02 효과적이거나 개선할 수 있는 부분 제안

고블로EDMS 개발 계획을 살펴보니, 꽤 체계적으로 단계별 개발 로드맵을 수립하셨네요! 🚀
각 단계별 핵심 기능과 기술 스택도 잘 정의되어 있어서, 실행만 제대로 되면 강력한 AI 기반 문서 관리 시스템(EDMS)이 탄생할 것 같아요.

💡 가장 효과적으로 구현하는 방법 검토

인터넷에서 최신 트렌드와 도구를 검토해 보면서, **더 효과적이거나 개선할 수 있는 부분**을 몇 가지 제안드릴게요.

✅ 1단계: 개발환경 구축

⚡ 최적화 포인트

- Docker Compose 스택이 (PostgreSQL, Qdrant, Ollama, N8N)으로 잡혀 있는데, **Dev 환경과 Prod 환경을 분리**하면 더 안정적일 수 있음.
  → `docker-compose.override.yml` 을 활용해 개발 & 운영 환경을 분리 가능
- FastAPI + PostgreSQL + Swagger 조합은 검증된 스택! 하지만 **초기 API 스키마 설계를 신중하게** 해야 이후 확장이 수월함.
- Flutter 상태 관리 (Riverpod vs Bloc) → **앱의 복잡성을 고려할 때 Bloc이 적절할 가능성이 큼**
  → 다만, 단순 CRUD 성향이 강하면 Riverpod도 괜찮음.

🔍 추가 검토할 사항

✔ Qdrant vs Weaviate vs Pinecone

- Qdrant를 선택하셨는데, 의미 기반 검색(벡터 DB)에서는 Weaviate나 Pinecone도 고려 가능
- Qdrant는 Rust 기반이라 성능이 좋지만, Weaviate는 GraphQL 지원이 강점
  ✔ 문서 업로드 후 변환
- PDF, DOCX → Markdown 변환 시 `pandoc` 외에 **unstructured.io 라이브러리**도 검토 가능

✅ 2단계: 백엔드 핵심 기능 개발

⚡ 최적화 포인트

- **파일 업로드 후 변환 속도 개선 필요**
  → 비동기 작업을 위해 **Celery + Redis** 도입 고려 (FastAPI와 궁합 좋음)
- AI 요약을 Ollama(LLaMA)로 진행하는데, 문서 요약엔 **T5 모델 계열이 강력**
  → Ollama가 아니라 **LangChain + OpenAI API** 조합도 성능 대비 나쁘지 않음.

- API 문서화는 Swagger를 사용하지만, **Redoc을 병행하면 가독성이 더 좋음.**

🔍 **추가 검토할 사항**
✔ **PostgreSQL Full-Text Search 최적화**

- GIN 인덱스 활용하면 속도 개선 가능
  ✔ **OCR 기반 검색 (Tesseract vs AWS Textract vs Google Vision)**
- 자체 구축이면 Tesseract, 클라우드 기반이면 Textract가 OCR 성능이 더 좋음

✅ 3단계: 프론트엔드 주요 기능 개발

⚡ **최적화 포인트**

- Markdown 편집기: `zefyr` 패키지가 있지만, **새로운** `super_editor`**가 더 강력**
- AI 요약 UI → FastAPI WebSocket을 활용하면 실시간 피드백 가능
- 버전 관리: Git 방식이 아닌 **DVC(Data Version Control) 적용 가능**

🔍 **추가 검토할 사항**
✔ **Diff Highlighting (버전 비교 UI) →** `diff-match-patch` **라이브러리 추천**
✔ **Markdown 실시간 미리보기 → WebView 기반 OR 자체 렌더링? (성능 비교 필요)**

✅ 4단계: 기능 고도화 및 테스트

⚡ **최적화 포인트**

- API 성능 개선: Redis 캐싱, DB 인덱싱 필수 적용
- 검색 최적화: 의미 기반 검색 속도 개선을 위해 **ONNX 기반 벡터 모델 고려 가능**

🔍 **추가 검토할 사항**
✔ **Flutter UI 테스트 자동화 →** `integration_test` **패키지 활용**
✔ **로그 분석 → Grafana + Loki 조합 추천**

✅ 5단계: 배포 및 운영

⚡ **최적화 포인트**

- AWS, GCP, DigitalOcean 중 선택 시 **CDN 적용 가능성 검토 (Cloudflare or AWS CloudFront)**
- 서버 로그 분석: Prometheus + Grafana 활용 가능

🔍 **추가 검토할 사항**

✔ **배포 자동화 → GitHub Actions / GitLab CI/CD 스크립트 정리 필요**

✔ **보안 점검 → JWT 토큰 만료 시간 & 세션 관리 정책 강화 필요**

🎯 결론: 이렇게 하면 더 효과적!

✔ **개발환경 최적화** → Docker Dev/Prod 분리 & FastAPI + Celery

✔ **벡터 검색 개선** → Qdrant vs Weaviate 성능 비교 후 선택

✔ **AI 요약 최적화** → Ollama 대신 LangChain + OpenAI 조합 검토

✔ **Markdown 편집기** → `super_editor`가 최신 트렌드

✔ **성능 & 보안** → DB 인덱싱 + 캐싱, 로그 분석 강화

이제 **구체적인 구현 방향**을 정하고, 필요한 도구/라이브러리 선정 후 작업을 시작하면 될 것 같아요! 😊

어떤 부분부터 먼저 적용할지 고민되시면 말씀 주세요. 🚀