**Coding Standard**

**&**

**Repository Management**

**&**

**Review Process**

시대평

팀명:SoGoing

팀원:2020920032 석우진

2020920039 이강민

2020920055정구홍

2020920059조종빈

2021120009김고은

2022920027박상윤

변 경 이 력

|  |  |  |  |
| --- | --- | --- | --- |
| 버전 | 일자 | 변경 내역 | 작 성 자 |
| 1.0 | 2024.11.18 | 초안 작성 | 박상윤 |
| 1.1 | 2024.11.18 | Repository Management, Review Process 내용 추가 | 이강민 |
| 1.2 | 2024.11.18 | Coding Standard 내용 추가 | 박상윤 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Coding Standard](#_Toc178012895)

[1.1 Naming Convention](#_Toc178012896)

[1.2 Code Quality & Formatting](#_Toc178012897)

[2. Repository Management & Review Process](#_Toc178012900)

[2.1 Git Convention](#_Toc178012898)

**Coding Standard**

**1. Naming Convention**

**기본 명명 규칙**

* Lower Cammel case를 사용합니다.
  + 첫 글자는 소문자로 시작합니다.
  + 첫 글자를 제외한 각 단어는 대문자로 시작합니다.

1. **Class**
   * 명사 또는 명사구이며 대문자로 시작합니다.
2. **Method**
   * 동사구나 값을 설명하는 명사구이며 소문자로 시작합니다.
   * Field에 접근하려는 함수는 get/set으로 시작합니다.
   * 조건을 묻는 Bool형 함수의 이름은 is로 시작합니다.
3. **Field**
   * 명사
   * 소문자로 시작
   * 상수는 모두 대문자로 적으며 Snake case를 사용합니다.
4. **Package**
   * 명사
   * 소문자
5. **Component**
   * **Pascal Case 을 사용합니다.**
   * 첫글자는 대문자로 시작하며, 각 단어는 대문자로 구분됩니다.
   * 컴포넌트 변수명 뿐만 아니라, 이를 담고 있는 파일의 이름도 동일하게 적용합니다.

**2. Code Quality & Formatting**

**아래 tool들을 사용하여, CI(Continuous Integration) 단계에서 코드 품질 및 스타일 규칙을 자동으로 검사하고 규칙을 위반한 경우 빌드 프로세스를 중단합니다.**

**Frontend - ESLint Plugin, Prettier 사용**

**ESLint-Plugin-React**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

1. **files: ['\*\*/\*.{js,jsx,mjs,cjs,ts,tsx}']**
   * **설명**: ESLint 규칙이 적용될 파일 확장자를 모든 디렉토리 내의 .js, .jsx, .mjs, .cjs, .ts, .tsx로 정의합니다.
2. **languageOptions.parserOptions.ecmaFeatures.jsx = true**
   * **설명**: JSX 문법을 활성화합니다. 이를 통해 ESLint가 JSX 문법을 이해하고 규칙을 적용할 수 있습니다.
3. **globals: { ...globals.browser }**
   * **설명**: 브라우저 환경의 전역 변수를 정의하여 브라우저 환경에서 사용 가능한 전역 변수를 참조할 때 경고를 방지합니다.
     + **globals.browser**: 브라우저에서 제공하는 전역 객체를 자동으로 포함.
4. **rules**
   * **설명**: ESLint에서 적용할 규칙을 정의합니다.
   * **react/jsx-uses-react**
     + **설명**: JSX에서 React 변수를 사용했는지 확인합니다.
   * **react/jsx-uses-vars**
     + **설명**: JSX에서 선언된 변수가 사용되었는지 확인하고, 사용되지 않는 JSX 변수에 대해 경고를 표시하는 것을 방지합니다.

**Prettier**

* **printWidth = 80**
  + **설명**: 한 줄의 최대 길이를 80으로 설정합니다.
* **tabWidth = 2**
  + **설명**: 들여쓰기에 사용되는 공백의 크기를 2로 설정합니다.
* **singleQuote = true**
  + **설명**: 문자열을 나타낼 때 작은따옴표(')를 사용합니다.
* **trailingComma = all**
  + **설명**: 객체나 배열에서 마지막 항목 뒤에 쉼표를 추가하도록 설정합니다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

* **bracketSpacing = true**
  + **설명**: 객체 리터럴에서 중괄호 안의 공백을 추가하도록 설정합니다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

* **semi = true**
  + **설명**: 문장의 끝에 세미콜론(;)을 추가하도록 설정합니다.

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

* **useTabs = false**
  + **설명**: 들여쓰기에 탭 대신 공백을 사용하도록 설정합니다.
* **endOfLine = “lf”**
  + **설명**: 줄바꿈 방식(Line Ending)을 LF(Line Feed)로 설정합니다.

**Backend - Kotlin Formatter 사용**

1. **maxWidth = 100**
   * **설명**: 코드를 한 줄로 작성할 때 허용되는 글자수를 100으로 설정합니다.
2. **blockIndent = 2**
   * **설명**: 새 블록이 열릴 때 사용하는 들여쓰기 크기(공백의 개수)를 2로 설정합니다.
3. **continuationIndent = 4**
   * **설명**: 한 줄이 너무 길어 줄 바꿈이 필요한 경우, 그 줄의 들여쓰기 크기(공백의 개수)를 4로 설정합니다.

**Dropbox style**

* **blockIndent = 4**
  + **설명**: 새 블록이 열릴 때 사용하는 들여쓰기 크기(공백의 개수)를 4로 설정합니다.
* **continuationIndent = 4**
  + **설명**: 줄이 길어서 끊어진 경우, 새 줄의 들여쓰기 크기(공백의 개수)를 4로 설정합니다.

**Repository Management & Review Process**

**Git 컨벤션**

**브랜치 전략 - Github Flow**

* 브랜치 명: 영어로 만들어야 합니다.

**Main 브랜치**

* Main 브랜치는 언제든지 배포가 가능한 안정적인 버전입니다.
* Main 브랜치는 빌드가 가능하며 모든 테스트를 통과해야 합니다.
* Main 브랜치가 Github에서 Default 브랜치가 됩니다.

**Feature 브랜치**

* Feature 브랜치는 새로운 기능을 개발하는 역할입니다.
* Feature 브랜치는 항상 Main 브랜치에서 만듭니다.
* 버그 수정도 Feature 브랜치를 생성하여 진행합니다.
* Feature 브랜치의 이름은 개발할 기능의 {기능명}/{feature-name} 으로 생성합니다.
  + ex) feat/env-setting (영어로 작성)

**Git Commit 메세지**

* Commit 메세지는 무엇을 작업했는지 알아보기 쉽게 작성합니다.
* {TYPE}: {SUBJECT}형식으로 작성합니다.
  + ex) feat: add login api
* TYPE
  + feat : 새로운 기능 추가
  + fix : 기능에 대한 버그 수정
  + build : 빌드 관련 수정
  + chore : 패키지 매니저 수정, 그 외 기타 수정
    - ex) .gitignore
  + ci : CI 관련 설정 수정
  + docs : 문서(주석) 수정
  + style : 코드 스타일, 포맷팅에 대한 수정
  + refactor : 기능의 변화가 아닌 코드 리팩터링
    - ex) 변수 이름 변경
  + test : 테스트 코드 추가/수정
  + release : 버전 릴리즈

**Github PR/Issue 규칙**

* 네이밍: **[기능명] 제목**
  + ex) [Feat] 일기 작성 기능 구현
* PR 리뷰 최소 1명
  + PR은 팀원들이 이해할 수 있게끔 자세하게 설명합니다.
  + 이미지는 선택 사항