PROJECT-STRUCTURE

- PROJECT-STRUCTURE 개발 관련 내용을 기술한다.

이력

- 2022.07.22
  - 최초 등록
  - 개발프로세스, 소스코드관리 방법
- 2022.07.26
  - Node 버전관리 추가
- 2022.08.16
  - Node 버전 업데이트
- 2022.08.20
  - 프로젝트명 변경 (PROJECT-INIT > PROJECT-STRUCTURE)

구성

- 환경
  - windows 10, mac
  - visual studio code
  - node v16.17.0
  - npm v8.15.0
- 라이브러리
  - next / 12.2.2
  - react / 18.2.0
  - react-dom / 18.2.0

실행

- Visual Studio Code 실행
- npm install

로컬환경

- serve

개발환경

- serve-development

운영환경

- serve-production

배포

- Visual Studio Code 실행
- npm install
- npm run build

CSR (Client Side Rendering)

로컬환경

- build-export

개발환경

- build-export-development

운영환경

- build-export-production

SSR (Server Side Rendering)

로컬환경

- build
- start-pm2

개발환경

- build-development
- start-pm2-development

운영환경

- build-production
- start-pm2-production

Node 버전관리

NVM(Node Version Manager)은 말 그대로 Node.js의 버전을 관리하는 도구입니다. 협업, 여러가지 프로젝트를 동시에 진행해야 할때와 라이브러리 버전 호환 문제에 유용하게 사용할 수 있습니다.

1. nvm 설치 mac : brew install nvm windows : windows-nvm
2. `.nvmrc` 생성 project root 에 `.nvmrc` 생성후 Node Version 을 기입 한다.

```
// .nvmrc
{Node Version}
```

3. nvm use `.nvmrc` 에 명시된 Node Version 으로 변경 된다.

주석

- TODO : 좀더 최적화시키고 리팩토링시킬 수 있을만한 구석이 있을때. 미래에 뭔가 의미있는 작업을 더 해야 할 필요성을 느낄때.
- FIXME : 문제가 있는것이 확실하지만, 그걸 지금 당장 그것을 수정할 필요는 없을 때.
- XXX : 해당 부분에 대해서는 더 생각해볼 필요성이 있을 때. 또는 해당 부분에 질문이 생길 때. 또는 코드에서 문제가 일어날만한 부분을 강조 표기할때. 완벽하게 정확히 구현되지 않은 부분이 있을 때. 나중에 고쳐야만하는 부분일 때.

개발

구조

```
src
├── folder
│   ├── folder
│   └── folder
└── folder
```

개발 프로세스

- 이슈 단계 issues - boards(open, todo, doing, merged develop, merged master, close)
  - 'open' 단계에 앞으로 진행 될 이슈 생성(마일스톤, 작업자, 일정, 내용)
  - 스프린트 단위로 한 주가 시작되기 전 'open' -> 'todo' 상태 변경하여 해당 주의 스프린트가 시작 됨을 알림
  - 해당 작업자는 이슈 상태 'doing' 변경하여 이슈 해결 시작을 알림
  - /develop 브랜치를 기반으로 /feature/이슈번호 브랜치 생성하여 개발 시작
  - 개발 완료 후 /feature/이슈번호에 commit/push 후 merge request 생성
  - /feature/이슈번호 맞는 merge request 를 해당 팀원은 모두 코드를 확인하고 변경사항이 있다면 소스 위치에 수정내용 작성, 없다면 '확인 완료' 커맨트 작성
  - 작업자는 merge request 의 모든 커맨트를 확인하고 변경 작업 후에 완료했다는 커맨트를 등록, 'mark as resolved' 선택, 만약 '확인 완료' 만 있다면 'mark as resolved' 선택하고 병합
  - 변경요청 한 팀원은 변경 된 내용을 확인하고 작업자의 커맨트에 'mark as resolved'
  - 모든 작업이 완료되었다면 /develop 브랜치로 병합
  - 해당 프로젝트의 빌드전(배포서버에서) slack의 해당 채널에 배포 시작을 모든 팀원에게 알리고, 해당 프로젝트 deploy를 시작
  - deploy가 모두 완료되면 slack의 해당 채널에 배포 완료를 모든 팀원에게 알림
  - 해당 테스크 상태 'merged develop' 변경
  - 테스트 진행
  - 위 내용은 스프린트(단위:1주) 단위로 이루어 지며 모든 이슈에 대해서 위와 같이 동일하게 반복

소스 버전 관리

- 해당 브랜치 구조는 git-flow 정책을 따름
- 브랜치 구조 feature - develop - release - tag - master - hotfix
    - feature : 테스크 단위 브랜치
    - develop : 전체 개발 단위 브랜치
    - release : 릴리즈 단위 브랜치
    - master : 전체 운영 단위 브랜치
    - hotfix : 전체운영 핫픽스 단위 브랜치