GIT 컨벤션

branch 규칙

- master docs, 자동 배포
- develop 완성된 기능 단위로 머지
- develop 하단 기능 브랜치

[브랜치명]

feature/브랜치명

[커밋메시지]

Merge branch 'login' into 'develop'

commit message 작성 규칙

```
术 feat: Summarize changes in around 50 characters or less
```

This is a body part. Please describe the details of commit.

JIRA ISSUE: [S05P1A5075-39]

- 제목의 길이는 50글자를 넘기지 않는다
- 본문을 작성할 때 한 줄에 72글자 넘기지 않는다
- 마침표를 사용하지 않는다
- 과거형을 사용하지 않는다
- 커밋 메시지는 영어로 작성한다

git commitizen 사용하기 Comment

1. git-cz 설치

• 아래 명령어를 입력해서 전역에 설치를 해준다.

```
npm install -g git-cz
```

2. git-cz custom 하기

- 팀원들과 합의한 컨벤션을 changelog.config.js 파일에 포함시킨다.
 - 。 이 파일은 프로젝트 폴더 가장 상단에 위치시킨다(.git 파일과 동일한 위치!)

```
module.exports = {
  "disableEmoji": false,
  "format": "{emoji}{type}{scope}: {subject}",
  "closedIssueMessage": 'JIRA ISSUE: ',
  "closedIssuePrefix": '',
  "list": [
    "feat",
    "fix",
    "remove",
    "docs",
    "style"
    "chore",
    "refactor",
  ],
  "maxMessageLength": 64,
  "minMessageLength": 3,
  "questions": [
    "type",
    "subject",
    "body",
    "issues"
  ],
  "types": {
     "description": "새로운 기능 추가",
      "emoji": "☆",
      "value": "feat"
    },
    "fix": {
      "description": "버그 및 코드 수정",
      "emoji": "\\",
      "value": "fix"
    "remove": {
      "description": "코드 제거",
      "emoji": "✓",
      "value": "remove"
    },
    "docs": {
      "description": "문서 작업",
      "emoji": "<u></u>",
```

GIT 컨벤션 2

```
"value": "docs"
   },
   "style": {
     "description": "코드에 영향을 주지 않는 디자인 및 포맷 변경사항",
     "emoji": "😍",
     "value": "style"
   },
    "chore": {
     "description": "빌드 관련 파일 수정 및 설정 변경",
     "emoji": " ] ",
     "value": "chore"
   },
   "refactor": {
     "description": "성능 개선 및 리팩토링",
     "emoji": "@",
     "value": "refactor"
   }
 }
}
```

3. commit 과정

- git add 명령어를 통해 파일을 스테이지에 추가한다.
- git commit 대신 git cz 명령어를 사용한다.
- git cz 를 입력하고 엔터를 치면 아래 그림과 같이 타입을 선택할 수 있다.

```
multicampus@DESKTOP-KVCQHCD MINGW64 /c/Users/self-study/git-cz-test (main)
$ git cz
? Select the type of change that you're committing: (Use arrow keys or type to search)

> ☆ feat: 새로운 기능 추가

글 docs: 문서 작업

• style: 코드에 영향을 주지 않는 디자인 및 포맷 변경사항

□ chore: 빌드 관련 파일 수정 및 설정 변경

• fix: 버그 수정

□ refactor: 성능 개선 및 리팩토링
```

• 타입을 선택하면 commit 타이틀을 입력한다.

• 해당 commit에 대한 세부 설명이 필요하다면 작성한다.

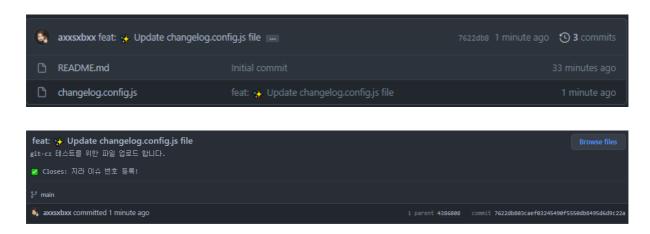
GIT 컨벤션 3

。 설명이 없다면 enter를 치면 다음 항목으로 넘어간다.

• commit에 할당된 지라 이슈 번호가 있다면 입력해준다.

• 이후 git push 명령어를 실행한다.

4. 결과 확인



GIT 컨벤션 4