Створюємо додаток

express --view=ejs myapp

cd myapp

npm install

npm start

Встановлення сервера

<https://www.mongodb.com/download-center/community>

Графічний клієнт для роботи з базою даних

<https://www.mongodb.com/download-center/compass>

Встановлення модуля:

npm install mongoose --save

Використання .env

Файл використовується для зберігання змінних середовища (параметри налаштування та ін.)

<https://medium.com/the-node-js-collection/making-your-node-js-work-everywhere-with-environment-variables-2da8cdf6e786>

<https://www.twilio.com/blog/2017/08/working-with-environment-variables-in-node-js.html>

|  |  |
| --- | --- |
|  |  |
| Створюємо файл .env у кореневому каталозі |  |
| Додаємо правило ігнорування у.gitignore |  |
| Встановлюємо модуль dotenv | npm install dotenv --save |
| Зчитування значень з файлу  const dotenv = require('dotenv'); dotenv.config(); process.env. **змінна** 🡨Зчитуємо | const dotenv = require('dotenv'); 🡨 піключаємо модуль dotenv.config(); 🡨 ініціалізуємо значення змінних console.log(`Your port is ${process.env.**PORT**}`); 🡨Зчитуємо значення змінної середовища  ======================================================== використовуємо тільки при розробці ========  if (process.env.NODE\_ENV !== 'production') {  require('dotenv').config();  }  console.log(`Your port is ${process.env.PORT}`);  console.log(`Your port is ${process.env.DATABASE\_NAME}`); |

Як правило для спрощення роботи з змінними середовища використовують файл конфігурації config.js

|  |  |
| --- | --- |
| Створюємо файл конфігурації | ---------- config.js -----------------  require('dotenv').config();  module.exports = {  databaseName: process.env.DATABASE\_NAME,  databseUrl: process.env.MONGODB\_URL,  mongoURI: `${ process.env.MONGODB\_URL}${ process.env.DATABASE\_NAME}`,  port: process.env.PORT  }; |
| Підключаємо і використовуємо значення | ------------------------- app.js -------------------------  const {port, databaseName}=require('./config')    console.log(`Your port : ${port}`);  console.log(`Database name: ${databaseName}`); |

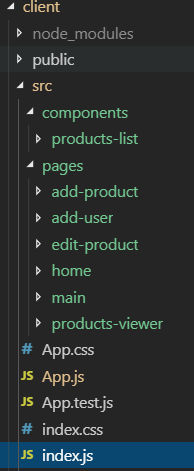
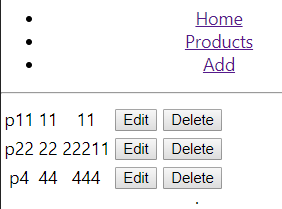
Підключаємо базу даних

|  |  |
| --- | --- |
| Створюємо файл для підключення бази даних (db.js) | ---------------- db.js ----------------  const mongoose=require('mongoose')  mongoose.set('useFindAndModify', false);  const {mongoURI} =require('./config')  mongoose.connect(mongoURI)  mongoose.connection.on('connected',()=>{  console.log('Database connected')  })  mongoose.connection.on('desconnected',()=>{  console.log('Database disconnected')  })  mongoose.connection.on('error',(err)=>{  console.log('Error:',err)  }) |
| Підключаємо до проекту | ----------- app.js --------------  require('./db') |

Приклад розробки API

|  |  |
| --- | --- |
| Створюємо додаткову папку app\_api та додаткові вкладені папки |  |
| Створюємо моделі | const mongoose = require('mongoose');  const productSchema = new mongoose.Schema(  {  title: { type: String, required: true },  price: { type: Number, required: true },  count: { type: Number, required: true },  },  { timestamps: true }  );  module.exports = mongoose.model('Product', productSchema); |
| Створюємо контролери (функції для роботи з базою даних)  (на етапі розробки щоб уникнути проблем з CORS можна використати модуль cors)  <https://www.npmjs.com/package/cors>  Встановити:  npm install cors  Підключити до додатку (в app.js)  var cors **=** require('cors')  var app **=** express()  app.use(cors()) | const mongoose =require('mongoose')  const Product=require('../models/product')  const sendJSONResponse=(res,status,content)=>{  res.setHeader("Access-Control-Allow-Origin", "\*")  res.status(status).json(content);  }  module.exports.getProducts=function(req,res){  const filter= req.body.filter ||{}  Product.find(filter)  .exec((err,products)=>{  if(err){  sendJSONResponse(res,404,  {message:"Products not found"})  return  }  if(products.length==0){  sendJSONResponse(res,404,  {message:"Products not found (empty)"})  return  }  sendJSONResponse(res,200,products)  })  }  module.exports.getProductById=function(req,res){  if(req.params && req.params.productid){  Product  .findById(req.params.productid,(err,product)=>{  if(err){  sendJSONResponse(res,404,  {  message:'Product not found'  });  return  }  sendJSONResponse(res,200,product)  })  }  }  module.exports.addProduct=function(req,res){  if(!req.body ||!req.body.title ||!req.body.price ||!req.body.count){  sendJSONResponse(res,400,{  message:"No data"  })  return  }  const newProduct=new Product({  title:req.body.title,  price:req.body.price,  count:req.body.count  })  newProduct.save((err)=>{  if(err){  sendJSONResponse(res,500,{  message:err  })  return  }  sendJSONResponse(res,200,{  message:'added'  })  })  }  module.exports.updateProduct=function(req,res){  if(req.body.productid ){  // Знаходимо і оновлюємо  Product.findByIdAndUpdate(req.body.productid,  {  title:req.body.title,  price:req.body.price,  count:req.body.count  },  (err,product)=>{  if(err){  sendJSONResponse(res,404,  {  message:'Product not found'  });  return  }  sendJSONResponse(res,200,{  message:'changed'  })  }  )  // Або знаходимо, змінюємо і зберігаємо  // Product  // .findById(req.body.productid,(err,product)=>{ //Знаходимо  // if(err){  // sendJSONResponse(res,404,  // {  // message:'Product not found'  // });  // return  // }  // product.title=req.body.title, //Змінюємо  // product.price=req.body.price,  // product.count=req.body.count,  // product.save((err)=>{ //Зберігаємо  // if(err){  // sendJSONResponse(res,500,  // {  // message:'Not changed'  // });  // return  // }    // sendJSONResponse(res,200,{  // message:'changed'  // })  // })    // })  } else{  sendJSONResponse(res,400,{  message:'bad request'  })  }  }  module.exports.deleteProduct=function(req,res){  if(req.params.productid){  Product.findByIdAndDelete(req.params.productid,(err)=>{  if(err){  sendJSONResponse(res,500,{  message:"can't delete"  })  return  }  sendJSONResponse(res, 204,{  message:"deleted"  })  })  } else  sendJSONResponse(res,400,{  message:'bad request'  })  } |
| Створюємо роути для роботи з API | ------------------- products.js------------------------------  const express=require('express')  const router=express.Router()  const ctrProducts=require('../controllers/products')  const bodyParser = require("body-parser");  const urlencodedParser = bodyParser.urlencoded({extended: false});  router.get('/products',ctrProducts.getProducts);  router.get('/products/:productid',ctrProducts.getProductById);  router.post('/products',  urlencodedParser,  ctrProducts.addProduct);  router.put('/products',  urlencodedParser,  ctrProducts.updateProduct);  router.delete('/products/:productid',ctrProducts.deleteProduct);  module.exports=router    ------------------- index.js------------------------------  const products =require('./products')  module.exports=products |
| Підключення API роутів (вставляємо код у app.js) | --------------- app.js -----------  const apiRouter=require('./app\_api/routes')  app.use('/api', apiRouter); |
| App.js | var createError = require('http-errors');  var express = require('express');  var path = require('path');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  require('./db')  var indexRouter = require('./routes/index');  var usersRouter = require('./routes/users');  const {port, databaseName}=require('./config')  const cors = require('cors')  console.log(`Your port : ${port}`);  console.log(`Database name: ${databaseName}`);  var app = express();  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(cors())  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  const apiRouter=require('./app\_api/routes/products')  app.use('/api', apiRouter);  app.use('/', indexRouter);  app.use('/users', usersRouter);  // catch 404 and forward to error handler  app.use(function(req, res, next) {  next(createError(404));  });  // error handler  app.use(function(err, req, res, next) {  // set locals, only providing error in development  res.locals.message = err.message;  res.locals.error = req.app.get('env') === 'development' ? err : {};  // render the error page  res.status(err.status || 500);  res.render('error');  });  module.exports = app; |

Приклад простого фронтенда

|  |  |
| --- | --- |
|  | import React from 'react';  import Main from './pages/main'  import {BrowserRouter} from 'react-router-dom'  import './App.css';  function App() {  return (  <div className="App">  <BrowserRouter>  <Main />  </BrowserRouter>,    </div>  );  }    export default App; |
| Головна сторінка | import React, { Component } from 'react';  import ProductsViewer from '../products-viewer'  import AddProduct from '../add-product'  import {Link, Route,Switch} from 'react-router-dom';  import EditProduct from '../edit-product/edit-product';  import Home from '../home'  class Main extends Component {  constructor(props) {  super(props);  this.state = { }  }  render() {  return (  <>  <div>  <ul>  <li><Link to="/">Home</Link></li>  <li><Link to="/products">Products</Link></li>  <li><Link to="/products/add">Add</Link></li>  </ul>  </div>  <hr/>  <div>  <Switch>  <Route path="/" exact component={Home} />  <Route path="/products" exact component={ProductsViewer} />  <Route path="/products/add" exact component={AddProduct} />  <Route path="/products/edit/:productId" exact component={EditProduct} />  <Route component={Home} />  </Switch>;  </div>  </>  );  }  }    export default Main;  ========================== index.js =========================  import Main from './main'  export default Main |
| Компонент для відображення продуктів | import React, { Component } from 'react';  const ProductsList = ({products,onEdit,onDelete}) => {  console.log('products:',products)  return ( <table>  <tbody>  {  (products && products.length)?(  products.map((product)=>(  <tr key={product.\_id}>  <td>{product.title}</td>  <td>{product.price}</td>  <td>{product.count}</td>  <td>  <button onClick={()=>onEdit(product.\_id)}>  Edit  </button>  </td>  <td>  <button onClick={()=>onDelete(product.\_id)}>  Delete  </button>  </td>  </tr>  ))  ):'empty'  }  </tbody>  </table> );  }    export default ProductsList;  ========================== index.js =========================  import ProductsList from './products-list'  export default ProductsList |
| Сторінка для відображення продуктів | import React, { Component } from 'react';  import axios from 'axios'  import ProductList from '../../components/products-list'  class ProductsViewer extends Component {  constructor(props) {  super(props);  this.state = { }  }  componentDidMount(){  axios.get('http://localhost:3000/api/products')  .then(res=>res.data)  .then(products=>{  this.setState({  products  })  })  .catch((err)=>{  console.log('Fetch error:',err)  })  }  onEdit=(productId)=>{  this.props.history.push(`/products/edit/${productId}`)  }  onDelete=(productId)=>{  axios.delete(`http://localhost:3000/api/products/${productId}`)  .then(()=>console.log('deleted'))  .catch((err)=>{  console.log(err)  })  this.props.history.push(`/`)  }  render() {  const {products}=this.state  return (  <ProductList products={products} onEdit={this.onEdit} onDelete={this.onDelete}/>  );  }  }    export default ProductsViewer;  ========================== index.js =========================  import ProductsViewer from './products-viewer'  import {withRouter} from 'react-router-dom'  export default withRouter(ProductsViewer) //🡨 Підключаємо до роутера |
| Сторінка для додавання нових продутів (два варіанти) | ======================== Варіант 1 (використовуємо форму і action -автоматична відправка даних)================  import React, { Component } from 'react';  const AddProduct = () => {  return (  <form action='http://localhost:3000/api/products' method='post'>  <label>  Title:  <input name='title'/>  </label>  <label>  Price:  <input name='price' type='number'/>  </label>  <label>  Count:  <input name='count' type='number'/>  </label>  <input type='submit' value='send'/>  </form> );  }    export default AddProduct;  ======================== Варіант 12(самостійно формуємо запит post)================  import React, { Component } from 'react';  import axios from 'axios'  class AddProduct extends Component {  constructor(props) {  super(props);  this.state={  errorMsg:''  }  }  sendData=(e)=>{  e.preventDefault() //🡨 Відміняємо відправку форми  const newProduct={  'title':this.refs.title.value,  'price':this.refs.price.value,  'count':this.refs.count.value  }  axios.post('http://localhost:3000/api/products',newProduct)  .then(()=>{  this.props.history.push('/')  })  .catch((err)=>{  this.setState({  errorMsg:err.message  })  })  }  render() {  const {errorMsg}= this.state  return (  <>  <form >  <label>  Title:  <input ref="title" name='title'/>  </label>  <label>  Price:  <input ref="price" name='price' type='number' />  </label>  <label>  Count:  <input ref="count" name='count' type='number' />  </label>  <input type='submit' value='send' onClick={(e)=>this.sendData(e)}/>  </form>  {  errorMsg?errorMsg:''  }  </>  );  }  }  export default AddProduct;  ========================== index.js =========================  import AddProduct from './add-product'  export default AddProduct |
| Сторінка редагування товару | import React, { Component } from 'react';  import axios from 'axios'  class EditProduct extends Component {  constructor(props) {  super(props);  this.state={  errorMsg:''  }  }  componentDidMount(){  const productId=this.props.match.params.productId  axios.get(`http://localhost:3000/api/products/${productId}`)  .then(res=>res.data)  .then(product=>{  this.refs.title.value=product.title  this.refs.price.value=product.price  this.refs.count.value=product.count  this.id=product.\_id  })  .catch((err)=>{  console.log('Fetch error:',err)  })  }  sendData=(e)=>{  e.preventDefault()  const newProduct={  'productid':this.id,  'title':this.refs.title.value,  'price':this.refs.price.value,  'count':this.refs.count.value  }  axios.put('http://localhost:3000/api/products',newProduct)  .then(()=>{  this.props.history.push('/')  })  .catch((err)=>{  this.setState({  errorMsg:err.message  })  })  }  render() {  const {errorMsg}= this.state  return (  <>  <form >  <label>  Title:  <input ref="title" name='title'/>  </label>  <label>  Price:  <input ref="price" name='price' type='number' />  </label>  <label>  Count:  <input ref="count" name='count' type='number' />  </label>  <input type='submit' value='send' onClick={(e)=>this.sendData(e)}/>  </form>  {  errorMsg?errorMsg:''  }  </>  );  }  }    export default EditProduct;  ========================== index.js =========================  import EditProduct from './edit-product'  import {withRouter} from 'react-router-dom'  export default withRouter( EditProduct) |
|  | import React, { Component } from 'react';  const Home = () => {  return ( 'Home' );  }    export default Home;  ========================== index.js =========================  import Home from './home'  export default Home |
|  | ========================== index.js ========================= |
|  | ========================== index.js ========================= |
|  | ========================== index.js ========================= |
|  | ========================== index.js ========================= |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Дія | Успішно |  |
| get | read | 200 |  |
| post | create | 201 |  |
| put | update | 200 |  |
| delete | delete | 204 |  |

400 – bad request

401 – not enough access

403 – forbidden

404 – not found

500 – server error