JBIG Git 전략

이전 상황

따로 branch 생성 안하고 모든 작업을 main에서 처리, 코드 결합도가 높아져 유지보수에 문제가 생길 수 있음.

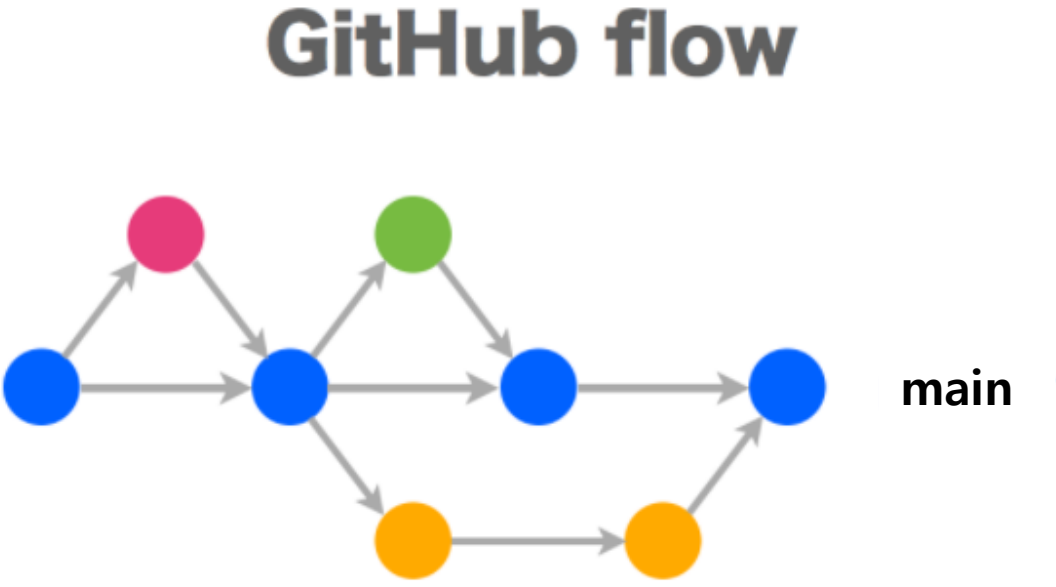
→ Github flow 전략 도입

Branch 전략

Github flow

1. Master(main) 브랜치는 언제든 배포 가능한 상태, 따라서 merge 이전 충분한 테스트를 거쳐야 함
2. 기능 작업은 항상 새로운 브랜치에서 시작할 것
3. Merge를 앞뒀거나 피드백이 필요할 때 pull request 생성(이때 github action이 테스팅 함)
4. CI를 통과하고 기능이 완성됐다면 master(main)으로 merge
5. 만약 master(main)으로 merge 되고 push 되었다면 즉시 배포 가능해야 함

선정 이유: CI/CD 도입으로 현재 프로젝트에 branch 전략 중 가장 적합할 것으로 예상



커밋 규칙:

항목	규칙
커밋 형식	[타입]: 변경 내용 # 이슈 번호(없을 시 생략)
타입	Feat, fix, refactor, docs, test 등
예시 메시지	Feat: 로그인 API 추가

Pull Request 규칙:

항목	규칙
PR 생성 시점	개발 완료 → 테스트 통과 후 push
PR 형식	[타입]: 작업 내용 # 이슈 번호(없을 시 생략)
PR 설명	변경 이유, 이슈 번호
코드 리뷰	최소 1명 이상 승인
Merge 방식	Rebase merge

Main에 문제가 생기면 hotfix 브랜치 생성 후 배포

Main에 태그 붙히기, 10/18일 기준, 가장 최근 커밋을 기준으로 v1.0.0으로 명명

Github Actions

발동 조건

Push, pull request

과정

1. PostgreSQL 15 DB Container를 이용해 Django 테스팅 DB로 활용
2. 환경: Python 3.12 버전
3. 프로젝트 패키지(requirements.txt) 설치
4. Lint check
   - ruff: 코드 스타일, 문법 위반 체크
   - black: 코드 포맷 일관성 검사
   - isort: import문 정렬 규칙 위반 검사
5. Type Checking – 10/18일자 해당 단계에서 오류 발생, 수정중

6. Security Check
   - pip-audit: 보안 취약점이 있는 패키지 탐지
   - bandit: 하드코딩된 비밀번호, 취약한 코드 패턴 탐지
7. Django 자체 점검
8. 마이그레이션 및 정적 파일 수집
9. 단위 테스트 + 커버리지 측정
   - pytest: Django 테스트 실행
   - coverage: 코드 커버리지 측정 후 XML 리포트 생성
10. 생성된 coverage.xml 파일을 github actions artifact로 업로드
11. Deploy 서버로 전송
12. Deploy 서버에서 systemctl 재시작

배민준: 작성자가 테스팅을 배우지 않아 GPT로 테스팅 코드를 작성했으니 코드에 문제가 있다면 연락 바랍니다.