

## Frontend & Backend разработка

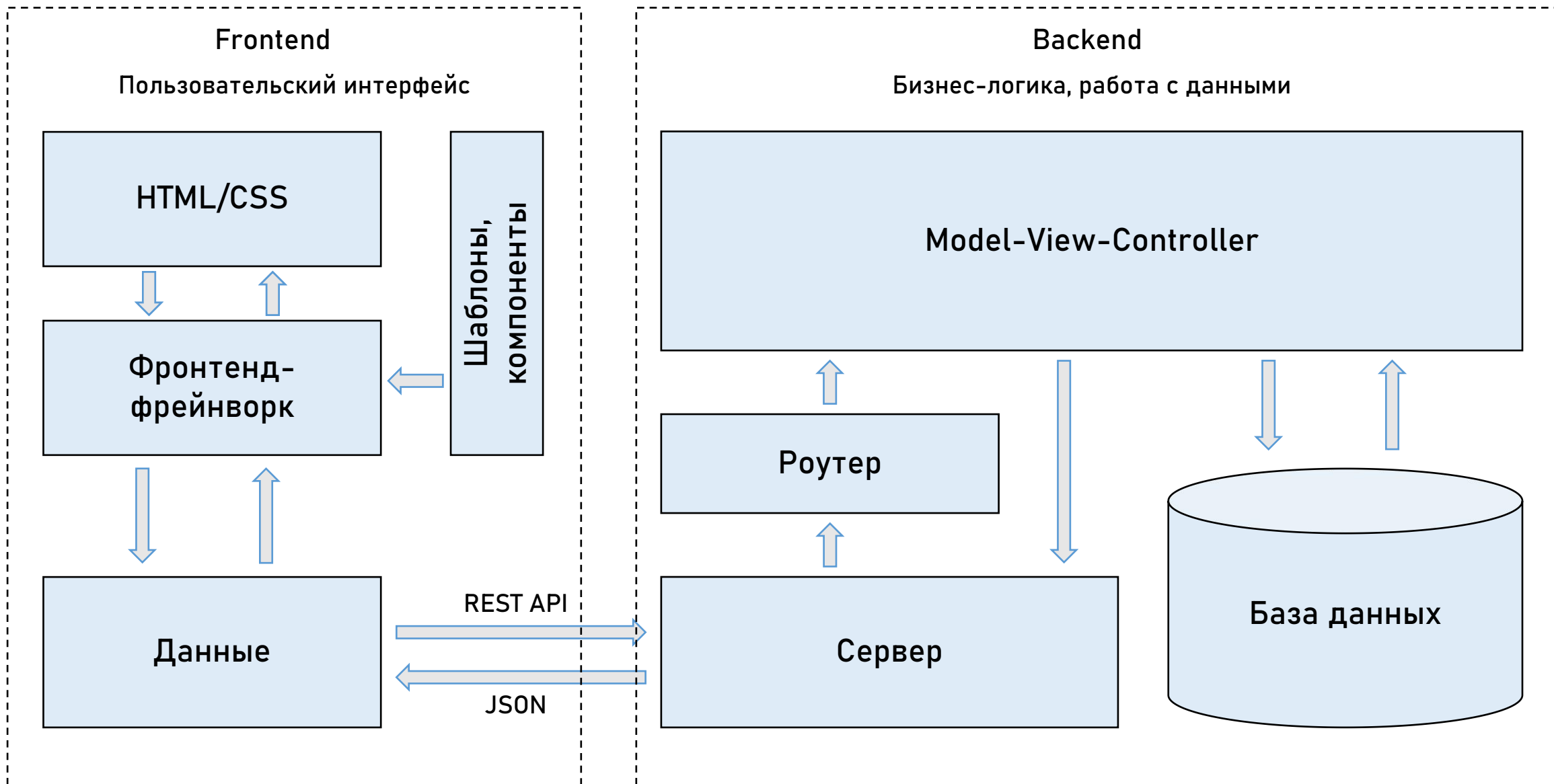
### Лекция 1

## Архитектура веб-приложения

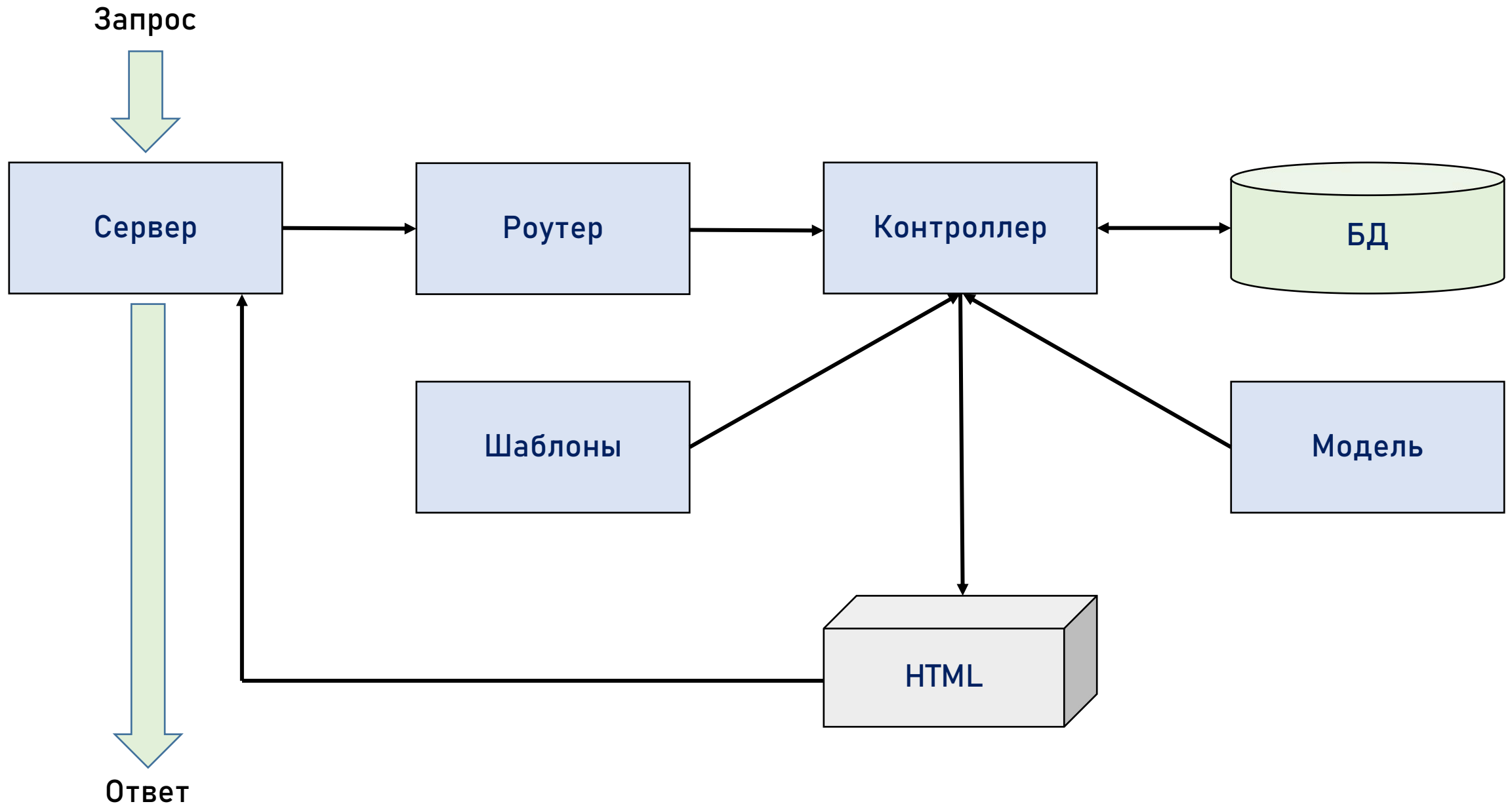
Р.В. Шамин

профессор кафедры индустриального программирования

# Что такое веб-приложение?



# Архитектура нашего приложения



# Создание приложения

npm init

PS C:\temp\Node> npm init  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields  
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (node) items

version: (1.0.0)

description:

entry point: (index.js)

test command:

git repository:

keywords:

author:

license: (ISC)

About to write to C:\temp\Node\package.json:

```
{
  "name": "items",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Is this OK? (yes)

package.json

До начала создания  
приложения нужно установить:

1. Платформу Node.js
2. БД данных MongoDB
3. IDE Visual Studio code

# Установка модулей

```
Windows PowerShell
PS C:\temp\Node> npm install express --save

added 58 packages, and audited 59 packages in 861ms

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\temp\Node> █
```

## Установка Express

сервер для Node.js

```
Выбрать Windows PowerShell
PS C:\temp\Node> npm install mongoose --save

added 24 packages, and audited 83 packages in 4s

9 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\temp\Node> █
```

## Установка Mongoose

модель для базы данных

```
Windows PowerShell
PS C:\temp\Node> npm install hbs --save

added 9 packages, and audited 92 packages in 1s

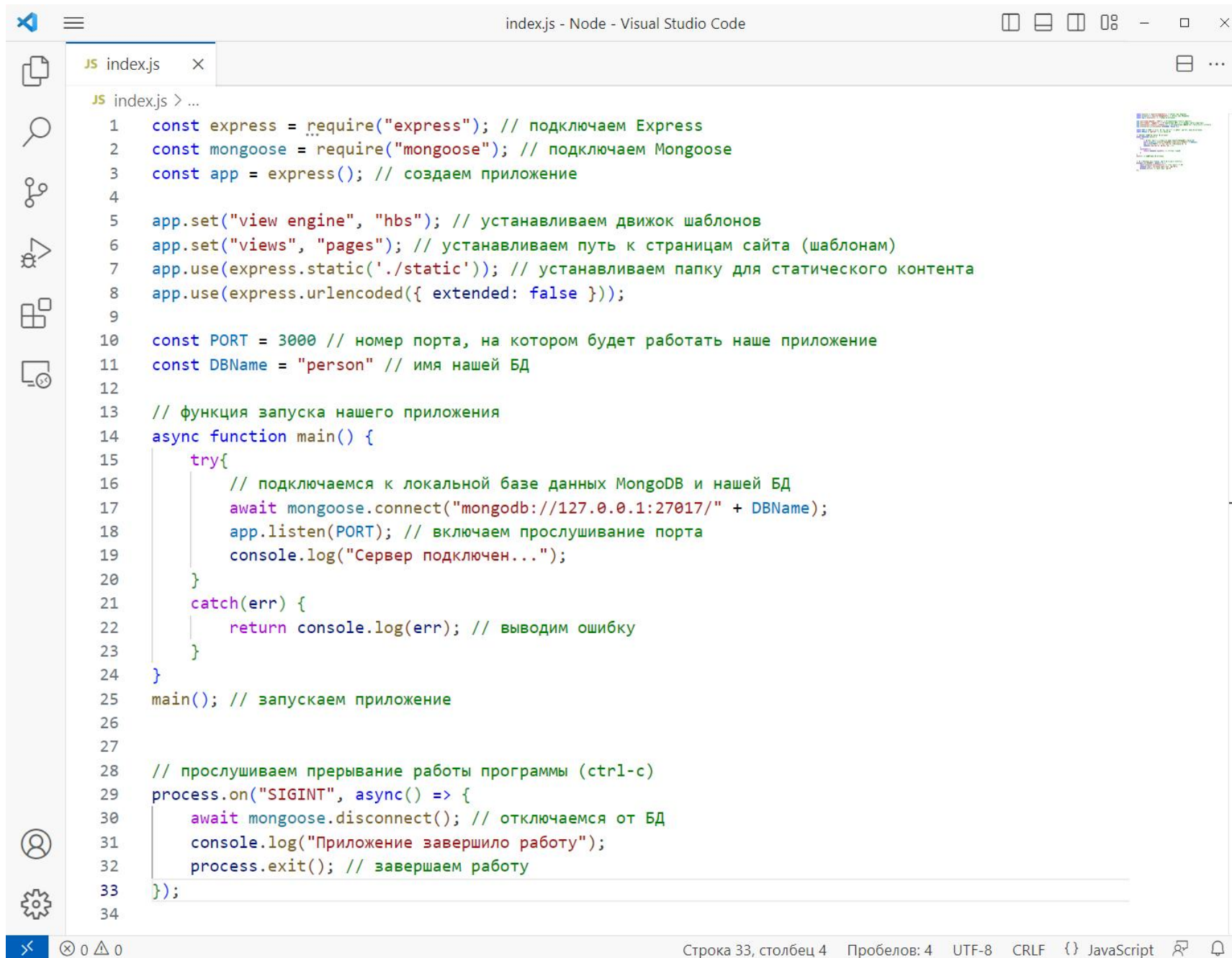
10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\temp\Node> █
```

## Установка Handelbars

шаблонизатор

# Пишем сервер



```
index.js - Node - Visual Studio Code

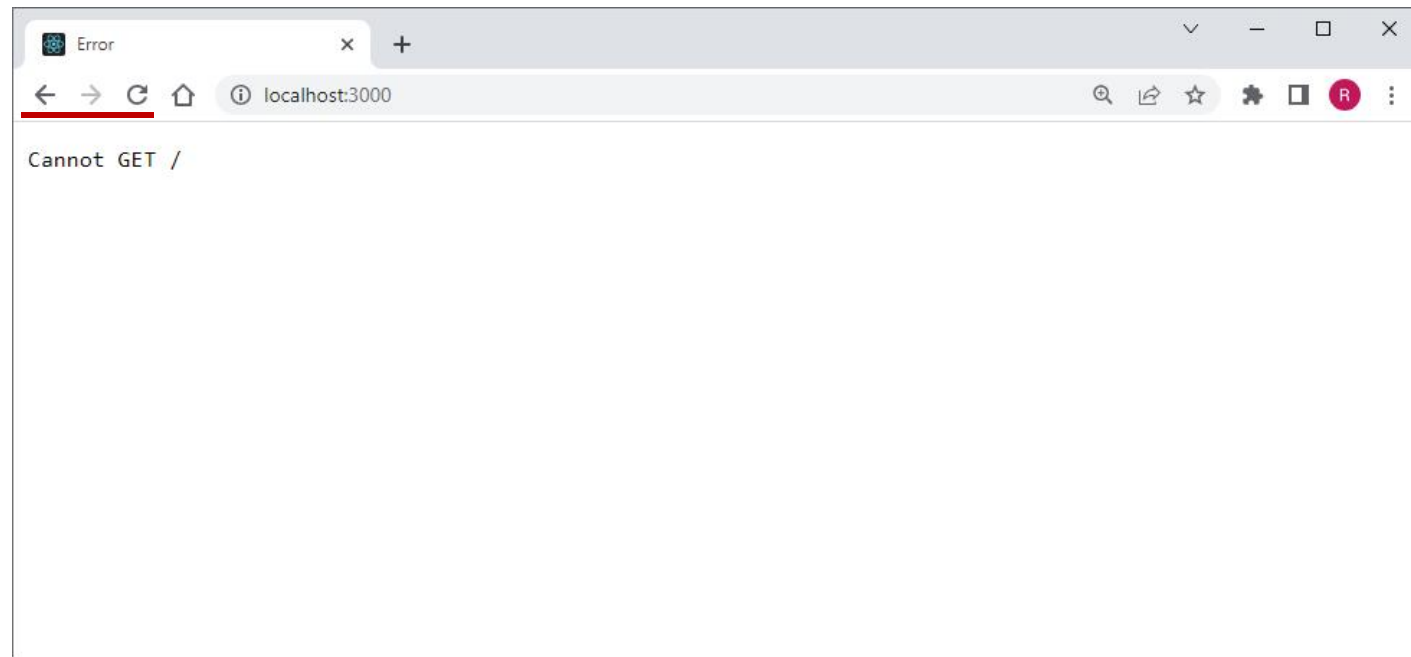
JS index.js x
JS index.js > ...
1  const express = require("express"); // подключаем Express
2  const mongoose = require("mongoose"); // подключаем Mongoose
3  const app = express(); // создаем приложение
4
5  app.set("view engine", "hbs"); // устанавливаем движок шаблонов
6  app.set("views", "pages"); // устанавливаем путь к страницам сайта (шаблонам)
7  app.use(express.static('./static')); // устанавливаем папку для статического контента
8  app.use(express.urlencoded({ extended: false }));
9
10 const PORT = 3000 // номер порта, на котором будет работать наше приложение
11 const DBName = "person" // имя нашей БД
12
13 // функция запуска нашего приложения
14 async function main() {
15     try{
16         // подключаемся к локальной базе данных MongoDB и нашей БД
17         await mongoose.connect("mongodb://127.0.0.1:27017/" + DBName);
18         app.listen(PORT); // включаем прослушивание порта
19         console.log("Сервер подключен...");
20     }
21     catch(err) {
22         return console.log(err); // выводим ошибку
23     }
24 }
25 main(); // запускаем приложение
26
27
28 // прослушиваем прерывание работы программы (ctrl-c)
29 process.on("SIGINT", async() => {
30     await mongoose.disconnect(); // отключаемся от БД
31     console.log("Приложение завершило работу");
32     process.exit(); // завершаем работу
33 });
34
```

Строка 33, столбец 4 Пробелов: 4 UTF-8 CRLF {} JavaScript

# Запускаем сервер

```
Выбрать Windows PowerShell
PS C:\temp\Node> node index.js
Сервер подключен...
```

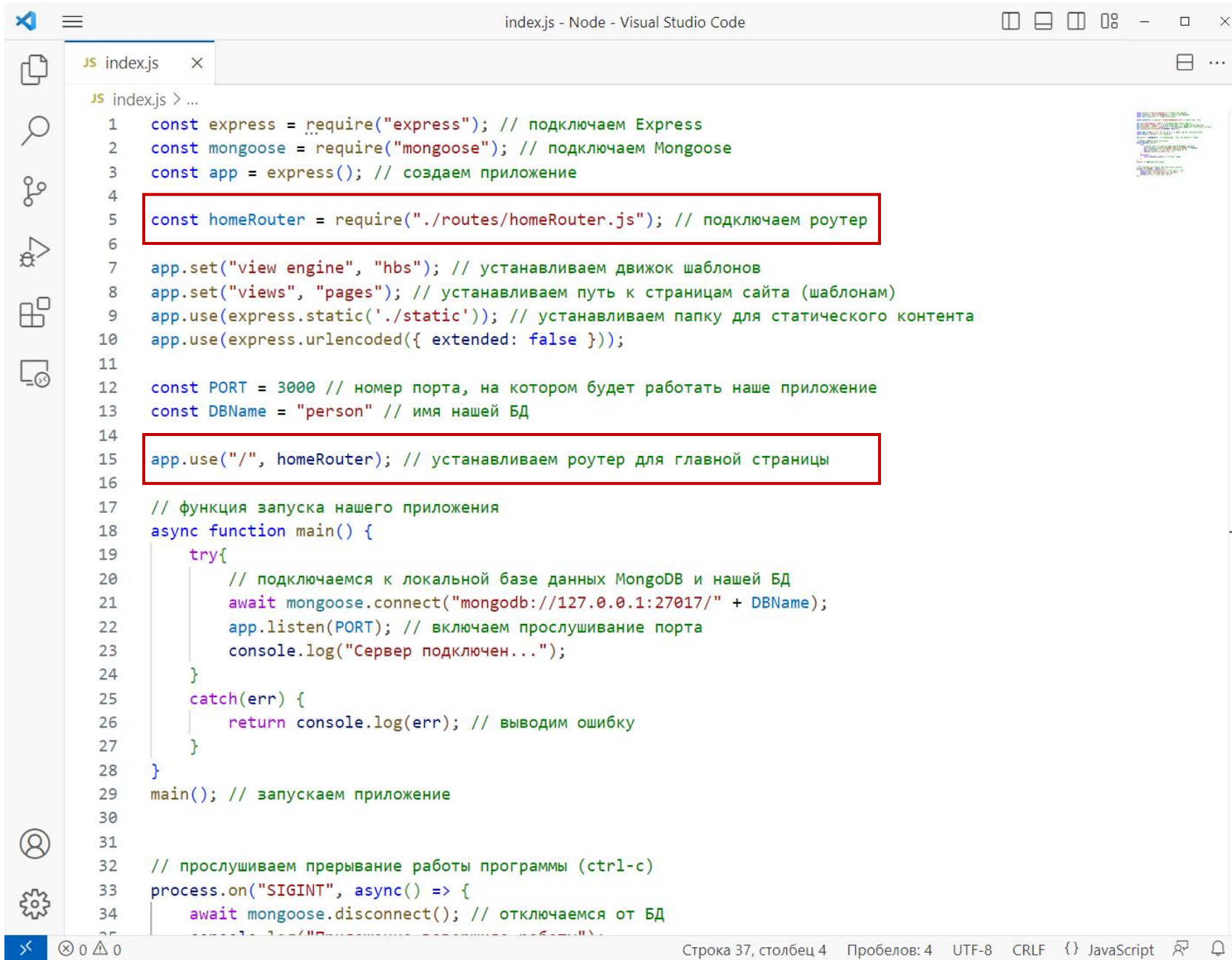
Работа сервера!



```
Выбрать Windows PowerShell
PS C:\temp\Node> node index.js
Сервер подключен...
Приложение завершило работу
PS C:\temp\Node> █
```



# Добавляем роутер



```
index.js - Node - Visual Studio Code

JS index.js x
JS index.js > ...
1  const express = require("express"); // подключаем Express
2  const mongoose = require("mongoose"); // подключаем Mongoose
3  const app = express(); // создаем приложение
4
5  const homeRouter = require("./routes/homeRouter.js"); // подключаем роутер
6
7  app.set("view engine", "hbs"); // устанавливаем движок шаблонов
8  app.set("views", "pages"); // устанавливаем путь к страницам сайта (шаблонам)
9  app.use(express.static('./static')); // устанавливаем папку для статического контента
10 app.use(express.urlencoded({ extended: false }));
11
12 const PORT = 3000 // номер порта, на котором будет работать наше приложение
13 const DBName = "person" // имя нашей БД
14
15 app.use("/", homeRouter); // устанавливаем роутер для главной страницы
16
17 // функция запуска нашего приложения
18 async function main() {
19     try{
20         // подключаемся к локальной базе данных MongoDB и нашей БД
21         await mongoose.connect("mongodb://127.0.0.1:27017/" + DBName);
22         app.listen(PORT); // включаем прослушивание порта
23         console.log("Сервер подключен...");
24     }
25     catch(err) {
26         return console.log(err); // выводим ошибку
27     }
28 }
29 main(); // запускаем приложение
30
31
32 // прослушиваем прерывание работы программы (ctrl-c)
33 process.on("SIGINT", async() => {
34     await mongoose.disconnect(); // отключаемся от БД
35     console.log("Приложение прерывание работы");
36 })
37
```

Строка 37, столбец 4 Пробелов: 4 UTF-8 CRLF {} JavaScript



# Добавляем роутер

Файл Правка Выделение Вид Переход ... homeRouter.js - Node - Visual Studio Code

ПРОВОДНИК ...

- ▼ NODE
  - > node\_modules
  - ▼ routes
    - JS homeRouter.js**
    - JS index.js
    - { } package-lock.json
    - { } package.json

routes > JS homeRouter.js > ...

```
1 const express = require("express");
2 const homeController = require("../controllers/home.js"); // подключаем контроллер
3 const homeRouter = express.Router(); // создаем роутер
4
5 homeRouter.use("/about", homeController.about); // используем контроллер для "/about/"
6 homeRouter.use("/", homeController.index); // используем контроллер для "/"
7
8 module.exports = homeRouter; // экспортируем роутер
```

Строка 8, столбец 52 Пробелов: 4 UTF-8 CRLF {} JavaScript

# Добавляем контроллер

The screenshot shows the Visual Studio Code interface with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with 'controllers' and 'home.js' highlighted. The code editor shows the content of 'home.js' with two functions: 'index' and 'about'. Both functions use 'response.render' to render templates. The status bar at the bottom indicates the current position is at line 1, column 1.

Файл Правка Выделение Вид Переход ... home.js - Node - Visual Studio Code

ПРОВОДНИК ...

- ▼ NODE
  - ▼ controllers
    - JS home.js**
  - > node\_modules
  - ▼ routes
    - JS homeRouter.js
    - JS index.js
  - { } package-lock.json
  - { } package.json

controllers > JS home.js > ...

```
1  
2 exports.index = function (request, response) {  
3   response.render("index.hbs"); // ссылаемся на файл-шаблон  
4 };  
5  
6 exports.about = function (request, response) {  
7   response.render("about.hbs"); // ссылаемся на файл-шаблон  
8 };
```

Строка 1, столбец 1 Пробелов: 4 UTF-8 CRLF {} JavaScript

# Добавляем шаблоны

The screenshot shows the Visual Studio Code editor with the file explorer on the left. The file explorer shows the project structure with folders like controllers, node\_modules, pages, routes, and static. The pages folder is expanded, showing about.hbs and index.hbs. The static folder is also expanded, showing styles.css. The index.hbs file is selected and its content is displayed in the editor. The content is an HTML template with a head section containing meta, link, and title tags, and a body section with a main content area and a footer area.

```
index.hbs - Node - Visual Studio Code
pages > index.hbs > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <link href="../styles.css" rel="stylesheet" type="text/css">
6   <title>Document</title>
7 </head>
8 <body>
9 <hr>
10 <a href="/">Главная</a>|<a href="/about/">О приложении</a>|<a href="/items/">Список вещей</a>
11 <hr>
12
13 <div>Главная страница веб-приложения</div>
14
15 </body>
16 </html>
```

Строка 16, столбец 8 Пробелов: 4 UTF-8 CRLF Handlebars

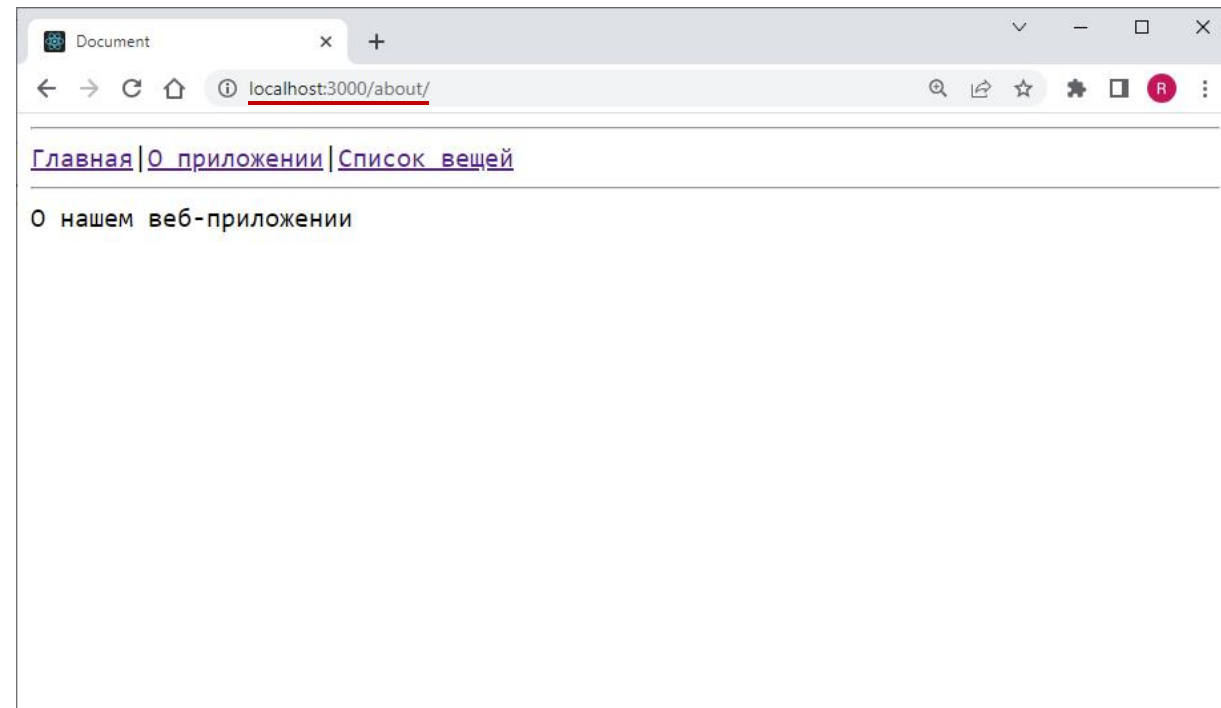
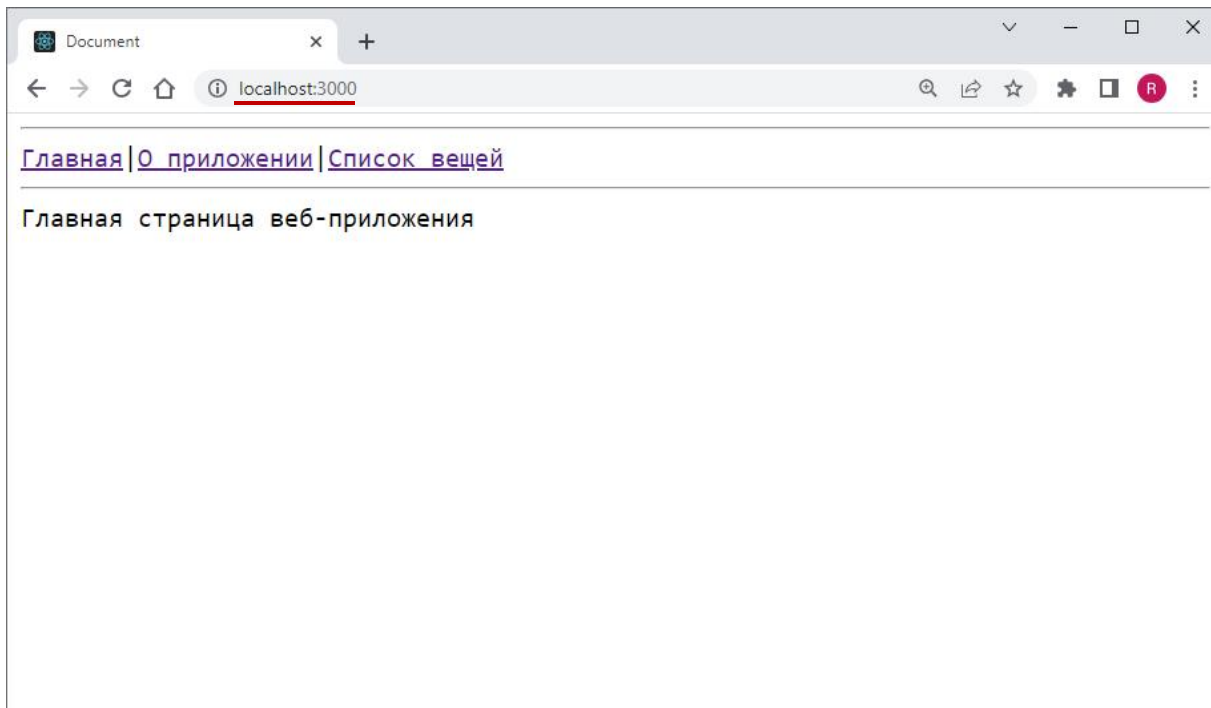
The screenshot shows the Visual Studio Code editor with the file explorer on the left. The file explorer shows the project structure with folders like controllers, node\_modules, pages, routes, and static. The pages folder is expanded, showing about.hbs and index.hbs. The static folder is also expanded, showing styles.css. The about.hbs file is selected and its content is displayed in the editor. The content is an HTML template with a head section containing meta, link, and title tags, and a body section with a main content area and a footer area.

```
about.hbs - Node - Visual Studio Code
pages > about.hbs > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <link href="../styles.css" rel="stylesheet" type="text/css">
6   <title>Document</title>
7 </head>
8 <body>
9 <hr>
10 <a href="/">Главная</a>|<a href="/about/">О приложении</a>|<a href="/items/">Список вещей</a>
11 <hr>
12
13 <div>О нашем веб-приложении</div>
14
15 </body>
16 </html>
```

Строка 16, столбец 8 Пробелов: 4 UTF-8 CRLF Handlebars

# Запускаем...

```
Windows PowerShell
Сервер подключен...
Приложение завершило работу
PS C:\temp\Node> node index.js
Сервер подключен...
```



# Добавляем роутер для списка

index.js - Node - Visual Studio Code

Файл Правка Выделение Вид Переход ...

index.js

```
1  const express = require("express"); // подключаем Express
2  const mongoose = require("mongoose"); // подключаем Mongoose
3  const app = express(); // создаем приложение
4
5  const itemRouter = require("./routes/itemRouter.js"); // подключаем роутер
6  const homeRouter = require("./routes/homeRouter.js"); // подключаем роутер
7
8  app.set("view engine", "hbs"); // устанавливаем движок шаблонов
9  app.set("views", "pages"); // устанавливаем путь к страницам сайта (шаблонам)
10 app.use(express.static('./static')); // устанавливаем папку для статического контента
11 app.use(express.urlencoded({ extended: false }));
12
13 const PORT = 3000 // номер порта, на котором будет работать наше приложение
14 const DBName = "person" // имя нашей БД
15
16 app.use("/items", itemRouter); // устанавливаем роутер работы с данными
17 app.use("/", homeRouter); // устанавливаем роутер для главной страницы
18
19 // функция запуска нашего приложения
20 async function main() {
21   try{
22     // подключаемся к локальной базе данных MongoDB и нашей БД
23     await mongoose.connect("mongodb://127.0.0.1:27017/" + DBName);
24     app.listen(PORT); // включаем прослушивание порта
25     console.log("Сервер подключен...");
26   }
27   catch(err) {
28     return console.log(err); // выводим ошибку
29   }
30 }
31 main(); // запускаем приложение
32
33
34 // прослушиваем прерывание работы программы (ctrl-c)
35 process.on("SIGINT", async () => {
```

Строка 40, столбец 1 Пробелов: 4 UTF-8 CRLF {} JavaScript

# Добавляем роутер для списка

Файл Правка Выделение Вид Переход ... itemRouter.js - Node - Visual Studio Code

PROBODNIK ... JS index.js JS itemRouter.js X

▼ NODE

- > controllers
- > node\_modules
- > pages
- ▼ routes
  - JS homeRouter.js
  - JS itemRouter.js
  - > static
- JS index.js
- { } package-lock.json
- { } package.json

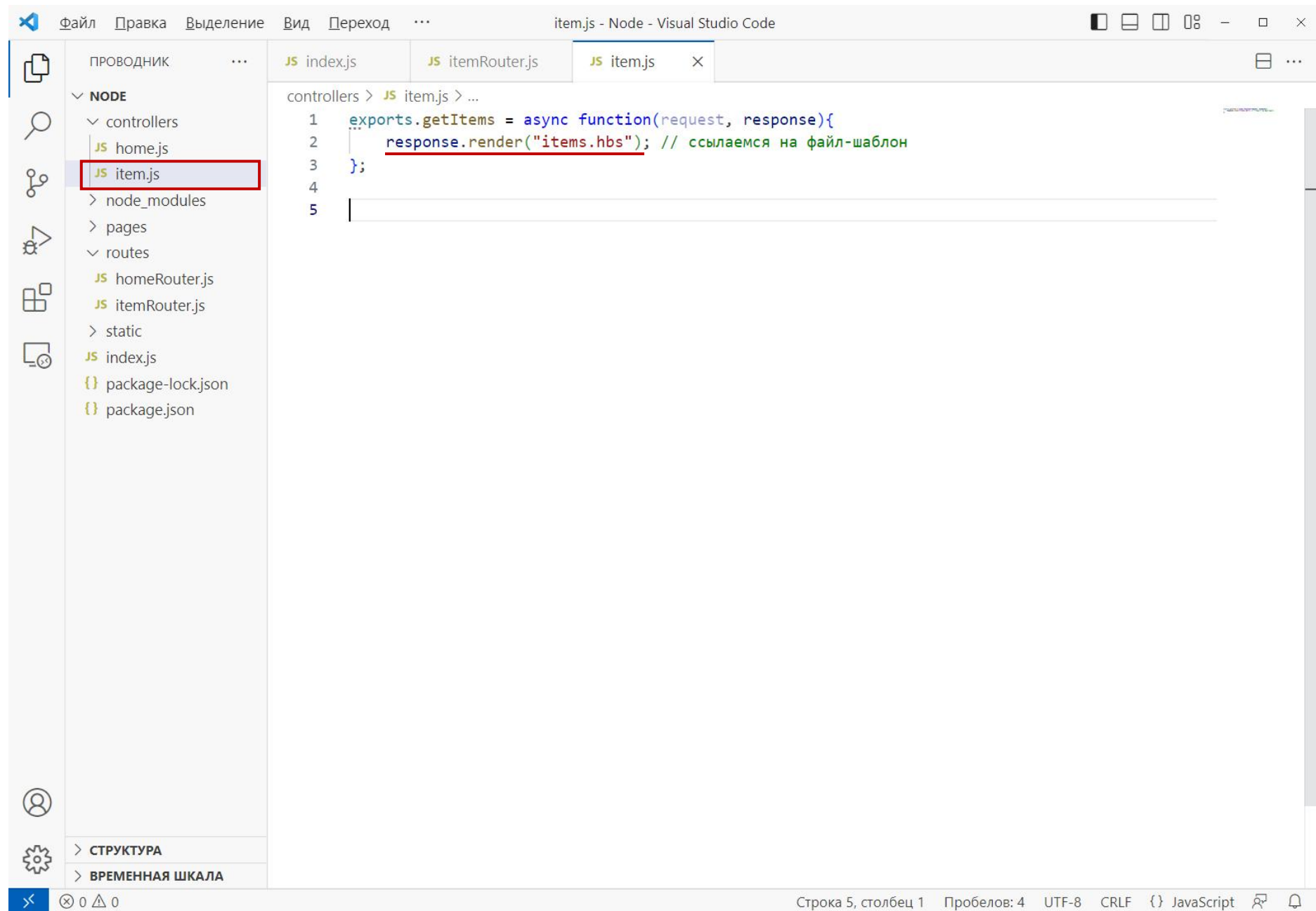
routes > JS itemRouter.js > ...

```
1 const express = require("express");
2
3 const itemController = require("../controllers/item.js"); // подключаем контроллер
4 const itemRouter = express.Router(); // создаем роутер
5
6 itemRouter.use("/", itemController.getItems); // используем контроллер для "/"
7
8 module.exports = itemRouter; // экспортируем роутер
```

Строка 8, столбец 52 Пробелов: 4 UTF-8 CRLF {} JavaScript



# Добавляем контроллер для списка





# Добавляем шаблон для списка

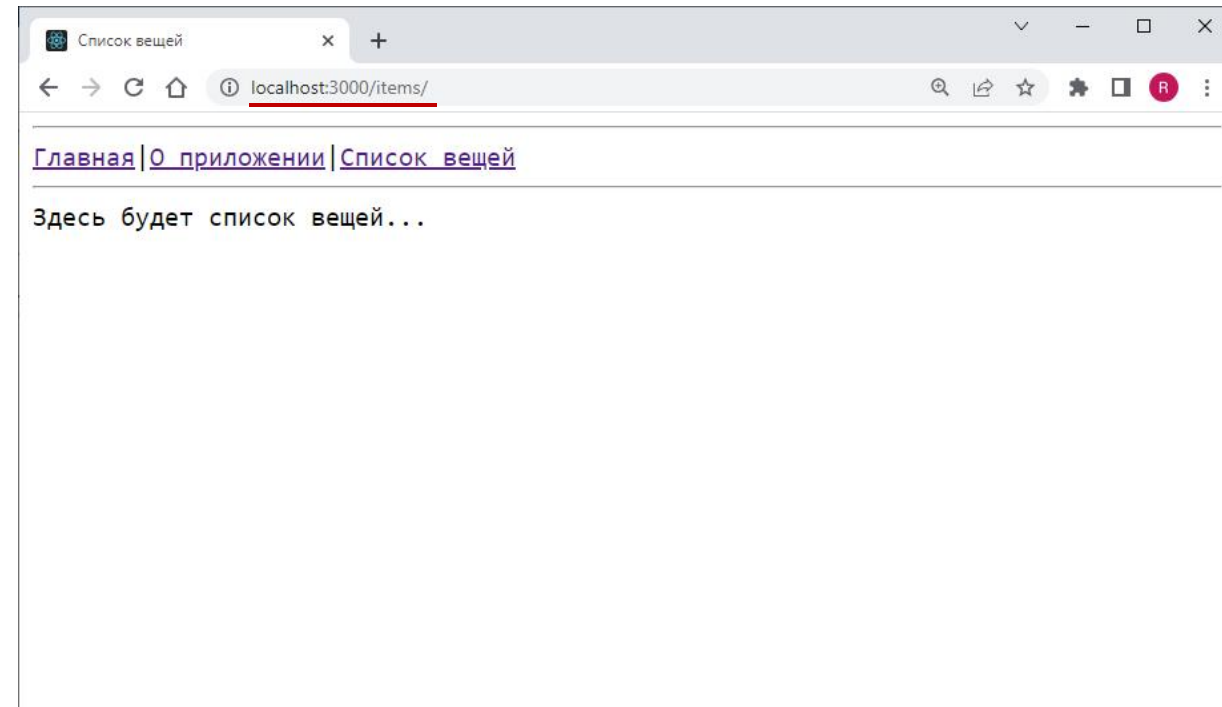
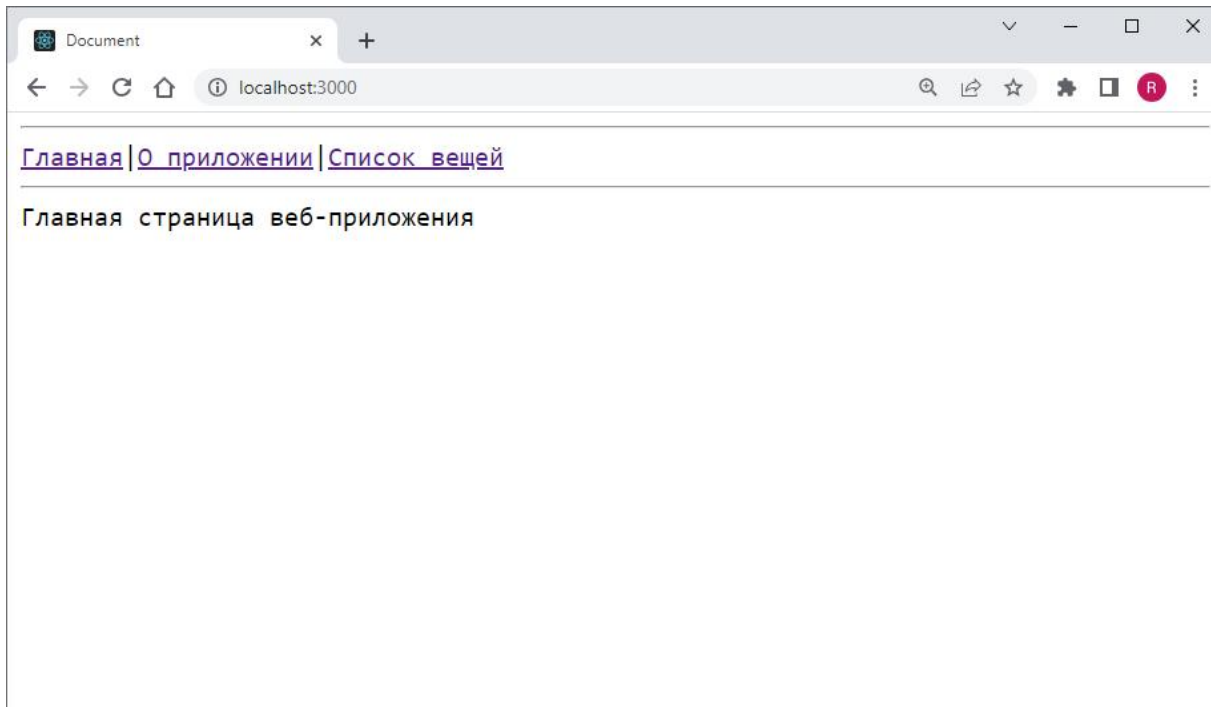
The screenshot shows the Visual Studio Code interface with the following components:

- Top Bar:** Contains the menu bar (Файл, Правка, Выделение, Вид, Переход, Выполнить) and the title bar (items.hbs - Node - Visual Studio Code).
- Left Panel:** Displays the Explorer view with the file tree. The 'pages' folder is expanded, and 'items.hbs' is highlighted with a red rectangle. Other files in the tree include 'index.hbs', 'about.hbs', 'homeRouter.js', 'itemRouter.js', 'index.js', 'package-lock.json', and 'package.json'.
- Editor:** Shows the content of 'items.hbs' with the following HTML template code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Список вещей</title>
5   <meta charset="utf-8" />
6   <link href=" ../styles.css" rel="stylesheet" type="text/css">
7 </head>
8 <body>
9 <hr>
10 <a href="/">Главная</a>|<a href="/about/">О приложении</a>|<a href="/items/">Список вещей</a>
11 <hr>
12
13 <div>Здесь будет список вещей...</div>
14
15 </body>
16 </html>
```
- Bottom Panel:** Displays the status bar with the text 'Строка 1, столбец 1 Пробелов: 4 UTF-8 CRLF Handlebars'.

# Запускаем...

```
Windows PowerShell
Сервер подключен...
Приложение завершило работу
PS C:\temp\Node> node index.js
Сервер подключен...
```



# Добавляем функционал. Роутер

The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor in the center. The file explorer shows a project structure with folders like 'controllers', 'pages', 'routes', and 'static'. The 'routes' folder is expanded, showing 'homeRouter.js' and 'itemRouter.js'. The 'itemRouter.js' file is selected and its content is displayed in the code editor. The code defines an Express.js router with two routes: '/add' and '/'. The status bar at the bottom indicates the current line and column (9, 52), the number of spaces (4), the encoding (UTF-8), the line endings (CRLF), and the language (JavaScript).

Файл Правка Выделение Вид Переход ... itemRouter.js - Node - Visual Studio Code

PROBODNIK ...

NODE

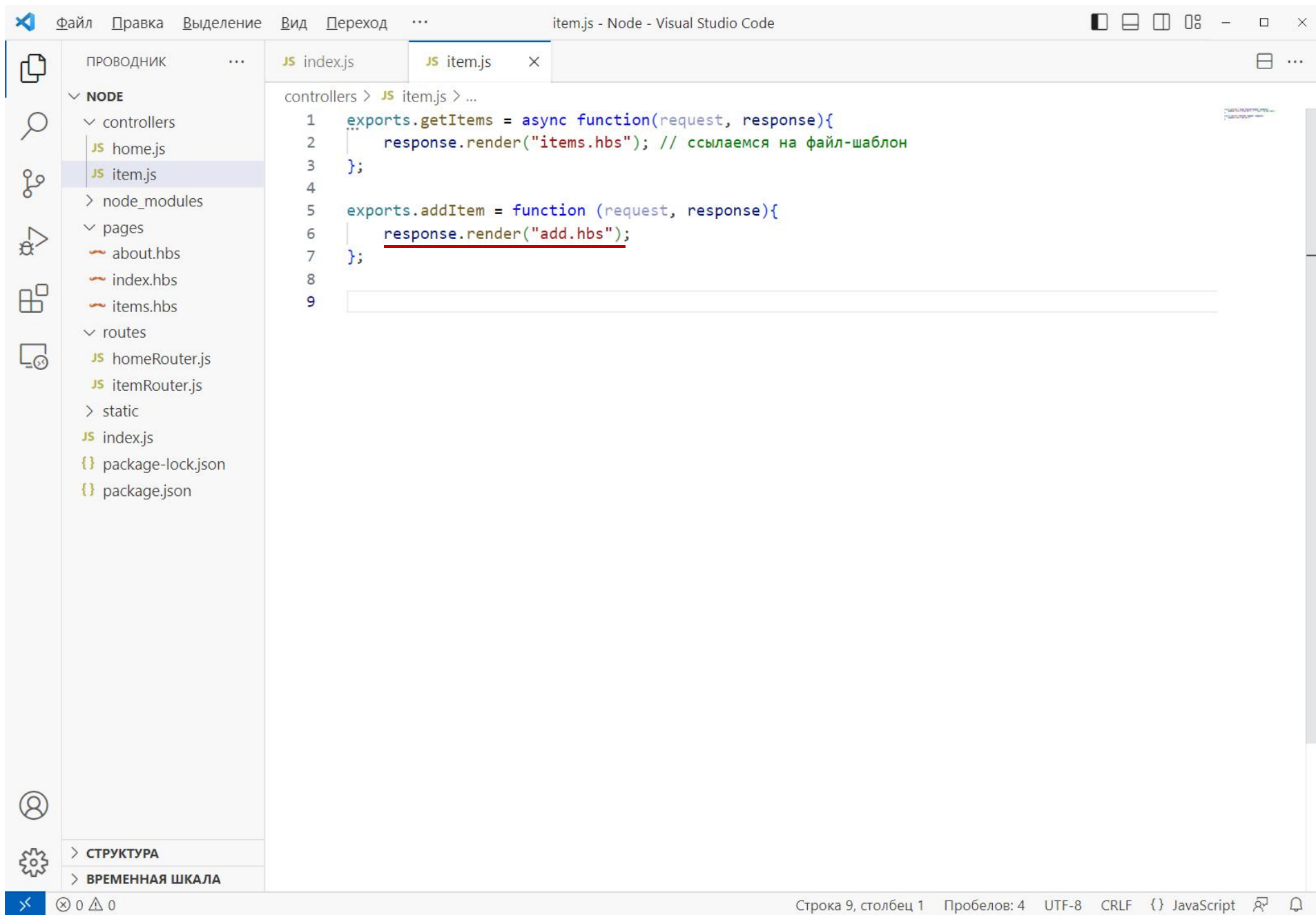
- controllers
  - home.js
  - item.js
- node\_modules
- pages
  - about.hbs
  - index.hbs
  - items.hbs
- routes
  - homeRouter.js
  - itemRouter.js
- static
- index.js
- package-lock.json
- package.json

routes > JS itemRouter.js > ...

```
1 const express = require("express");
2
3 const itemController = require("../controllers/item.js"); // подключаем контроллер
4 const itemRouter = express.Router(); // создаем роутер
5
6 itemRouter.use("/add", itemController.addItem); // используем контроллер для "/add"
7 itemRouter.use("/", itemController.getItems); // используем контроллер для "/"
8
9 module.exports = itemRouter; // экспортируем роутер
```

Строка 9, столбец 52 Пробелов: 4 UTF-8 CRLF {} JavaScript

# Добавляем функционал. Контроллер



Visual Studio Code interface showing the file explorer on the left and the code editor on the right. The file explorer shows the project structure, including the 'controllers' directory. The code editor displays the contents of 'item.js'.

File Explorer (Left):

- PROВОДНИК
- NODE
  - controllers
    - home.js
    - item.js (selected)
  - node\_modules
  - pages
    - about.hbs
    - index.hbs
    - items.hbs
  - routes
    - homeRouter.js
    - itemRouter.js
  - static
  - index.js
  - package-lock.json
  - package.json
- СТРУКТУРА
- ВРЕМЕННАЯ ШКАЛА

Code Editor (Right):

```
controllers > JS item.js > ...
1  exports.getItems = async function(request, response){
2    response.render("items.hbs"); // ссылаемся на файл-шаблон
3  };
4
5  exports.addItem = function (request, response){
6    response.render("add.hbs");
7  };
8
9
```

Status Bar (Bottom):

Строка 9, столбец 1 Пробелов: 4 UTF-8 CRLF {} JavaScript

# Добавляем функционал. Шаблон

The screenshot shows the Visual Studio Code editor interface. The top menu bar includes 'Файл', 'Правка', 'Выделение', 'Вид', 'Переход', 'Выполнить', and a dropdown menu. The title bar indicates the active file is 'items.hbs - Node - Visual Studio Code'. The left sidebar shows the 'PROVODNIK' (Explorer) view with a tree structure of files and folders. The 'pages' folder is expanded, showing 'about.hbs', 'index.hbs', and 'items.hbs'. The 'items.hbs' file is selected. The main editor area displays the content of 'items.hbs', which is an HTML template. The template includes a DOCTYPE declaration, a head section with a title 'Список вещей', a meta charset declaration, and a link to 'styles.css'. The body section contains a list of links: 'Главная', '0 приложений', and 'Список вещей'. The status bar at the bottom shows 'Строка 16, столбец 7', 'Пробелов: 4', 'UTF-8', 'CRLF', and 'Handlebars'.

Файл Правка Выделение Вид Переход Выполнить ... items.hbs - Node - Visual Studio Code

PROVODNIK ...

NODE

- controllers
  - home.js
  - item.js
- node\_modules
- pages
  - about.hbs
  - index.hbs
  - items.hbs
- routes
  - homeRouter.js
  - itemRouter.js
- static
- index.js
- package-lock.json
- package.json

items.hbs

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Список вещей</title>
5   <meta charset="utf-8" />
6   <link href="../styles.css" rel="stylesheet" type="text/css">
7 </head>
8 <body>
9 <hr>
10 <a href="/">Главная</a>|<a href="/about/">0 приложений</a>|<a href="/items/">Список вещей</a>
11 <hr>
12
13 <a href="/items/add">Добавить вещь</a>
14
15 </body>
16 </html>
```

Строка 16, столбец 7 Пробелов: 4 UTF-8 CRLF Handlebars

# Добавляем функционал. Шаблон

add.hbs - Node - Visual Studio Code

Файл Правка Выделение Вид Переход Выполнить ...

PROBODNIK ... JS index.js items.hbs add.hbs

NODE

- controllers
- JS home.js
- JS item.js
- node\_modules
- pages
  - about.hbs
  - add.hbs**
  - index.hbs
  - items.hbs
- routes
  - JS homeRouter.js
  - JS itemRouter.js
- static
- JS index.js
- package-lock.json
- package.json

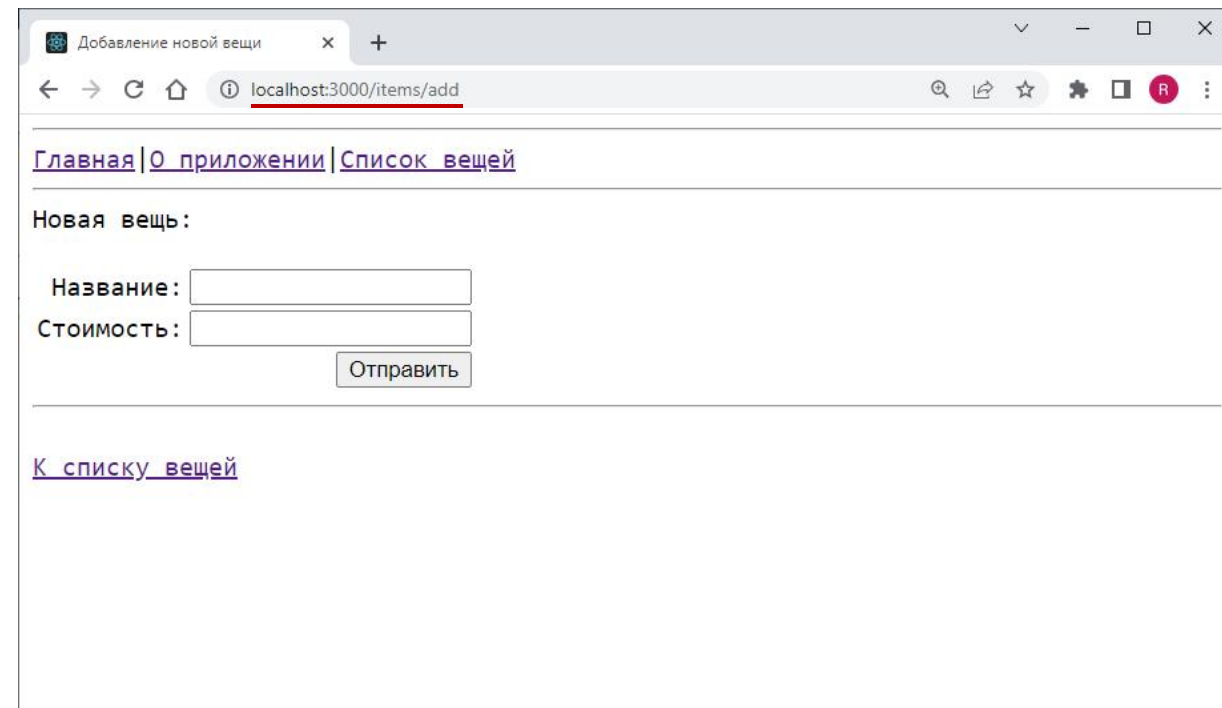
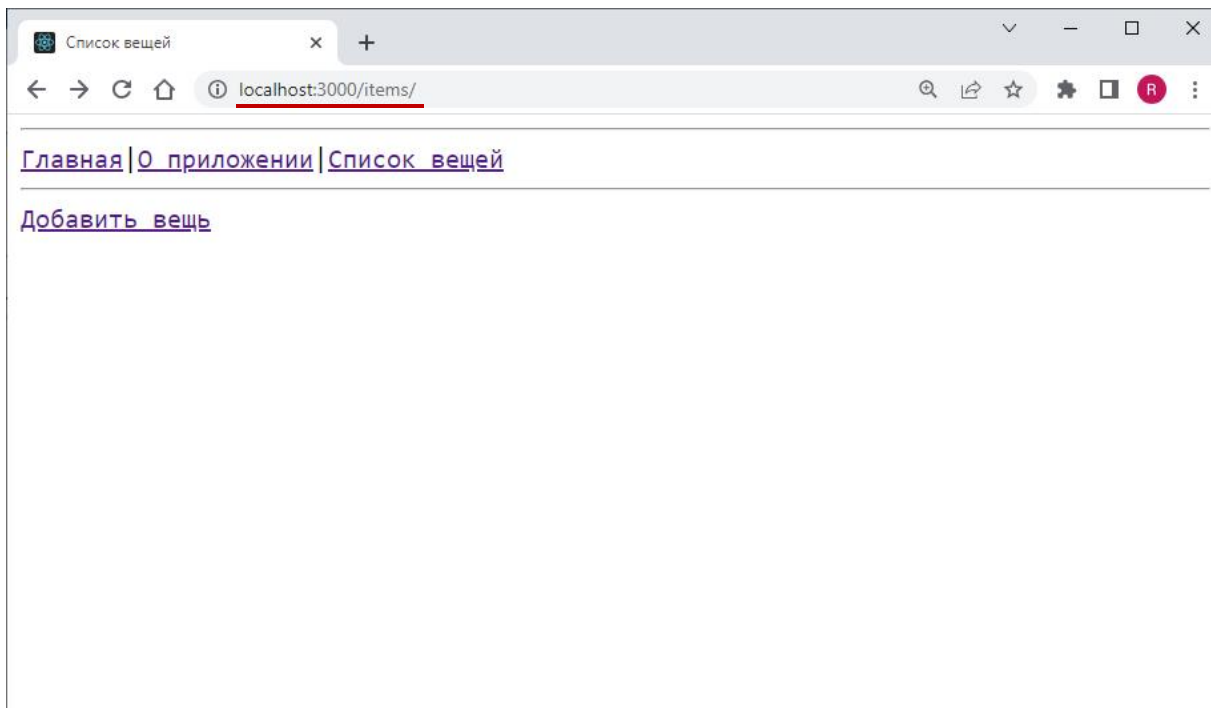
pages > add.hbs > html > body > form > table > tr > td

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Добавление новой вещи</title>
5   <meta charset="utf-8" />
6   <link href=" ../styles.css" rel="stylesheet" type="text/css">
7 </head>
8 <body>
9 <hr>
10 <a href="/">Главная</a>|<a href="/about/">О приложении</a>|<a href="/items/">Список вещей</a>
11 <hr>
12 <div>Новая вещь:</div>
13 <br>
14   <form action="postItem" method="POST">
15     <table>
16       <tr>
17         <td><div class="label">Название:</div></td>
18         <td><input name="name" /></td>
19       </tr>
20       <tr>
21         <td><div class="label">Стоимость:</div></td>
22         <td><input name="cost" type="number" min="1" /></td>
23       </tr>
24       <tr>
25         <td></td>
26         <td><div class="label"><input type="submit" value="Отправить" /></div></td>
27       </tr>
28     </table>
29   </form>
30   <hr>
31   <br>
32   <a href="/items">К списку вещей</a>
33 </body>
34 <html>
```

Строка 26, столбец 17 Пробелов: 4 UTF-8 CRLF Handlebars

# Запускаем...

```
Windows PowerShell
Сервер подключен...
Приложение завершило работу
PS C:\temp\Node> node index.js
Сервер подключен...
```





# Реализуем добавление в базу данных. Роутер

The screenshot shows the Visual Studio Code interface with the following components:

- Explorer (Left):** Displays the file structure of a project. The 'routes' folder is expanded, showing 'homeRouter.js' and 'itemRouter.js' (which is selected).
- Editor (Center):** Displays the code for 'itemRouter.js'. The code is as follows:

```
1 const express = require("express");
2
3 const itemController = require("../controllers/item.js"); // подключаем контроллер
4 const itemRouter = express.Router(); // создаем роутер
5
6 itemRouter.use("/postitem", itemController.postItem); // используем контроллер для "/postitem"
7 itemRouter.use("/add", itemController.addItem); // используем контроллер для "/add"
8 itemRouter.use("/", itemController.getItems); // используем контроллер для "/"
9
10 module.exports = itemRouter; // экспортируем роутер
```
- Bottom Status Bar:** Shows the current cursor position as 'Строка 10, столбец 52' (Line 10, Column 52), the number of spaces as 'Пробелов: 4', the encoding as 'UTF-8', the line endings as 'CRLF', and the language as 'JavaScript'.

# Реализуем добавление в базу данных. Модель

The screenshot shows the Visual Studio Code interface with the following components:

- Explorer (Left):** Displays the project structure. The `models` folder is expanded, and `item.js` is selected and highlighted with a red rectangle.
- Editor (Center):** Shows the code in `models > JS item.js > ...`. The code implements a Mongoose model for an 'Item'.
- Code:**

```
1  const mongoose = require("mongoose"); // подключаем модуль "mongoose"
2
3  const Schema = mongoose.Schema;
4  // установка схемы
5  const itemScheme = new Schema({
6    name: String,
7    cost: Number
8  });
9
10 module.exports = mongoose.model("Item", itemScheme); // экспортируем схему
```
- Bottom Bar:** Displays status information: "Строка 1, столбец 1", "Пробелов: 4", "UTF-8", "CRLF", and "JavaScript".

# Реализуем добавление в базу данных. Контроллер

The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor in the center. The file explorer shows a project structure with a 'controllers' folder containing 'item.js'. The code editor displays the content of 'item.js', which implements a controller for adding items to a database. The code includes imports for 'Item' and 'require', and defines three export functions: 'addItem', 'getItems', and 'postItem'. The 'postItem' function handles the logic for adding a new item, including validation, object creation, saving to the database, and redirecting to the items list.

```
1  const Item = require("../models/item.js");
2
3  exports.addItem = function (request, response){
4    response.render("add.hbs");
5  };
6
7  exports.getItems = async function(request, response){
8    response.render("items.hbs"); // ссылаемся на файл-шаблон
9  };
10
11 exports.postItem= async function(request, response){
12   if(!request.body) return response.sendStatus(400); // проверяем, что запрос не пустой
13
14   const itemName = request.body.name; // копируем значения из формы
15   const itemCost = request.body.cost;
16
17   const item = new Item({name: itemName, cost: itemCost}); // создаем объект согласно схеме
18
19   await item.save(); // сохраняем в базе данных
20   response.redirect("/items"); // переходим на список вещей
21 };
```

Строка 21, столбец 3 Пробелов: 4 UTF-8 CRLF {} JavaScript

# Запускаем...

```
Windows PowerShell
Сервер подключен...
Приложение завершило работу
PS C:\temp\Node> node index.js
Сервер подключен...
```

Добавление новой вещи

localhost:3000/items/add

[Главная](#) | [О приложении](#) | [Список вещей](#)

Новая вещь:

Название:

Стоимость:

[К списку вещей](#)

# Запускаем...

```
Windows PowerShell
Сервер подключен...
Приложение завершило работу
PS C:\temp\Node> node index.js
Сервер подключен...
```

Список вещей

localhost:3000/items

Главная | О приложении | Список вещей

Добавить вещь

MongoDB Compass - localhost:27017/person.items

localhost:27017

Documents  
person.items

My Queries

Databases

Search

admin

config

items

local

person

items

todo

usersdb

person.items

1 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find Options

ADD DATA EXPORT DATA

1 - 1 of 1

```
_id: ObjectId('64ad155ea3668eda8c8b19b4')
name: "Шкатулка"
cost: 512
__v: 0
```

>\_MONGOSH

# Реализуем отображение списка данных. Контроллер

Файл Правка Выделение Вид Переход Выполнить ... item.js - Node - Visual Studio Code

PROBODNIK ... JS index.js JS item.js X

NODE

- controllers
  - JS home.js
  - JS item.js
- models
  - JS item.js
- node\_modules
- pages
  - about.hbs
  - add.hbs
  - index.hbs
  - items.hbs
- routes
  - JS homeRouter.js
  - JS itemRouter.js
- static
- JS index.js
- package-lock.json
- package.json

СТРУКТУРА

ВРЕМЕННАЯ ШКАЛА

```
1 const Item = require("../models/item.js");
2
3 exports.addItem = function (request, response){
4     response.render("add.hbs");
5 };
6
7 exports.getItems = async function(request, response){
8     const allItems = await Item.find({}); // получаем все элементы из БД в массив
9     response.render("items.hbs", { items: allItems }); // отправляем в шаблон массив с данными
10 };
11
12 exports.postItem = async function(request, response){
13     if(!request.body) return response.sendStatus(400); // проверяем, что запрос не пустой
14
15     const itemName = request.body.name; // копируем значения из формы
16     const itemCost = request.body.cost;
17
18     const item = new Item({name: itemName, cost: itemCost}); // создаем объект согласно схеме
19
20     await item.save(); // сохраняем в базе данных
21     response.redirect("/items"); // переходим на список вещей
22 };
```

Строка 22, столбец 3 Пробелов: 4 UTF-8 CRLF {} JavaScript



# Реализуем отображение списка данных. Шаблон

The screenshot shows the Visual Studio Code editor with the file `items.hbs` open. The left sidebar displays the project structure, including a `pages` folder containing `items.hbs`. The main editor area shows the following HTML template code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Список вещей</title>
5   <meta charset="utf-8" />
6   <link href=" ../styles.css" rel="stylesheet" type="text/css">
7 </head>
8 <body>
9 <hr>
10 <a href="/">Главная</a>|<a href="/about/">О приложении</a>|<a href="/items/">Список вещей</a>
11 <hr>
12
13 <a href="/items/add">Добавить вещь</a>
14 <br><br>
15
16 <table border="1" width="100%">
17   <tr><th>Название</th><th>Стоимость</th></tr>
18   {{#each items}}
19     <tr><td>{{this.name}}</td><td>{{this.cost}}</td></tr>
20   {{/each}}
21 </table>
22
23 </body>
24 </html>
```

A red rectangular box highlights the table structure in lines 16 through 21, which uses Handlebars' `each` helper to iterate over the `items` array and generate table rows.

At the bottom of the editor, the status bar indicates: `Строка 24, столбец 7 Пробелов: 4 UTF-8 CRLF Handlebars`.



# Запускаем...

```
Windows PowerShell
Сервер подключен...
Приложение завершило работу
PS C:\temp\Node> node index.js
Сервер подключен...
```

Список вещей

localhost:3000/items

[Главная](#) | [О приложении](#) | [Список вещей](#)

[Добавить вещь](#)

Название	Стоимость
Шкатулка	512
Кубик	36
Пирамидка	48

MongoDB Compass - localhost:27017/person.items

localhost:27017

Documents person.items

3 DOCUMENTS 1 INDEXES

person.items

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find Options

ADD DATA EXPORT DATA

1 - 3 of 3

```
{
  "_id": ObjectId('64ad155ea3668eda8c8b19b4'),
  "name": "Шкатулка",
  "cost": 512,
  "__v": 0
}
```

```
{
  "_id": ObjectId('64ad174165b3d9f95fac0efa'),
  "name": "Кубик",
  "cost": 36,
  "__v": 0
}
```

```
{
  "_id": ObjectId('64ad17a465b3d9f95fac0efd'),
  "name": "Пирамидка",
  "cost": 48,
  "__v": 0
}
```

> MONGOSH