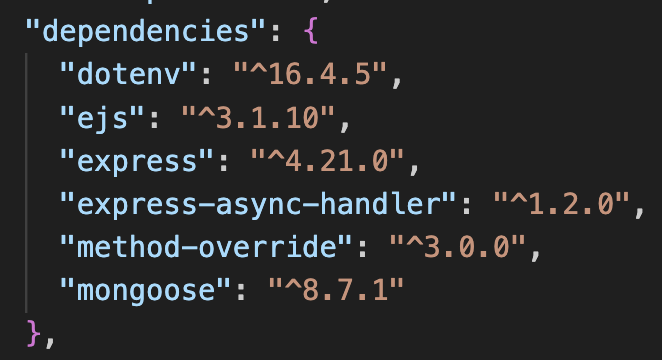
**스크린샷**

**텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

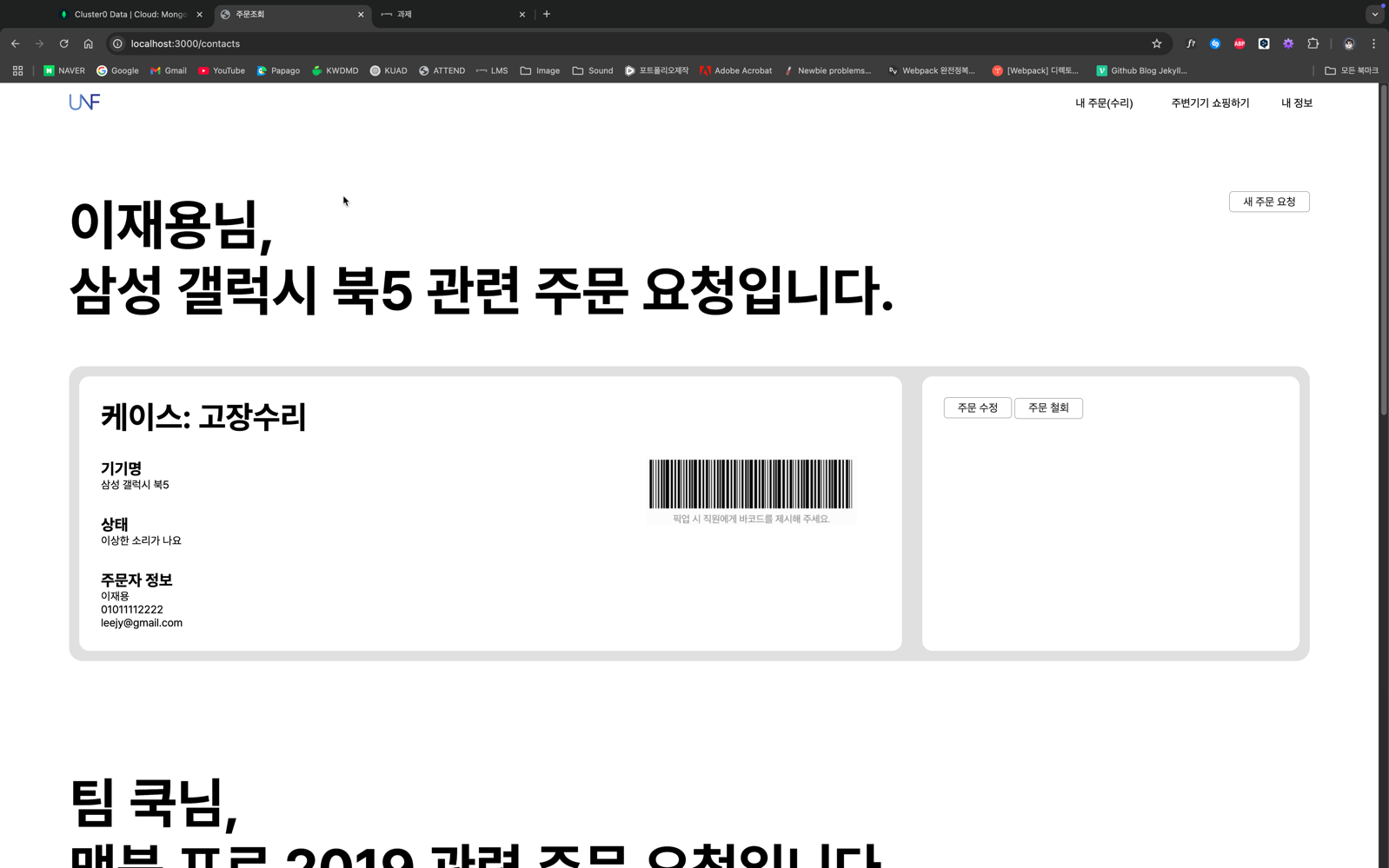
**VSC익스플로러 package.json**

**텍스트, 스크린샷, 웹 페이지, 소프트웨어이(가) 표시된 사진

자동 생성된 설명MongoDB 사이트**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**VSC MongoDB**

**DB 페이지**

**코드**

**dbConnect.js**

const mongoose = require('mongoose');

require('dotenv').config();

const dbConnect = async () => {

try {

const connect = await mongoose.connect(process.env.DB\_CONNECT);

console.log('DB Connectd');

} catch (err) {

console.log(err);

}

};

module.exports = dbConnect;

**contactController.js**

const asyncHandler = require('express-async-handler');

const Contact = require('../models/contactModel')

const path = require("path");

// @desc Get all contacts

// @route Get /contacts

// 수정 전

// const getAllContacts = async(req, res) => {

// try {

// res.status(200).send('Contacts Page');

// } catch(err) {

// res.send(error.message);

// }

// };

// 수정 후

const getAllContacts = asyncHandler(async (req, res) => {

const contacts = await Contact.find();

// const users = [

// { name: "John", email: "john@aaa.bbb", phone: "123456789"},

// { name: "Craig", email: "craig@aaa.bbb", phone: "67891234"},

// ]

// res.status(200).send("<h1 style='color: green'>Contacts Page</h1>");

// const filePath = path.join(\_\_dirname, "../assets", "getAll.html")

// res.sendFile(filePath);

res.render("index", { contacts: contacts });

});

// mid 추가

const addContactForm = (req, res) => {

res.render("newOrder");

}

const createContact = asyncHandler(async (req, res) => {

console.log(req.body);

const { devicename, casewhat, sangtae, name, phone, email } = req.body;

if(!devicename || !casewhat || !sangtae || !name || !phone || !email) {

return res.status(400).send('필수값이 입력되지 않았습니다.');

}

const contact = await Contact.create({

devicename,

casewhat,

sangtae,

name,

phone,

email,

});

// res.status(201).send("사용자 데이터 생성됨");

res.redirect("/contacts"); //mid add

});

const getContact = asyncHandler(async (req, res) => {

const contact = await Contact.findById(req.params.id);

//mid add

// res.status(200).send(contact);

res.render('update', { contact: contact })

});

const updateContact = asyncHandler(async (req, res) => {

// res.status(200).send(`Update Contact for ID: ${req.params.id}`);

const id = req.params.id;

const { devicename, casewhat, sangtae, name, phone, email } = req.body;

const contact = await Contact.findById(id);

if(!contact) {

res.status(404);

throw new Error('사용자 데이터 찾을 수 없음');

}

// 수정

contact.devicename = devicename;

contact.casewhat = casewhat;

contact.sangtae = sangtae;

contact.name = name;

contact.email = email;

contact.phone = phone;

// 저장

contact.save();

//mid add

// res.status(200).json(contact);

res.redirect("/contacts");

});

const deleteContact = asyncHandler(async (req, res) => {

//mid add

// const contact = await Contact.findById(req.params.id);

// if(!contact) {

// res.status(404);

// throw new Error("사용자 데이터를 찾을 수 없음");

// }

// await Contact.deleteOne({ \_id: req.params.id });

// res.status(200).send(`해당 사용자 ID 삭제됨: ${req.params.id}`)

await Contact.findByIdAndDelete(req.params.id);

res.redirect("/contacts")

});

module.exports = {

getAllContacts,

createContact,

getContact,

updateContact,

deleteContact,

addContactForm,

};

**contactModel.js**

const mongoose = require('mongoose');

const userdataSchema = new mongoose.Schema(

{

devicename: {

type: String,

},

casewhat: {

type: String,

},

sangtae: {

type: String,

},

name: {

type: String,

},

email: {

type: String,

},

phone: {

type: String,

},

},

{

timestamps: true,

}

);

const Contact = mongoose.model('Userdata', userdataSchema);

module.exports = Contact;

**errorhandler.js**

const errorhandler = (err, req, res, next) => {

const status = err.status || 500;

switch (status) {

// 400: 잘못된 요청

case 400:

res.status(status).json({

title: 'Bad Request',

message: err.message

});

break;

// 401: 권한 없음

case 401:

res.status(status).json({

title: 'Unauthorized',

message: err.message

});

break;

// 403: 금지됨

case 403:

res.status(status).json({

title: 'Forbidden',

message: err.message

});

break;

// 404: 찾을 수 없음

case 404:

res.status(status).json({

title: 'Not Found',

message: err.message

});

break;

// 500: 서버 오류

case 500:

res.status(status).json({

title: 'Internal Server Error',

message: err.message

});

break;

default:

res.status(status).json({

message: 'No Error!'

});

break;

}

}

module.exports = errorhandler

**contactRoutes.js**

const express = require('express');

const router = express.Router();

const {

getAllContacts,

createContact,

getContact,

updateContact,

deleteContact,

addContactForm, //mid add

} = require('../controllers/contactController');

const { create } = require('../models/contactModel');

// router

// .route("/")

// 모든 연락처 가져오기

// .get((req, res) => {

// res.status(200).send("Contacts Page");

// })

// .post((req, res) => {

// // 새 연락처 추가하기

// console.log(req.body);

// const { name, email, phone } = req.body;

// if(!name || !email || !phone) {

// return res.status(400).send('필수값이 입력되지 않았습니다.');

// }

// res.status(201).send("Create Contacts");

// })

// router

// .route("/:id")

// .get((req, res) => {

// res.status(200).send(`View Contact for ID: ${req.params.id}`);

// })

// .put((req, res) => {

// res.status(200).send(`Update Contact for ID: ${req.params.id}`);

// })

// .delete((req, res) => {

// res.status(200).send(`Delete Contact for ID: ${req.params.id}`);

// });

router.route("/").get(getAllContacts);//.post(createContact);

router.route("/newOrder").get(addContactForm).post(createContact); //mid add

router.route("/:id").get(getContact).put(updateContact).delete(deleteContact);

module.exports = router;

**app.js**

const express = require('express');

const dbConnect = require('./config/dbConnect');

const methodOverride = require("method-override"); //mid add

const errorhandler = require('./middlewares/errorhandler');

const app = express();

app.set("view engine", "ejs");

app.set("views", "./views");

// load public files

app.use(express.static("./public"))

app.use(methodOverride("\_method")); //mid add

const port = 3000;

// const logger = (req, res, next) => {

// console.log('User Logged');

// next();

// }

// app.use(logger);

const requestTime = (req, res, next) => {

let today = new Date(); // Date 객체 만들기

let now = today.toLocaleTimeString(); // 현재 시간을 문자열로 바꾸기

req.requestTime = now; // req 객체에 requestTime 속성 추가하기

next();

};

// app.use(requestTime)

// app.route("/contacts").get( (req, res) => {

// const responseText = `Hello Node! \n 요청시간: ${req.requestTime}`;

// res.set('Content-type', 'text/plain');

// res.send(responseText)

// // res.status(200).send("Hello Node!");

// });

dbConnect();

// contactRoutes를 통해 /contacts 라우트를 처리

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use("/contacts", require("./routes/contactRoutes"));

app.get("/test", (req, res, next) => {

const error = new Error('테스트용 에라');

error.status = 401;

next(error);

});

app.use(errorhandler)

app.listen(port, () => {

console.log(`${port}번 포트에서 서버 실행 중`);

});

**\_header.ejs**

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>주문조회</title>

<link rel="stylesheet" href="/css/style.css">

<link rel="stylesheet" as="style" crossorigin href="https://cdn.jsdelivr.net/gh/orioncactus/pretendard@v1.3.9/dist/web/static/pretendard.min.css" />

</head>

<body>

<!-- HEADER -->

<header>

<nav>

<div id="nav\_logo\_white">

<a href="/contacts/getAll.html"><img src="https://synologykct.synology.me/img/unifix\_logo.png" alt="nav\_logo" height="25"></a>

</div>

<ul class="nav\_menu">

<li><button class="order">내 주문(수리)</button></li>

<li><button class="shopping">주변기기 쇼핑하기</button></li>

<li><button class="info">내 정보</button></li>

</ul>

</nav>

</header>

<!-- /HEADER -->

**\_footer.ejs**

</body>

</html>

**index.ejs**

<%- include('./include/\_header') %>

<div class="clear"></div>

<!-- MAIN -->

<main>

<div id="sec1\_bg">

<a href="/contacts/newOrder" class="btn new">새 주문 요청</a>

<% contacts.forEach(contact => { %>

<section id="sector1">

<div id="title\_box">

<h1><%= contact.name %>님,<br><%= contact.devicename %> 관련 주문 요청입니다.</h1>

</div>

<article id="status\_box">

<div id="flexbox">

<div class="status one">

<div id="flexbox2">

<div class="left\_status one">

<p class="current\_status">케이스: <%= contact.casewhat %><p>

<p class="main\_title">기기명</p>

<span class="subtitle"><%= contact.devicename %></span>

<p class="main\_title">상태</p>

<span class="subtitle"><%= contact.sangtae %></span>

<p class="main\_title">주문자 정보</p>

<span class="subtitle">

<%= contact.name %> <br>

<%= contact.phone %> <br>

<%= contact.email %>

</span>

</div>

<div class="left\_status two">

<img src="https://synologykct.synology.me/img/code.png" alt="code">

</div>

</div>

</table>

</div>

<div class="status two">

<a href="/contacts/<%= contact.\_id %>" class="btn update">주문 수정</a>

<form action="/contacts/<%= contact.\_id %>?\_method=DELETE" method="POST" style="display: inline">

<input type="submit" class="btn" value="주문 철회">

</form>

</div>

</div>

</article>

</section>

</div>

<% }); %>

</main>

<!-- /MAIN -->

<%- include('./include/\_footer') %>

**newOrder.ejs**

<%- include('./include/\_header') %>

<div class="clear"></div>

<!-- MAIN -->

<main>

<div id="sec1\_bg">

<section id="sector\_neworder">

<h1 class="title">새 주문서 작성</h1>

<form action="" method="POST" id="addOrder">

<p class="subtitle">기기 정보</p>

<label for="deviceName">수리맡길 기기 이름을 적어주세요.</label><br>

<input type="text" class="orderInput" placeholder="ex. 삼성 노트북" name="devicename"><br><br>

<label for="">케이스를 선택해 주세요.</label><br>

<select name="casewhat" class="orderInput">

<option value="고장수리">고장수리</option>

<option value="업그레이드(추가)">업그레이드(추가)</option>

<option value="유지보수(청소)">유지보수(청소)</option>

</select> <br><br>

<label for="sangtae">현재 상태를 간략하게 적어주세요.</label><br>

<input type="text" class="orderInput" placeholder="ex. 전원이 안 켜져요" name="sangtae"><br><br>

<br><br>

<p class="subtitle">주문자 정보</p>

<label for="name">주문자명을 입력해 주세요.</label><br>

<input type="text" class="orderInput" placeholder="ex. 홍길동" name="name"><br><br>

<label for="name">이메일 주소를 입력해 주세요.</label><br>

<input type="email" class="orderInput" placeholder="example@aaa.bbb" name="email"><br><br>

<label for="phone">전화번호를 입력해 주세요.</label><br>

<input type="number" class="orderInput" placeholder="01000000000" name="phone"><br><br>

<input type="submit" value="제출" class="orderSubmit">

<a href="/contacts" class="btn back">주문 페이지로 돌아가기</a>

</form>

</section>

</div>

</main>

<!-- /MAIN -->

<%- include('./include/\_footer') %>

**update.ejs**

<%- include('./include/\_header') %>

<div class="clear"></div>

<!-- MAIN -->

<main>

<div id="sec1\_bg">

<section id="sector\_neworder">

<h1 class="title">주문 수정</h1>

<form action="/contacts/<%= contact.\_id %>?\_method=PUT" method="POST" id="addOrder">

<p class="subtitle">기기 정보</p>

<label for="deviceName">수리맡길 기기 이름을 적어주세요.</label><br>

<input type="text" class="orderInput" name="devicename" value="<%= contact.devicename %>"><br><br>

<label for="">케이스를 선택해 주세요.</label><br>

<select name="casewhat" class="orderInput" value="<%= contact.casewhat %>">

<option value="고장수리">고장수리</option>

<option value="업그레이드(추가)">업그레이드(추가)</option>

<option value="유지보수(청소)">유지보수(청소)</option>

</select> <br><br>

<label for="sangtae">현재 상태를 간략하게 적어주세요.</label><br>

<input type="text" class="orderInput" name="sangtae" value="<%= contact.sangtae %>"><br><br>

<br><br>

<p class="subtitle">주문자 정보</p>

<label for="name">주문자명을 입력해 주세요.</label><br>

<input type="text" class="orderInput" name="name" value="<%= contact.name %>"><br><br>

<label for="name">이메일 주소를 입력해 주세요.</label><br>

<input type="email" class="orderInput" name="email" value="<%= contact.email %>"><br><br>

<label for="phone">전화번호를 입력해 주세요.</label><br>

<input type="number" class="orderInput" name="phone" value="<%= contact.phone %>"><br><br>

<input type="submit" value="제출" class="orderSubmit">

<a href="/contacts" class="btn back">주문 페이지로 돌아가기</a>

</form>

</section>

</div>

</main>

<!-- /MAIN -->

<%- include('./include/\_footer') %>

**style.css**

\* {

margin: 0;

padding: 0;

}

body {

font-family: "Pretendard", sans-serif;

word-break: keep-all;

}

.clear {

clear: both;

}

header {

top: 0;

left: 0;

position: fixed;

/\*border: solid 1px red;\*/

width: 100%;

color: #1f1f1f;

z-index: 1;

transition: ease .5s;

}

nav {

/\*border: solid 1px blue;\*/

width: 90%;

height: 60px;

margin: 0 auto;

}

#nav\_logo\_white {

display: flex;

place-items: center;

float: left;

/\*border: solid 1px green;\*/

height: 60px;

}

.nav\_menu {

list-style-type: none;

float: right;

line-height: 60px;

}

.nav\_menu li {

margin-left: 50px;

display: inline-block;

vertical-align: center;

}

.nav\_menu button {

width: 120%;

height: 20px;

font-size: 1em;

/\* font-weight: 200; \*/

position: relative; /\* absolute 위치 설정을 위한 상대적 위치 설정 \*/

border: none;

background: none;

}

.nav\_menu button:hover {

font-weight: 700;

color: var(--blueB);

}

.nav\_menu button:hover::after {

content: "";

position: absolute;

left: 0;

bottom: -5px;

width: 100%;

height: 2px; /\* border-bottom의 두께 \*/

background-color: var(--blueB);

animation: slideIn 0.5s ease; /\* slideIn 애니메이션을 0.5초 동안 적용 \*/

}

.nav\_menu button:active {

opacity: .5;

}

#sec1\_bg {

/\* border: solid 1px blue; \*/

width: 100%;

/\* height: 100%; \*/

/\* padding-bottom: 10rem; \*/

margin-top: 10em;

}

section#sector1 {

/\* border: solid 1px red; \*/

display: block;

width: 90%;

margin: 0 auto;

padding-bottom: 10rem;

}

#title\_box h1 {

font-size: 7vmin;

}

article#status\_box {

background: rgb(224, 224, 224);

border-radius: 20px;

width: 100%;

/\* height: 450px; \*/

margin-top: 4em;

}

#flexbox {

display: flex;

}

#flexbox .status {

background: #fff;

/\* border: solid 1px red; \*/

width: 70%;

/\* height: calc(450px - 6em); \*/

/\* height: 100%; \*/

border-radius: 15px;

margin: 15px;

padding: 2em;

}

#flexbox .status.two {

/\* border: solid 1px red; \*/

width: 30%;

}

.current\_status {

font-size: 4vmin;

font-weight: bold;

}

.main\_title {

margin-top: 1.5em;

font-size: 2vmin;

font-weight: bold;

}

#flexbox2 {

display: flex;

}

.left\_status {

/\* border: solid 1px red; \*/

/\* width: calc(50% - 10px); \*/

justify-content: space-between;

width: 70%;

}

.left\_status.two {

width: 30%;

}

.left\_status img {

/\* border: solid 1px red; \*/

width: 90%;

margin-top: 25%;

}

.status.two img {

width: 100%;

}

/\* NEW ORDER \*/

section#sector\_neworder {

border: solid .5px rgb(213, 213, 213);

border-radius: 20px;

background: #fff;

box-shadow: 2px 2px 15px rgba(0, 0, 0, .1);

width: 600px;

height: 600px;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

padding: 30px;

}

h1.title {

text-align: center;

font-size: xx-large;

}

p.subtitle {

margin-top: 20px;

margin-bottom: 5px;

font-weight: 700;

}

label {

color: #606060;

font-style: italic;

}

.orderInput {

width: 100%;

height: 30px;

border: solid .5px #606060;

border-radius: 5px;

background: #fff;

}

.orderSubmit {

width: 30%;

height: 30px;

background: #fff;

border: solid .5px #606060;

border-radius: 5px;

}

.orderSubmit:hover {

background: #dddddd;

}

.btn {

background: #fff;

border: solid .5px #606060;

text-decoration: none;

color: #1f1f1f;

padding: 5px 20px;

border-radius: 5px;

font-size: 16px;

}

.btn.new {

position: absolute;

right: 5%;

}

.btn.back {

position: absolute;

right: 5%;

background: #eef5ff;

}

.btn:hover {

background: #ddd;

}