

MERN 스택 개발 보고서

2022년 04월 27일

비트캠프 KDT 5기 권솔이

Index

1. MEAN Stack

1.1 MongoDB

1.2 Express

1.3 React

1.4 Node

2. 개발환경

2.1 Node

2.2 React

2.3 Express

2.4 MongoDB

3. 프로그램 개발 과정

3.1 React(components)

3.2 Next(pages)

3.3 Redux+Saga

3.4 Entry point(server.js)

3.5 Router(routes)

3.6 Middleware(service)

3.7 models

3.8 Connector(lambdas)

1. MERN stack이란?

MERN은 MongoDB, Express, React, Node를 의미하며, 이는 스택을 구성하는 네 가지 핵심 기술을 의미한다. LIFO(Last In First Out)형식의 자료구조이다. Node.js -> React.js -> Express.js -> MongoDB 순서의 개발 과정을 가진다.

1.1 MongoDB는 BSON 문서 형태로 데이터를 저장하는 NoSQL 데이터베이스 시스템이다.

NoSQL은 NO not on SQL로 비관계형 데이터 베이스의 작동 방식을 가지고 있다. 주로 데이터가 고정되어 있지 않는 데이터베이스를 말하며 유연한 스키마를 제공한다.

1.2 Express는 서버 프레임워크이다

1.3 React는 컴포넌트기반 자바스크립트 라이브러리이다.

1.4 Node는 환경이다. Node.js를 통해 서버를 실행할 수 있다. 내장 HTTP서버 라이브러리를 포함하고 있어 웹 서버 동작에 있어 더 많은기능을 가능하게 한다.

2. 개발환경

2.1 Node

노드를 설치한다. 고급 시스템 설정에서 환경변수를 클릭하여 시스템 변수를 새로 만든다. 시스템 변수 이름을 설정하고 변수 값에는 설치경로를 입력해준다. 시스템 변수에서 Path를 찾아 설정값을 입력해준다. cmd 창에 "node -version"을 입력하여 설치 확인한다.

2.2 React

VS code를 설치 후 왼쪽 바 extensions에서 필요한 확장 프로그램을 찾아 설치해준다.(Node Exec,Node.js Extension Pack,Node.js Modules Intellisense,React Native Tools,ES7React/Redux/GraphQL/React-Nativesnippets,React-Native/React/Reduxsnippets for es6/es7,Reactjs code snippets 등)

폴더를 생성하기 위해 가장 먼저 VS code를 관리자 권한으로 실행한다. 관리자 권한으로 실행하지 않을시 권한에러가 발생하는 경우가 있다.터미널에 “npm -version”을 입력하여 설치 여부 체크 후 없다면“npm i -g yarn” 입력하여 설치한다. 터미널에 폴더 생성 명령어를 입력해준다.

```
Set-ExecutionPolicy Unrestricted
```

```
ExecutionPolicy
```

```
npm install -g create-react-app
```

```
create-react-app my-app
```

Happy haking이 보이면 “cd my-app” 와 “npm start”를 입력한다. 방화벽이 보이면 액세스를 허용한다.

2.3 Express

Express를 설치하기 위해서는 Node가 설치 되어 있어야한다.cmd창에 하위 디렉토리를 생성하는 명령어를 입력한다.“mkdir 디렉토리명” “dir”을 입력하여 디렉토리가 만들어 졌나 확인 한다. 만들어 졌다면 익스프레스 설치 명령어를 입력한다.“npx express-generator”, 마지막으로 파일 생성과 로컬 호스트가 연결되었는지 확인한다.

2.4 MongoDB

가장 먼저 Docker을 설치한다. 도커 설치가 끝난 후 터미널에서 아래와 같이 입력하여 잘 설치되었는지 확인한다. “docker -version” , “docker pull adoptopenjdk/open/openjdk11”,”java -version”

명령어를 입력하여 도커에 MongoDB를 설치한다.

```
docker run -d -p 80:80 docker/getting-started
```

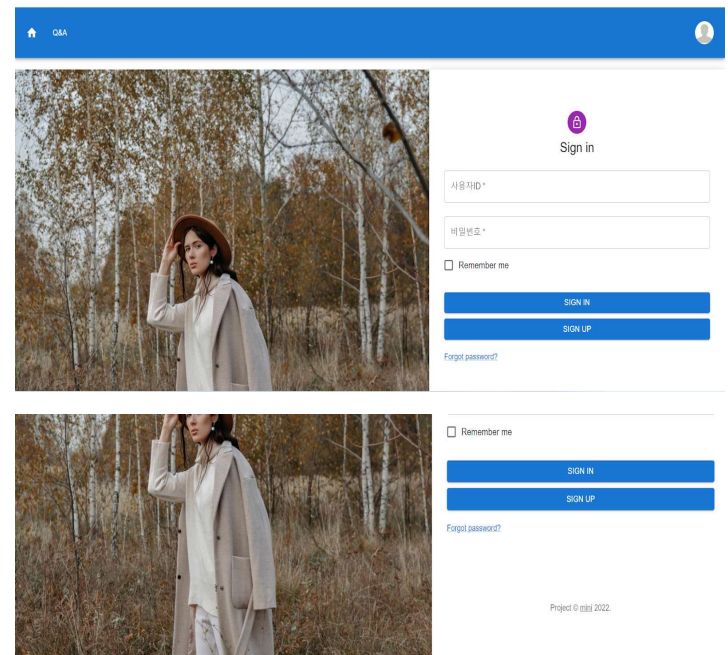
```
docker pull mongo
```

```
docker run --name mongodb -e
MONGO_INITDB_ROOT_USERNAME=root -e
MONGO_INITDB_ROOT_PASSWORD=root -d -p
27017:27017 mongo
mongosh "mongodb://localhost:27017" --username "root"
--password "root"
```

3. 프로그램 개발 과정

3.1 React(components)

내 삶을 바꾼 Q&A



“카르페 디엠, 지금 이 순간을 기록하라”
누구나 자신의 삶을 기록으로 남기고 싶어 한다. 책상 무엇을 써야할지 몰라 포기하는 경우가 비일비재하다. 우리의 삶을 포착하게 해주는 이미 있는 365개의 질문이 하루에 하나씩 제공된다. “마지막으로 물었던 책은 언제인가?” “토요일 오전을 보내는 나만의 가장 행복한 방법이란?” 등 나 자신에 대해 생각하고 공들 쓰는 기회를 제공한다. 바쁜 일상 속 지레로 쓴 방법으로 우리의 일기 쓰기 쉽게 하고, 삶의 가장 빛나는 순간을 담게 된다.

Component는 사용자에게 보여지는 화면이다 .Component 폴더 안에 auth 폴더를 생성하여Login.js,Register.js,Profile.js 파일을 만들었다. 또한 공용으로 사용하는 common이라는 폴더안에 Header.js, Nav.js ,Footer.js 파일을 생성했다. 홈페이지 접속시 바로 로그인 할 수 있도록 메인에 배치했다. 메인 사진은 재미를 더 하기위해 랜덤으로 바뀔 수 있도록 코딩했다. Nav에는 홈화면과 Q&A 게시판 사용자화면으로 구성했다. Header에는 텍스트에 링크를 걸어 즐겨 찾기 한 Q&A로 바로 갈 수 있도록 했다. 마지막으로 Footer에는 간략하게 프로그램의 목적을 보여질 수 있도록 했다.

사용자가 SIGN UP 버튼을 눌렀을때 회원가입 화면이 보여지도록 했다.components는 사용자에게 보여지는 화면이기 때문에 상태가 없이 코딩했다. 사용자가 입력한 데이터를 onChange,onSubmit을 넥스트에게 넘겨준다.

```
export function Register({onChange, onSubmit}){
  return (
    <ThemeProvider theme={theme}>
      <Head>
        <title>사용자| </title>
      </Head>
      <Container component="main" maxWidth="xs">
        <CssBaseline />
        <Box
          sx={{
            marginTop: 8,
            display: 'flex',
            flexDirection: 'column',
            alignItems: 'center',
          }}
        >
          <Avatar sx={{ m: 1, bgcolor: 'secondary.main' }}>
            <LockOutlinedIcon />
          </Avatar>
          <Typography component="h1" variant="h5">
            회원가입
          </Typography>
          <Box component="form" noValidate sx={{ mt: 3 }} onSubmit={onSubmit} >
            <Grid container spacing={2}>
              <Grid item xs={12} sm={6} >
                <TextField
                  autoComplete="given-name"
                  name="userid"
                  required
                  fullWidth
                  id="userid"
                  label="사용자ID"
                  autoFocus
                  onChange={onChange}
                />
              </Grid item>
            </Grid>
          </Box>
        </Container>
      </ThemeProvider>
    )
  }
}
```

3.2 Next(pages)

Components와 동일한 구조를 이룬다. pages 폴더 안에 auth 폴더를 생성하고 login.js,register.js 파일을 생성한다.

```
import { registerRequest, unregisterRequest } from '@modules/auth/register';
import { Register } from '@components';

const RegisterPage = () => {
  const [user, setUser] = useState({
    userid: '', password: '', email: '', name: '', phone: '', birth: '', address: ''
  });
  const dispatch = useDispatch();
  const onChange = e => {
    e.preventDefault();
    const {name, value} = e.target;
    setUser({...user, [name]: value});
  };
  const onSubmit = e => {
    e.preventDefault();
    alert('회원가입정보: ' + JSON.stringify(user));
    dispatch(registerRequest(user));
  };
  return (
    <Register onChange={onChange} onSubmit={onSubmit} />
  );
};

const mapStateToProps = state => ({ isRegistered: state.register.isRegistered });
const registerActions = {registerRequest, unregisterRequest};
export default connect(mapStateToProps, registerActions)(RegisterPage);
```

리액트에서 입력 받은 데이터를 넥스트로 넘겨준다. 프롭스가 객체일때는 부모가 자식한테 넘겨주고, 함수일때는 자식이 부모한테 넘겨준다. 넥스트는 상태가 있고 return안에 리액트를 가지고 있기 때문에 부모이다.

3.3 Redux+Saga

```
const SERVER = 'http://127.0.0.1:5000'
const headers = {
  "Content-Type": "application/json",
  Authorization: "JWT fefege..."
}
export const initialState = {
  isRegistered: false
}

const REGISTER_REQUEST = 'auth/REGISTER_REQUEST';
const REGISTER_SUCCESS = 'auth/REGISTER_SUCCESS';
const REGISTER_FAILURE = 'auth/REGISTER_FAILURE';
const UNREGISTER_REQUEST = 'auth/UNREGISTER_REQUEST';
const UNREGISTER_SUCCESS = 'auth/UNREGISTER_SUCCESS';
const UNREGISTER_FAILURE = 'auth/UNREGISTER_FAILURE';

export const registerRequest = createAction(REGISTER_REQUEST, data => data)
export const unregisterRequest = createAction(UNREGISTER_REQUEST, data => data)
// 위 아래 동일한 코드임.
// export const registerRequest = data => (
//   {type: REGISTER_REQUEST, payload: data}
// )
export function* registerSaga() {
  yield takeLatest(REGISTER_REQUEST, signup);
  yield takeLatest(UNREGISTER_REQUEST, membershipWithdrawal);
}
function* signup(action) {
  try {
    const response = yield call(registerAPI, action.payload)
    console.log(" 회원가입 서버다녀옴: " + JSON.stringify(response.data))
    yield put({type: REGISTER_SUCCESS, payload: response.data})
    //put이니까 리덕스에 데이터 보내라.
  } catch {
    //
  }
}
```

리덕스 사가란, 리덕스를 사용할 때 리듀스 액션을 관찰하고 있다가 액션이 실행되면 다른 행동을 하게 하는 함수이다. 이펙트함수를 사용할 수 있다. Generate Function을 사용하여 회원가입에 성공하면 Express로 request하는 액션을 생성한다.

3.4 Entry point(server.js)

```
import dotenv from 'dotenv'
import express from 'express'
import passport from 'passport'
import morgan from 'morgan'
import db from './app/models/index.js'
import api from './app/routes/api.js'
import basic from './app/routes/basic.js'
import board from './app/routes/board.js'
import user from './app/routes/user.js'
import index from './app/routes/index.js'
import todo from './app/routes/todo.js'
import getResponse from './app/lambda/getResponse.js'
import applyPassport from './app/lambda/applyPassport.js'
import applyDotenv from './app/lambda/applyDotenv.js'

async function startServer() {
  const app = express();
  const {mongoUri, port, jwtSecret} = applyDotenv(dotenv);
  app.use(express.static('public'));
  app.use(express.urlencoded({extended: true}));
  app.use(express.json());
  const passport = applyPassport(passport, jwtSecret);
  app.use(passport.initialize());
  app.use("/", index);
  app.use("/api", api);
  app.use("/basic", basic);
  app.use("/board", board);
  app.use("/todo", passport.authenticate('jwt', { session: false}), todo);
  app.use("/user", user);
  app.use(morgan('dev'))
  db
}
```

Express(서버)는 라우더, 미들웨어, 커넥터를 반드시 가진다. Express는 엔트리 포인트 server.js에서 경로를 찾아 routes로 이동한다.

3.5 Router(routes)

```
const mongoUri = process.env.MONGO_URI
const port = process.env.PORT
const jwtSecret = process.env.JWT_SECRET
const origin = process.env.ORIGIN
const corsOptions = {
  origin: process.env.ORIGIN,
  optionsSuccessStatus: 200
}
const app = express()
app.use(cors());
app.use(function (req, res, next) {
  res.header(
    "Access-Control-Allow-Headers",
    "x-access-token, Origin, Content-Type, Accept"
  );
  next();
});
app.post('/join', cors(corsOptions), (req, res) => {
  UserService().join(req, res)
})
app.post('/login', cors(corsOptions), (req, res) => {
  UserService().login(req, res)
})
app.get(
  '/logout',
  //passport.authenticate('jwt', {session: false}),
  function (req, res) {
    console.log('logout 진입')
    //UserService().logout(req, res)
  }
)
```

request와 response의 경로 연결이다. 라우터는 URL을 갖는다.

3.6.Middleware(service)

중간에서 복잡한 연산이나 작업을 수행한다.(알고리즘)

회원가입 실패 성공 여부를 콘솔창 나타내고 models로 이동한다.

3.7 models

토큰을 걸어주고 스키마 생성해준다.

3.7.Connector(lambdas)

MongDB에 데이터를 저장해준다. MongDB에 저장된 데이터를 확인 하기 위해서 터미널에 아래와 같이 명령어를 입력한다.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대 한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows

PS C:\Users\#> mongosh "mongodb://localhost:27017/" --username "root" --password "root"
Current Mongosh Log ID: 627f24d89fa605c5348d3af6
Connecting to: mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Using MongoDB: 5.0.8
Using Mongosh: 1.1.7


For mongosh info see: https://docs.mongodb.com/mongodb-shell/

The server generated these startup warnings when booting:
2022-05-14T03:09:24.972+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2022-05-14T03:09:25.672+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'madvise' to reduce memory overcommitment.

test> show dbs
admin      102 kB
config     111 kB
local      73.7 kB
minidb     156 kB
soccordb   156 kB
test> use minidb
switched to db minidb
minidb> show collections
collections
users
minidb> db.users.find()
{
  "_id": ObjectId("627f2451dfaeda09a729792c"),
  "userId": "과제제출",
  "password": "$2b$10$uZBxIAQKzNwTz8GDUtpNe3MoQ1F/nD0KE5Z/Ez7KbVaFLm4FxN.6",
  "email": "123@naver.com",
  "name": "이슬권",
  "phone": "010-555-111",
  "birth": "19950222",
  "address": "경기도 화성시",
  "createdAt": ISODate("2022-05-14T03:38:57.974Z"),
  "updatedAt": ISODate("2022-05-14T03:38:57.974Z"),
  "_v": 0
}
```

회원가입할때 입력한 정보가 MongoDB를 넘어간 것을 확인할 수 있다. 또한 비밀번호가 보이지 않는 것을

확인할 수 있다. 성공적으로 회원가입이 되었다면 자동으로 홈 화면으로 넘어간다.로그인시에도 회원가입과 처리 경로가 동일하다.



Sign in

사용자ID*

과제 제출

비밀번호*


....

☐ Remember me


SIGN IN

SIGN UP

[Forgot password?](#)



User Profile Card



이솔권

CEO & Founder, Example

이메일 : 123@naver.com

전화번호 : 010-555-111

생년월일 : 19950222

주소 : 경기도 화성시

Contact

로그인이 성공적으로 실행되면 프로필(Profile.js)화면으로 넘어간다.

-참조-

A MERN Stack introduction-2022.05.14 검색
<https://www.mongodb.com/mern-stack>

MERN Stack-2022.05.14 검색
<https://parksrazor.tistory.co>