MyApp Backend

FastAPI 기반의 사용자 초대 및 회원가입 시스템 백엔드 서버입니다.

프로젝트 개요

이 프로젝트는 초대코드 기반 회원가입 시스템을 제공하는 RESTful API 서버입니다. Prisma ORM을 사용하여 PostgreSQL 데이터베이스와 연동하며, bcrypt를 통한 안전한 비밀번호 해시 처리를 지원합니다.

기술 스택

- **프레임워크**: FastAPI 0.115.9
- **ASGI 서버**: Uvicorn 0.34.2
- **ORM**: Prisma (Python Client) 0.15.0
- **데이터베이스**: PostgreSQL (Supabase)
- **비밀번호 암호화**: passlib + bcrypt
- **데이터 검증**: Pydantic 2.12.3

아키텍처

하이브리드 스택

- **FastAPI (Python)**: HTTP 서버 및 비즈니스 로직
- **Prisma (Node.js)**: 데이터베이스 스키마 관리 및 마이그레이션
- **Prisma Client Python**: Python 코드에서 데이터베이스 접근

데이터베이스 모델

1. Invitation (초대코드)

```
model Invitation {
  id        Int      @id @default(autoincrement())
  code      String   @unique
  is_used   Boolean  @default(false)
  used_by   Int?
}
```

- 초대코드 관리 및 사용 여부 추적

2. RegisterSession (회원가입 세션)

```
model RegisterSession {
  sessionId     String   @id
  invitationId  Int
  name          String?
  password      String?
  birthday      String?
  phone         String?
}
```

- 다단계 회원가입 과정의 임시 데이터 저장
- 세션 ID로 각 단계를 추적

3. User (사용자)

```
model User {
  id            Int      @id @default(autoincrement())
  name          String
  password      String
  birthday      String
  phone         String
  invitationId  Int
}
```

- 최종 사용자 정보 저장
- 비밀번호는 bcrypt로 해시 처리되어 저장

API 엔드포인트

1. 초대코드 검증

```
POST /auth/invitation/verify
Content-Type: application/json

{
  "invitation_code": "string"
}
```

성공 응답:

```
{
  "valid": true,
```

```
  "sessionId": "hex-string"
}
```

2. 이름 및 비밀번호 저장

```
PUT /auth/register/name
Content-Type: application/json

{
  "session_id": "string",
  "name": "string",
  "password": "string"
}
```

기능:

- 사용자 이름 저장
- 비밀번호를 bcrypt로 해시 처리 후 저장

3. 생년월일 저장

```
PUT /auth/register/birthday
Content-Type: application/json

{
  "session_id": "string",
  "birthday": "string"
}
```

4. 휴대폰 번호 저장 및 사용자 생성

```
PUT /auth/register/phone
Content-Type: application/json

{
  "session_id": "string",
  "phone": "string"
}
```

기능:

- 휴대폰 번호 저장
- 이전 단계 완료 검증 (name, password, birthday)

- 최종 User 레코드 생성
- Invitation을 사용됨으로 표시

성공 응답:

```
{
  "ok": true,
  "userId": 1
}
```

회원가입 플로우

```
1. POST /auth/invitation/verify
   ↓ (초대코드 검증, RegisterSession 생성)

2. PUT /auth/register/name
   ↓ (이름 + 비밀번호 저장 및 해시 처리)

3. PUT /auth/register/birthday
   ↓ (생년월일 저장)

4. PUT /auth/register/phone
   ↓ (휴대폰 저장, User 생성, Invitation 사용 처리)

✓ 회원가입 완료
```

설치 및 실행

1. 필수 요구사항

- Python 3.12+
- Node.js (Prisma CLI 사용)
- PostgreSQL 데이터베이스

2. Python 가상환경 설정

```
# 가상환경 생성
python -m venv venv

# 가상환경 활성화 (macOS/Linux)
source venv/bin/activate
```

```
# 가상환경 활성화 (Windows)
venv\Scripts\activate
```

3. 의존성 설치

```
# Python 패키지 설치
pip install -r requirements.txt

# Prisma CLI 설치
npm install -D prisma@5.17.0
```

4. 환경 변수 설정

`.env` 파일을 생성하고 데이터베이스 연결 정보를 입력합니다:

```
# Supabase Pooler URL (런타임용)
DATABASE_URL="postgresql://username:password@host:5432/postgres?
sslmode=require"
```

중요: 비밀번호에 특수문자(`@`, `#` 등)가 포함된 경우 URL 인코딩이 필요합니다:

- `@` → `%40`
- `#` → `%23`

5. 데이터베이스 마이그레이션

```
# Prisma 클라이언트 생성
npx prisma generate

# 데이터베이스 스키마 푸시
npx prisma db push

# 또는 마이그레이션 실행 (Direct URL 필요)
npx prisma migrate dev
```

6. 서버 실행

```
# 개발 서버 실행 (자동 리로드)
uvicorn main:app --reload

# 프로덕션 서버 실행
uvicorn main:app --host 0.0.0.0 --port 8000
```

서버가 시작되면 `http://127.0.0.1:8000`에서 접근할 수 있습니다.

개발 명령어

Prisma 관련

```
# Prisma Python 클라이언트 재생성
npx prisma generate

# 데이터베이스 스키마 푸시 (마이그레이션 없이)
npx prisma db push

# 마이그레이션 생성 및 실행
npx prisma migrate dev --name migration_name

# Prisma Studio로 데이터베이스 GUI 열기
npx prisma studio
```

서버 관련

```
# 개발 서버 (자동 리로드)
uvicorn main:app --reload

# 특정 포트로 서버 실행
uvicorn main:app --reload --port 8001

# 외부 접근 허용
uvicorn main:app --host 0.0.0.0 --port 8000
```

API 테스트

서버의 API를 테스트하는 다양한 방법을 제공합니다.

1. FastAPI 자동 생성 문서 (권장)

FastAPI는 Swagger UI와 ReDoc을 자동으로 제공합니다:

```
# 서버 실행 후 브라우저에서 접속
```

- **Swagger UI**: http://127.0.0.1:8000/docs
  - 인터랙티브 API 테스트 가능
  - "Try it out" 버튼으로 실제 요청 전송
  - 실시간 응답 확인

- **ReDoc**: http://127.0.0.1:8000/redoc
  - 깔끔한 API 문서 (읽기 전용)

2. Python 테스트 스크립트

프로젝트에 포함된 자동화 테스트 스크립트를 사용할 수 있습니다:

```
python test_api.py
```

제공되는 테스트:

- 전체 회원가입 플로우 테스트
- 개별 엔드포인트 테스트
- 에러 케이스 검증

테스트 초대코드:

- 스크립트는 `TEST001` ~ `TEST005` 초대코드를 자동으로 사용
- 첫 번째 사용 가능한 코드부터 순서대로 선택
- 테스트 전에 이 코드들이 데이터베이스에 존재해야 함 (아래 "테스트 데이터 준비" 참조)

테스트 결과 예시:

```
🚀 회원가입 플로우 테스트 시작

1️⃣ 초대코드 검증 중...
✅ 세션 ID 획득: 932b5ec...

2️⃣ 이름 및 비밀번호 저장 중...
✅ 성공

3️⃣ 생년월일 저장 중...
✅ 성공

4️⃣ 휴대폰 번호 저장 및 사용자 생성 중...
🎉 회원가입 완료! User ID: 1
```

3. curl 명령어

터미널에서 직접 API를 호출할 수 있습니다:

```
# 초대코드 검증
curl -X POST "http://127.0.0.1:8000/auth/invitation/verify" \
```

```
  -H "Content-Type: application/json" \
  -d '{"invitation_code": "TEST001"}'

# 이름 + 비밀번호 저장
curl -X PUT "http://127.0.0.1:8000/auth/register/name" \
  -H "Content-Type: application/json" \
  -d '{
    "session_id": "your-session-id",
    "name": "홍길동",
    "password": "securePass123!"
  }'

# 생년월일 저장
curl -X PUT "http://127.0.0.1:8000/auth/register/birthday" \
  -H "Content-Type: application/json" \
  -d '{
    "session_id": "your-session-id",
    "birthday": "1990-01-01"
  }'

# 휴대폰 저장 + User 생성
curl -X PUT "http://127.0.0.1:8000/auth/register/phone" \
  -H "Content-Type: application/json" \
  -d '{
    "session_id": "your-session-id",
    "phone": "010-1234-5678"
  }'
```

4. 테스트 데이터 준비

테스트 전에 데이터베이스에 초대코드를 추가해야 합니다.

권장: 여러 개의 테스트 코드 생성

`test_api.py` 스크립트는 `TEST001` ~ `TEST005` 초대코드를 자동으로 사용합니다. 다음 명령어로 한 번에 생성하세요:

```
# 5개의 테스트 초대코드 생성 (권장)
psql $DATABASE_URL << 'EOF'
INSERT INTO "Invitation" (code, is_used) VALUES
  ('TEST001', false),
  ('TEST002', false),
  ('TEST003', false),
  ('TEST004', false),
  ('TEST005', false)
```

```
ON CONFLICT (code) DO UPDATE SET is_used = false, used_by = NULL;
EOF
```

방법 1: Prisma Studio (GUI)

```
npx prisma studio
```

1. 브라우저에서 http://localhost:5555 접속
2. `Invitation` 테이블 선택
3. "Add record" 클릭
4. `code`: "TEST001" (또는 "TEST002", "TEST003" 등), `is_used`: false 입력
5. Save
6. 필요한 만큼 반복

방법 2: SQL 직접 실행 (단일 코드)

```
# PostgreSQL 클라이언트로 단일 초대코드 삽입
psql $DATABASE_URL -c "INSERT INTO \"Invitation\" (code, is_used) VALUES
('TEST001', false);"
```

방법 3: Python 스크립트

```
import asyncio
from prisma import Prisma

async def create_test_invitations():
    db = Prisma()
    await db.connect()

    # 여러 개의 테스트 초대코드 생성
    codes = ["TEST001", "TEST002", "TEST003", "TEST004", "TEST005"]

    for code in codes:
        invitation = await db.invitation.create(
            data={"code": code, "is_used": False}
        )
        print(f"Created invitation: {invitation.code}")

    await db.disconnect()

asyncio.run(create_test_invitations())
```

💡 **참고:**

- `test_api.py` 는 `TEST001` 부터 순서대로 사용합니다
- 사용된 코드는 자동으로 `is_used=true` 로 변경됩니다
- 다시 테스트하려면 새로운 코드를 사용하거나 기존 코드를 리셋하세요:

```
# 초대코드 리셋
psql $DATABASE_URL -c "UPDATE \"Invitation\" SET is_used = false, used_by
= NULL WHERE code = 'TEST001';"
```

5. Prisma Studio로 데이터 확인

테스트 후 생성된 데이터를 확인할 수 있습니다:

```
npx prisma studio
```

확인 가능한 내용:

- `Invitation` : 초대코드 사용 여부
- `RegisterSession` : 진행 중인 회원가입 세션
- `User` : 생성된 사용자 정보 (비밀번호는 해시됨)

테스트 체크리스트

테스트 전 확인사항:

- [ ] 서버가 실행 중 (`uvicorn main:app --reload`)
- [ ] 데이터베이스 연결 정상
- [ ] 테스트용 초대코드 존재 (`Invitation` 테이블)
- [ ] Prisma 클라이언트 생성됨 (`npx prisma generate`)

보안 고려사항

비밀번호 보안

- **bcrypt 해시**: 모든 비밀번호는 bcrypt 알고리즘으로 해시 처리
- **단방향 암호화**: 저장된 해시로부터 원본 비밀번호 복원 불가능
- **Salt**: bcrypt가 자동으로 랜덤 salt 생성 및 적용

세션 관리

- **UUID 기반 세션 ID**: 예측 불가능한 세션 식별자 사용
- **단계별 검증**: 각 회원가입 단계에서 세션 유효성 검증

- 임시 데이터: RegisterSession은 회원가입 완료 전까지만 사용

개선 필요 사항

- [ ] RegisterSession 레코드 자동 정리 (만료 시간 설정)
- [ ] 회원가입 완료 후 세션 삭제
- [ ] Rate limiting 구현
- [ ] HTTPS 강제 적용 (프로덕션)

데이터베이스 연결 문제 해결

Supabase Pooler vs Direct 연결

- **Pooler URL** (`*.pooler.supabase.com`): FastAPI 런타임용, 짧은 연결에 최적화
- **Direct URL** (`db.*.supabase.co`): Prisma 마이그레이션용, 긴 세션 연결 필요

데이터베이스 일시 중지

Supabase 무료 플랜은 7일간 활동이 없으면 자동으로 데이터베이스를 일시 중지합니다.

해결 방법:

1. Supabase Dashboard 접속
2. 프로젝트 선택
3. "Resume" 버튼 클릭

프로젝트 구조

```
myapp-backend/
├── main.py                 # FastAPI 애플리케이션 메인 파일
├── test_api.py             # API 자동화 테스트 스크립트
├── prisma/
│   ├── schema.prisma       # Prisma 스키마 정의
│   └── migrations/         # 데이터베이스 마이그레이션 파일
├── .env                    # 환경 변수 (gitignore)
├── requirements.txt        # Python 의존성
├── package.json            # Node.js 의존성 (Prisma)
├── CLAUDE.md               # Claude Code 가이드
└── README.md               # 프로젝트 문서 (이 파일)
```

코드 품질

명명 규칙

- **Prisma 모델 접근**: 모든 모델은 소문자로 통일 (`db.registersession` , `db.invitation` , `db.user`)
- **API 요청 파라미터**: Python 컨벤션에 따라 snake_case 사용 (`session_id` , `invitation_code` 등)
- **Pydantic 모델**: BaseModel 서브클래스는 PascalCase 사용 (`InvitationReq` , `NameReq` 등)

라이선스

MIT

작성자

이 프로젝트는 FastAPI + Prisma 하이브리드 스택으로 구현된 초대 기반 회원가입 시스템입니다.