

16. 선택적 도전 과제-1-2 구현

1. 필요 라이브러리 설치

```
npm i nodemailer  
npm i -D @types/nodemailer
```

2. 네이버 메일 smtp/pop3 설정

- **POP3/SMTP 사용** : 사용 하는 것에 체크해야함.

3. EmailService 구현

```
// email.service.ts  
import { Injectable } from '@nestjs/common';  
import * as nodemailer from 'nodemailer';  
  
interface EmailOptions {  
  from: string;  
  to: string;  
  subject: string;  
  html: string;  
}  
  
@Injectable()  
export class EmailService {  
  private transporter;  
  
  constructor() {  
    this.transporter = nodemailer.createTransport({  
      service: 'naver',  
      auth: {  
        user: 'practice93@naver.com',  
        pass: '1234',  
      },  
    });  
  }  
}
```

```

async sendVerificationToEmail(email: string, code: string): Promise<void> {
  const emailOptions: EmailOptions = {
    from: 'practice93@naver.com',
    to: email,
    subject: '가입 인증 메일',
    html: `<h1>인증 코드: ${code}</h1>`,
  };

  await this.transporter.sendMail(emailOptions);
}
}

```

4. 이메일 인증 프로토타입

- users.controller.ts

```

@Post('/sendcode')
sendCode(@GetUser() userToken: User) {
  // 여기서 @GetUser는 jwt 토큰 자체 정보를 가져옴
  return this.usersService.sendVerificationCode(userToken);
}

@Post('/confirmcode')
confirmCode(@Body('verificationCode') code: string, @GetUser() userToken: User) {
  return { "isVerified": this.usersService.confirmVerificationCode(code,
userToken) };
}

```

- users.service.ts

```

async sendVerificationCode(userToken: User): Promise<void> {
  const code = 'verifycode'; /* 인증 코드 생성 로직 */

  // 유저 정보 가져오기
  const user = await this.findByUsername(userToken.username);

  user.verificationCode = code;
}

```

```

    console.log(user);

    await this.userRepository.save(user);
    await this.emailService.sendVerificationToEmail(user.username, code);
  }

  async confirmVerificationCode(code: string, userToken: User): Promise<boolean> {

    const user = await this.findByUsername(userToken.username);

    // user 객체에서 verificationCode 필드를 검사
    if (user.verificationCode === code) {
      // 코드가 일치하면 사용자의 인증 상태 업데이트
      user.isVerified = true;

      await this.userRepository.save(user);
      return true; // 인증 성공
    } else {
      return false; // 인증 실패
    }
  }
}

```

- `user.entity.ts`

```

import { Entity, PrimaryGeneratedColumn, Column } from 'typeorm';

// UserRole enum 정의
export enum UserRole {
  MEMBER = 'MEMBER',
  ADMIN = 'ADMIN',
}

@Entity()
export class User {
  @PrimaryGeneratedColumn()
  id: number;

  @Column({ unique: true })
  username: string;
}

```

```
@Column()
password: string;

@Column({
  type: 'enum',
  enum: UserRole,
  default: UserRole.MEMBER
})
role: UserRole;

@Column({ default: '' })
verificationCode: string;

@Column({ default: false })
isVerified: boolean;
}
```

- verificationCode, isVerified 부분이 추가되었다.

5. 이메일 인증 서비스 확인

- `/users/sendcode` 로 요청을 보낼 때 다음과 같이 이메일이 잘 오는 것을 볼 수 있다.

가입 인증 메일 ➡ 받은편지함 x



practice93@naver.com

나에게 ▼

인증 코드: verifycode

← 답장

↪ 전달



6. 인증 확인

- /users/confirmcode 로 보내고

```
{
  "verificationCode": "verifycode"
}
```

- 다음과 같이 요청하면

```
mysql> select * from opstest.user;
```

```
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
| id | username          | password
| role | verificationCode | isVerified |
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
```

```
| 3 | newnyup@gmail.com |  
$2b$10$EZj6RLaIHh0eyiCpXlZybeP.X4uGYwhFuvjY7cWdPStMX4bEtY..y | MEMBER | verifycode  
| 0 |
```

```
+-----+  
-----+-----+  
1 row in set (0.00 sec)
```

```
mysql> select * from opstest.user;
```

```
+-----+  
-----+-----+  
| id | username          | password  
| role | verificationCode | isVerified |  
+-----+  
-----+-----+  
| 3 | newnyup@gmail.com |  
$2b$10$EZj6RLaIHh0eyiCpXlZybeP.X4uGYwhFuvjY7cWdPStMX4bEtY..y | MEMBER | verifycode  
| 1 |
```

```
+-----+  
-----+-----+  
1 row in set (0.00 sec)
```

- isVerified 되는 것을 확인할 수 있다.