Prisma 란?

1. Prisma는 무엇인가?

Prisma는 ORM(Object Relational Mapping) 이다.

ORM의 역할:

- 데이터베이스 테이블을 코드의 객체처럼 다룰 수 있게 해줌
- SQL을 직접 많이 작성하지 않아도 됨
- Node.js / TypeScript 환경에서 DB 접근을 단순화

예시:

```
await prisma.user.findMany();
```

➡ 내부적으로는 SQL로 변환되어 실행됨

---

2. schema.prisma = DB 설계도

`schema.prisma` 파일은 다음 역할을 동시에 수행한다:

- 데이터베이스 구조 정의 (테이블, 컬럼)
- 관계 정의 (1\:N, N\:M)
- 제약조건 정의 (PK, UNIQUE, FK)

즉, **ERD + DDL 역할을 함께 수행**

---

3. 현재 프로젝트 기준 모델 해석

User 모델

- id : 기본키 (자동 증가)
- email : 유니크
- username : 유니크
- password : 비밀번호
- createdAt / updatedAt : 생성/수정 시간 자동 관리
- memos : Memo와 1\:N 관계

Memo 모델

- id : 기본키
- content : 메모 내용
- userId : User를 참조하는 외래키
- user : User와의 관계

- User 삭제 시 Memo도 함께 삭제 (Cascade)

---

4. Prisma 전체 동작 흐름

```
schema.prisma (설계)
       ↓
prisma migrate / db push
       ↓
실제 DB (SQLite / Postgres / MySQL)
       ↓
prisma generate
       ↓
Prisma Client (JS에서 사용하는 ORM 객체)
```

역할 정리:

- schema.prisma → DB 설계서
- DB → 실제 데이터 저장소
- Prisma Client → 코드에서 사용하는 DB 접근 객체

---

5. server.js에서 Prisma 코드 의미

```
const { PrismaClient } = require('@prisma/client');
const prisma = new PrismaClient();
```

의미:

- schema.prisma 기준으로 생성된 ORM 클라이언트를 로드
- 이후 `prisma.user`, `prisma.memo` 형태로 DB 접근 가능

---

6. Prisma 6.x 사용에 대한 판단

- Prisma 7.x: 변경사항 많고 환경에 따라 오류 빈번
- Windows + 최신 Node 조합에서 특히 이슈 발생 가능
- 학습 / 사이드 프로젝트 / 데모 목적이라면 **6.x가 안정적**

---

7. 핵심 요약

- Prisma는 ORM이다
- schema.prisma는 DB 설계 그 자체다
- migrate는 DB 반영
- generate는 JS용 ORM 생성

- Prisma Client로 코드에서 DB를 다룬다