시작하기

이 가이드는 DBA Portal Auth 서버를 로컬 환경에서 빠르게 시작하는 방법을 안내합니다.

사전 요구사항

- Node.js 18 이상
- MySQL 8.0 이상 (또는 AWS RDS MySQL)
- npm 또는 yarn

빠른 시작

1. 프로젝트 클론 및 의존성 설치

```
cd dba-portal-auth
npm install
```

2. 환경변수 설정

`.env.example` 파일을 복사하여 `.env` 파일을 생성합니다:

```
cp .env.example .env
```

`.env` 파일을 열어 다음 값들을 설정합니다:

필수 설정

데이터베이스 (로컬 MySQL)

```
DATABASE_URL="mysql://root:password@localhost:3306/dba_portal_auth"
```

JWT 시크릿 키 (임의의 강력한 문자열)

```
JWT_ACCESS_SECRET="my-super-secret-access-key-change-this"
JWT_REFRESH_SECRET="my-super-secret-refresh-key-change-this"
```

OAuth 설정 (실제 사용 시 필요)

Google OAuth

1. Google Cloud Console에서 프로젝트 생성

2. OAuth 2.0 클라이언트 ID 생성
3. 리다이렉트 URI: `http://localhost:3000/auth/google/callback`
4. 발급받은 정보를 입력:

```
GOOGLE_CLIENT_ID="your-google-client-id"
GOOGLE_CLIENT_SECRET="your-google-client-secret"
```

Kakao OAuth

1. Kakao Developers에서 앱 생성
2. 카카오 로그인 활성화
3. Redirect URI: `http://localhost:3000/auth/kakao/callback`
4. 발급받은 정보를 입력:

```
KAKAO_CLIENT_ID="your-kakao-client-id"
KAKAO_CLIENT_SECRET="your-kakao-client-secret"
```

CoolSMS

1. CoolSMS 가입 및 API 키 발급
2. 발신번호 등록
3. 발급받은 정보를 입력:

```
COOLSMS_API_KEY="your-coolsms-api-key"
COOLSMS_API_SECRET="your-coolsms-api-secret"
```

3. 데이터베이스 설정

로컬 MySQL 사용

MySQL을 설치하고 데이터베이스를 생성합니다:

```
CREATE DATABASE dba_portal_auth CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

Prisma 마이그레이션 실행

```
# Prisma 클라이언트 생성
npm run prisma:generate

# 마이그레이션 실행
npm run prisma:migrate
```

마이그레이션 이름을 입력하라는 메시지가 나오면 `init` 또는 원하는 이름을 입력합니다.

4. 개발 서버 실행

```
npm run dev
```

서버가 성공적으로 시작되면 다음과 같은 메시지가 표시됩니다:

```
✅ 데이터베이스 연결 성공
🧹 만료된 토큰 정리 완료
🚀 서버가 포트 3000에서 실행 중입니다.
📝 API 문서: http://localhost:3000/health
🔧 개발 모드로 실행 중
📱 SMS는 콘솔에 출력됩니다
```

5. 서버 테스트

브라우저나 curl로 health check 엔드포인트를 호출합니다:

```
curl http://localhost:3000/health
```

응답:

```
{
  "status": "success",
  "data": {
    "status": "OK",
    "timestamp": "2025-10-08T10:30:00.000Z",
    "uptime": 123.45
  },
  "message": "Server is healthy"
}
```

개발 모드에서 테스트

전화번호 인증 테스트

개발 모드(`NODE_ENV=development`)에서는 실제 SMS를 발송하지 않고 콘솔에 인증 코드를 출력합니다.

```
# SMS 인증 코드 요청
curl -X POST http://localhost:3000/auth/send-verification \
  -H "Content-Type: application/json" \
  -d '{"phone": "01012345678"}'
```

콘솔에서 인증 코드를 확인:

```
[SMS] 전화번호: 01012345678, 인증코드: 123456
```

```
# 인증 코드 확인
curl -X POST http://localhost:3000/auth/verify-phone \
  -H "Content-Type: application/json" \
  -d '{"phone": "01012345678", "verificationCode": "123456"}'
```

OAuth 테스트

브라우저에서 다음 URL을 방문:

- Google 로그인: `http://localhost:3000/auth/google`
- Kakao 로그인: `http://localhost:3000/auth/kakao`

프로젝트 구조

```
dba-portal-auth/
├── src/
│   ├── controllers/      # API 컨트롤러
│   │   └── auth.controller.ts
│   ├── middleware/       # 미들웨어
│   │   ├── auth.middleware.ts
│   │   ├── error.middleware.ts
│   │   ├── passport.middleware.ts
│   │   └── rate-limit.middleware.ts
│   ├── routes/           # 라우터
│   │   └── auth.routes.ts
│   ├── services/         # 비즈니스 로직
│   │   ├── jwt-token.service.ts
│   │   ├── phone-verification.service.ts
│   │   ├── prisma.service.ts
│   │   ├── sms.service.ts
│   │   └── user.service.ts
│   ├── types/            # TypeScript 타입
│   │   ├── auth.types.ts
│   │   └── passport-kakao.d.ts
│   ├── utils/            # 유틸리티
│   │   ├── jwt.util.ts
│   │   ├── phone.util.ts
│   │   └── response.util.ts
│   ├── app.ts            # Express 앱 설정
│   └── index.ts          # 서버 진입점
├── prisma/
│   └── schema.prisma     # 데이터베이스 스키마
```

```
├── .env                    # 환경변수 (git에서 제외)
├── .env.example            # 환경변수 예시
├── package.json            # 프로젝트 메타데이터
├── tsconfig.json           # TypeScript 설정
├── nodemon.json            # Nodemon 설정
├── README.md               # 프로젝트 개요
├── API_DOCUMENTATION.md    # API 문서
├── DEPLOYMENT.md           # 배포 가이드
└── GETTING_STARTED.md      # 이 파일
```

다음 단계

1. **OAuth 설정 완료**: Google과 Kakao OAuth를 실제로 설정하여 소셜 로그인 테스트
2. **CoolSMS 설정**: 실제 SMS 발송을 위한 CoolSMS 설정
3. **프론트엔드 연동**: 클라이언트 애플리케이션과 연동
4. **배포**: AWS RDS와 EC2를 사용하여 프로덕션 배포

추가 리소스

- API 문서 - 모든 API 엔드포인트 상세 설명
- 배포 가이드 - 프로덕션 환경 배포 방법
- 프로젝트 계획 - 프로젝트 설계 및 계획

문제 해결

데이터베이스 연결 오류

```
Error: P1001: Can't reach database server
```

해결 방법:

- MySQL 서버가 실행 중인지 확인
- `.env`의 `DATABASE_URL`이 올바른지 확인
- 데이터베이스가 존재하는지 확인

Prisma 마이그레이션 오류

```
Error: P3014: Prisma Migrate could not create the shadow database
```

해결 방법:

```
GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost';
FLUSH PRIVILEGES;
```

TypeScript 컴파일 오류

```
# node_modules와 빌드 결과 삭제 후 재설치
rm -rf node_modules dist
npm install
npm run build
```

포트가 이미 사용 중

```
Error: listen EADDRINUSE: address already in use :::3000
```

해결 방법:

- `.env`에서 `PORT` 값을 변경하거나
- 3000 포트를 사용 중인 프로세스를 종료

지원

문제가 발생하면 다음을 확인하세요:

1. 모든 환경변수가 올바르게 설정되었는지
2. 데이터베이스가 실행 중이고 연결 가능한지
3. Node.js 버전이 18 이상인지
4. 의존성이 모두 설치되었는지

개발을 즐기세요! 🚀