21:29:30.147 +00:00	2024-12-16 21:29:30.147 +00:00	1	1
2	Smartphone	799.49	2024-12-16 21:29:30.160 +00:00	2024-12-16 21:29:30.160 +00:00	2	1
3	Science Fiction Book	19.99	2024-12-16 21:29:30.172 +00:00	2024-12-16 21:29:30.172 +00:00	2	2

SQL ▾

< 1 / 1 > 1 - 2 of 2

id	name	createdAt	updatedAt
1	Electronics	2024-12-16 21:29:30.082 +00:00	2024-12-16 21:29:30.082 +00:00
2	Books	2024-12-16 21:29:30.107 +00:00	2024-12-16 21:29:30.107 +00:00

Name	Price	Category	Supplier	Actions	
Laptop	1209.99	Electronics	TechCorp	Удалить	Изменить
Smartphone	799.49	Books	TechCorp	Удалить	Изменить
Science Fiction Book	19.99	Books	BookStore	Удалить	Изменить
Добавить продукт		Добавить категорию		Добавить поставщика	

Добавить продукт

Название продукта:

Цена:

Выберите категорию: >Electronics ▾

Выберите поставщика: TechCorp ▾

Сохранить

Добавить категорию

Название категории:

Сохранить

# Добавить поставщика

Название поставщика:

контактная информация:

Сохранить