이력서

깃허브주소- https://github.com/ShinEunJin
운영사이트- http://bigyeong.com

소개

- 약 10개월 정도 개발 공부를 독학으로 하였습니다.
- 아쉽지만 팀 프로젝트나 실무를 직접 해본적이 없습니다. 무경력
- 1인 프로젝트로 bigyeong이라는 사이트를 운영중입니다
  (http://bigyeong.com)(모바일로도 보실 수 있습니다)
- 혼자하는 개발 공부는 비효율적이고 끝이 없을거 같아 실무에서 직접 일하며 몸으로 성장하고 싶습니다.
- **제 스택에는 없지만 만약 필요한 기술스택이 요구된다면 밤낮 안가리고 배워서 들어갈 수 있습니다.**

기술 스택

front

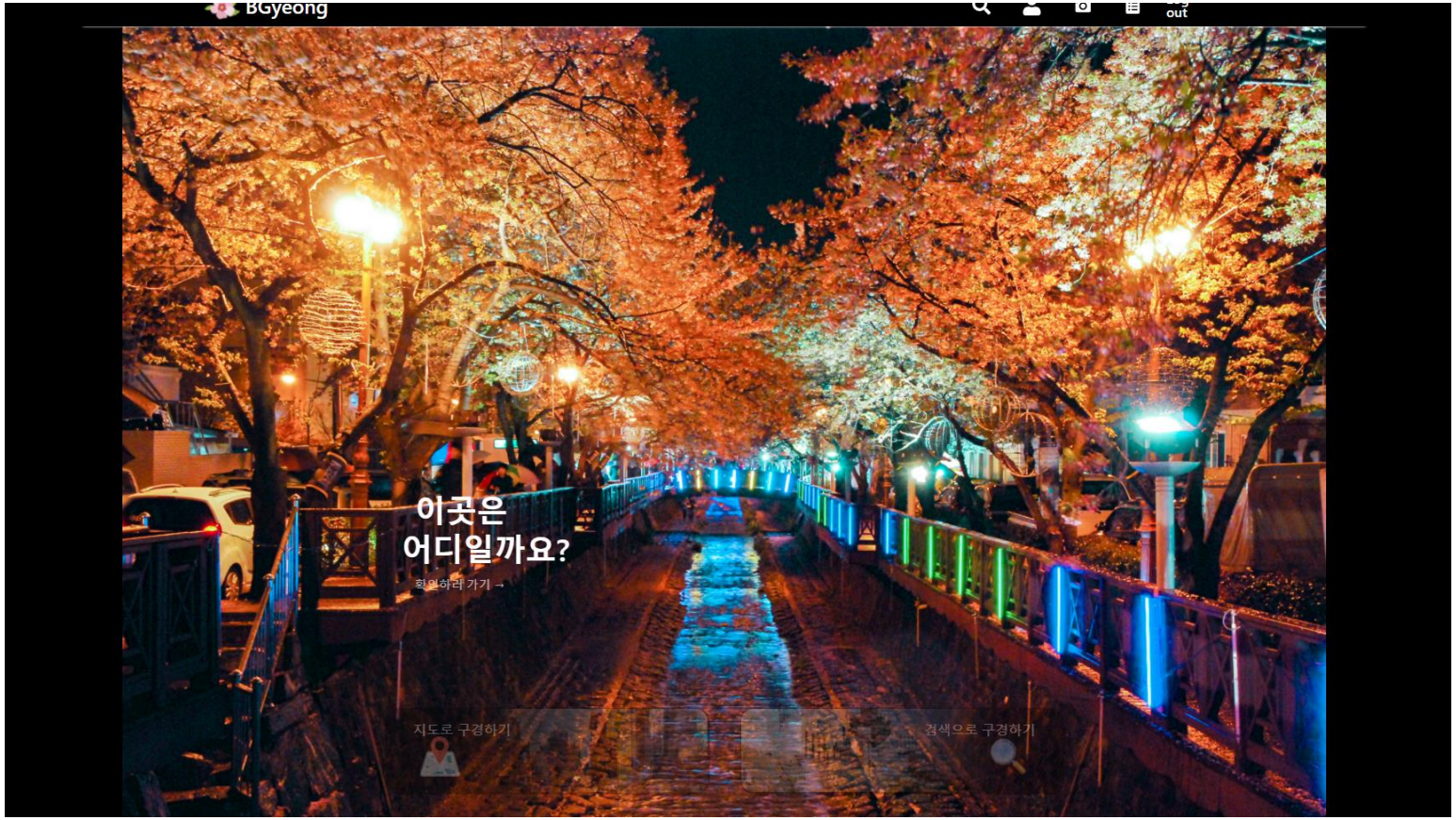
- react
- redux(thunk)
- html, css, js(ES6)

backend

- nodeJS
- express
- mongoDB

애매한 기술은 적지 않고 자신있는 것만 적었습니다

위 기술만큼은 부족함 없이 사용할 자신 있습니다



http://bigyeong.com

aws ec2를 통해 배포 후 직접 운영 중입니다. 방문자는 거의 없지만 피드백 받고 수정 하고 있으며 계속 개발 중입니다.

깃헙주소: https://github.com/ShinEunJin/bigyeong

포트폴리오

프로젝트 소개

사용 스택

Frontend

- react
- redux
- styled-components
- antd

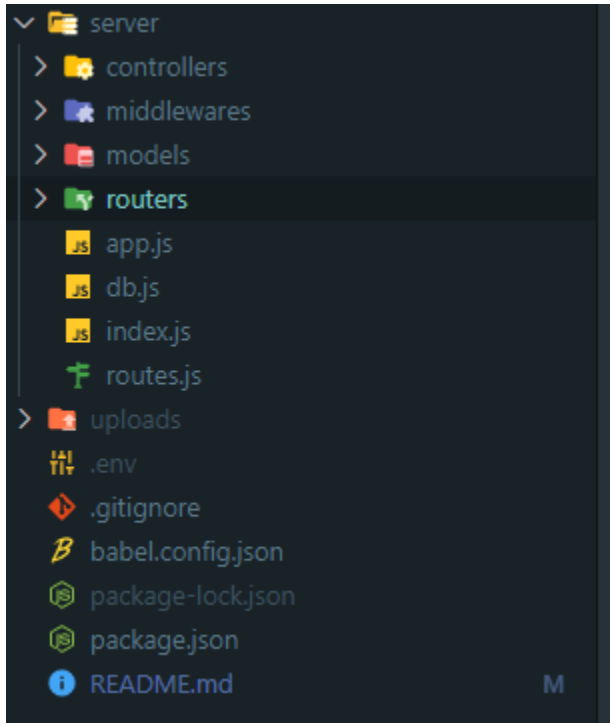
backend

- nodejs
- express
- mongoDB
- JWT
- multer
- ES6

devops

- AWS ec2
- AWS s3, route53

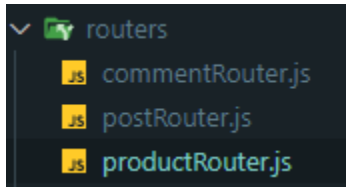
백엔드 부분



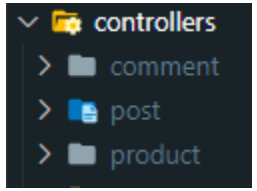
axios를 통해 클라이언트로 부터 restful api로 통신

```
app.use(express.urlencoded({ extended: true }))
app.use(express.json())
app.use(cookieParser())
```

```
app.use(routes.apiProduct, productRouter)
```

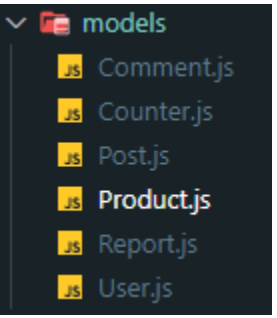



```
//product
productRouter.get("/", getProduct)
productRouter.post("/", uploadProduct)
productRouter.patch("/", updateProduct)
productRouter.delete("/", deleteProduct)
```




```
export const getProduct = async (req, res) => {
  const {
    query: { id },
  } = req
  try {
    const product = await Product.findOneAndUpdate(
      { _id: id },
      { $inc: { views: 1 } },
      { new: true }
    ).populate("writer", "-password -token")
    return res.status(200).json({ success: true, product })
  } catch (error) {
    return res.status(400).json({ success: false, error })
  }
}
```

NOSql DB

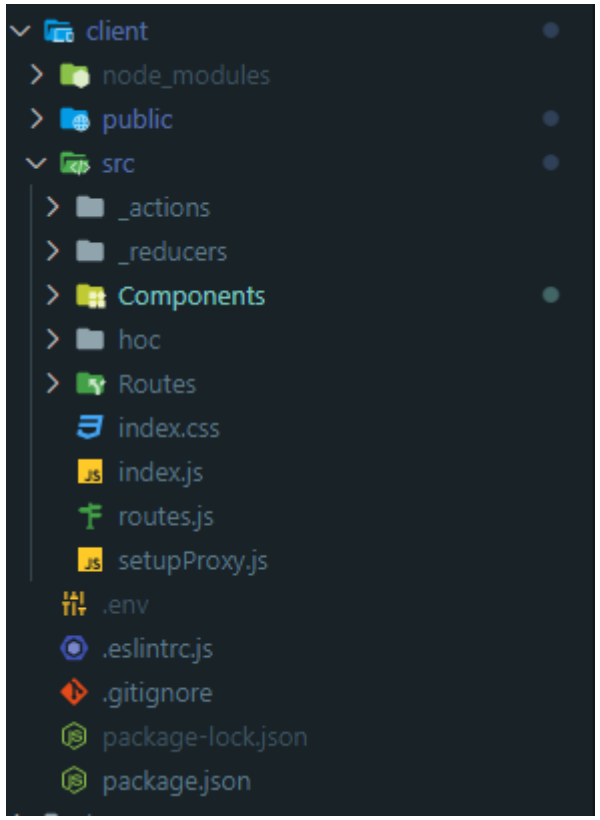


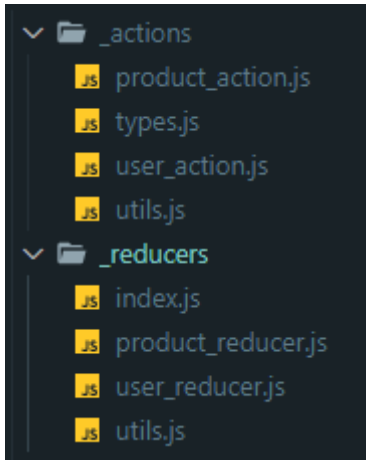

```
    },
    noWriter: Boolean,
    isRepresent: Boolean,
    comments: [
      {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Comment",
      },
    ],
  },
  { timestamps: true }
)

const Product = mongoose.model("Product", ProductSchema)

export default Product
```

```
if (dev) {
  mongoose
    .connect(process.env.MONGO_DEV_URI, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
      useFindAndModify: false,
      useCreateIndex: true,
    })
    .then(() => console.log("✅ Connected to Development MongoDB"))
    .catch((err) => console.log(`❌ ${err}`))
}
```

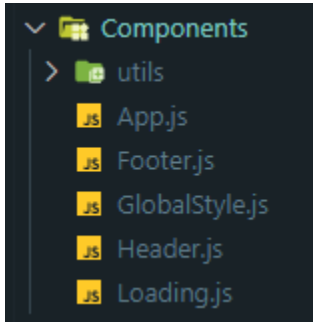
mongoose, mongoDB 사용

프론트엔드

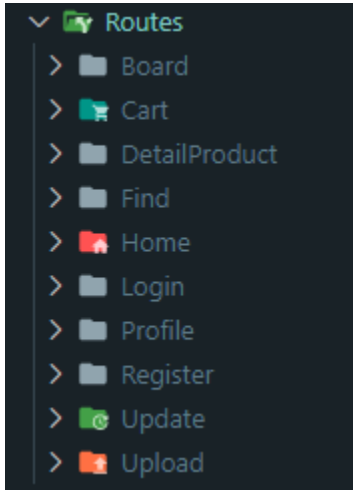




_actions, _reducers 리덕스



재사용 컴포넌트 및 네비게이션



page 요소

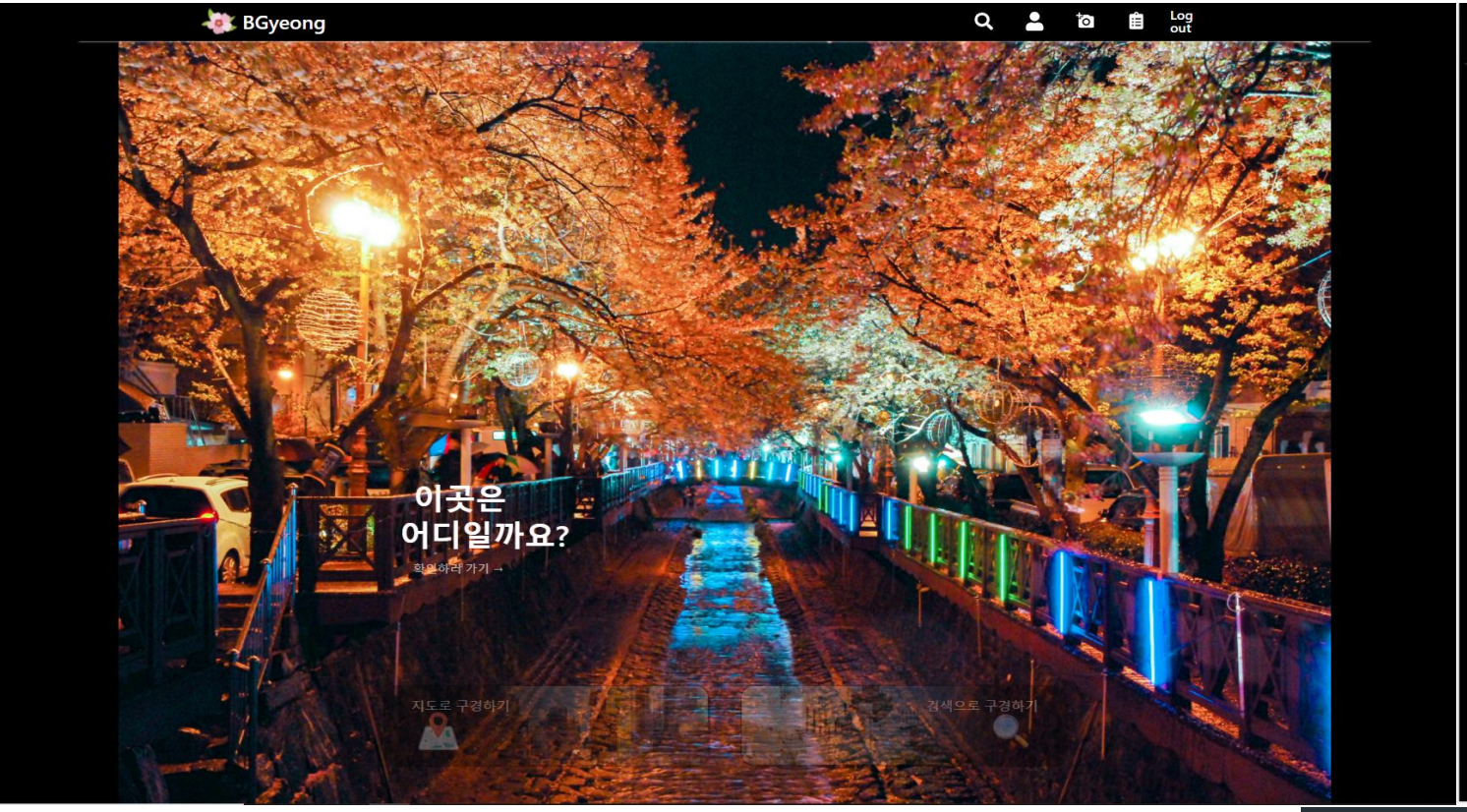
리덕스

```
export default (state = initialState, action) => {
  switch (action.type) {
    case GET_REP_PRODUCT:
    case GET_REP_PRODUCT_SUCCESS:
    case GET_REP_PRODUCT_FAILURE:
      return handleAsyncActions(GET_REP_PRODUCT, REP_PRODUCT)(state, action)
    case GET_PRODUCTS:
    case GET_PRODUCTS_SUCCESS:
    case GET_PRODUCTS_FAILURE:
      return handleAsyncActions(GET_PRODUCTS, PRODUCTS)(state, action)
export const asyncThunk = (type, request, key) => {
  const [SUCCESS, FAILURE] = [`${type}_SUCCESS`, `${type}_FAILURE`]
  return (param) => async (dispatch) => {
    dispatch({ type })
    try {
      const { data } = await request(param)
      dispatch({
        type: SUCCESS,
        payload: data[key], //키는 데이터 종류 구분 (product, products, user)
      })
    } catch (error) {
      dispatch({
        type: FAILURE,
        error,
      })
    }
  }
}
```

```
export const handleAsyncActions = (type, key) => {
  const [SUCCESS, ERROR] = [`${type}_SUCCESS`, `${type}_ERROR`]
  return (state, action) => {
    switch (action.type) {
      case type:
        return {
          ...state,
          ...reducerUtils.loading(),
        }
      case SUCCESS:
        return {
          ...state,
          ...reducerUtils.success(action.payload, key),
        }
      case ERROR:
        return {
          ...state,
          ...reducerUtils.success(action.error),
        }
      default:
        return state
    }
  }
}
```

redux thunk 사용

반응형




```
"react-responsive": "^8.2.0",
"react-reveal": "^1.2.2",
```

```
import { useMediaQuery } from "react-responsive"
```

```
const theme = {
  mobile: `(max-width: ${size.mobile})`,
  tablet: `(max-width: ${size.tablet})`,
  minTablet: `(min-width: ${size.tablet})`,
  laptop: `(max-width: ${size.laptop})`,
  desktop: `(min-width: ${size.desktop})`,
}
```

responsive를 이용한 media query 반응형 적용

읽어 주셔서 감사합니다. 최대한 보여드리고 싶은거 위주로 간추렸습니다.

직접 웹을 만들고 배포해보니 tdd를 이용한 코드 정리가 왜 중요한지,
docker를 이용하지 않고 운영하는게

얼마나 귀찮은지 등을 느꼈습니다. 무엇보다 협업하면서 현장에서 배우고 싶습니다.

비록 신입이고 무경력이지만 부족한 만큼 주제를 알아 정말 성실하게
일하고 배우고 적응하고 녹아들겠습니다.

감사합니다.