Seoul woman up

스타트업 개발자와 함께 공부하는

Node.js

10. 테스트 및 MVC 패턴

강의 내용은 강사가 별도로 명시하지 않는 한 비공개로 간주합니다. 녹음이나 사진 촬영를 허락하지 않으며 콘텐츠를 블로그, SNS 등에 게시하거나 공개적으로 공유하지 마세요.

콘텐츠 공유 가능 여부에 대해 궁금한 점이 있는 경우 강사에게 문의하시기 바랍니다.

목차

Seoul woman up

PostgreSQL

PostgreSQL

소개

- 관계형 데이터베이스 중 하나
- 무료 배포 데이터베이스 중 하나
- 오라클과 유사함
- DB 랭킹 4위

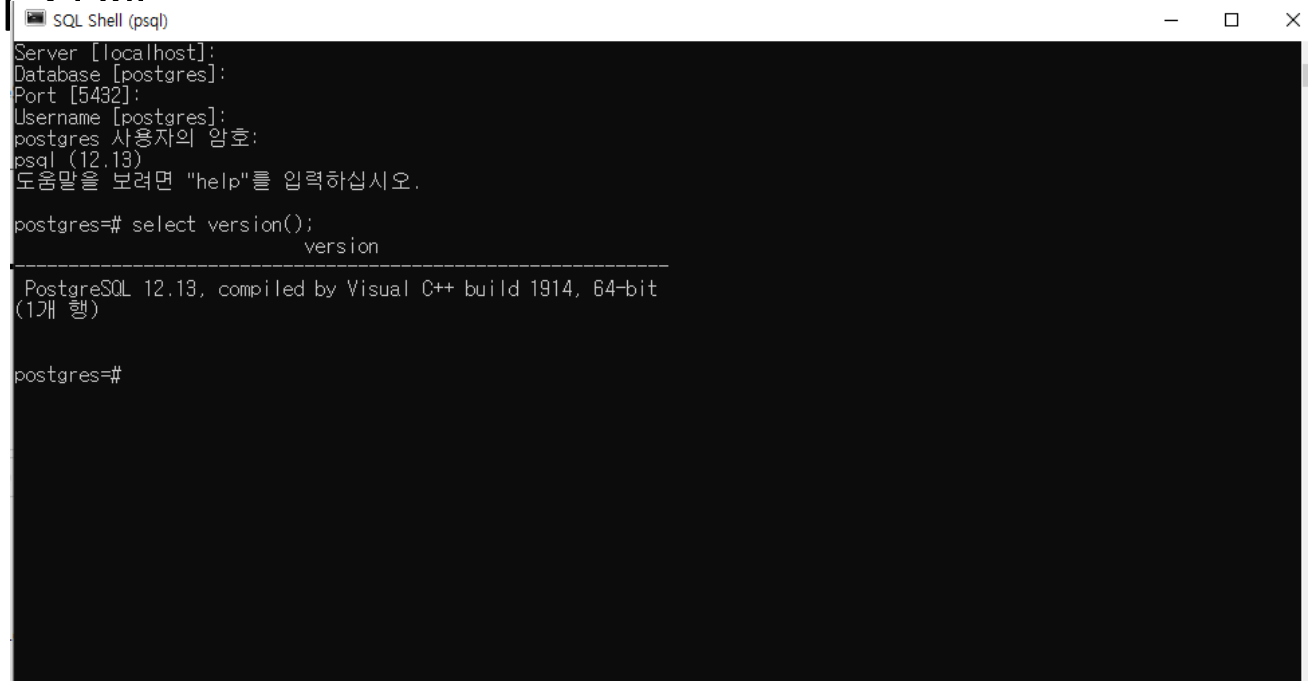
421 systems in ranking, July 2024

Rank			DBMS	Database Model	Score		
Jul 2024	Jun 2024	Jul 2023			Jul 2024	Jun 2024	Jul 2023
1.	1.	1.	Oracle	Relational, Multi-model	1240.37	-3.72	-15.64
2.	2.	2.	MySQL	Relational, Multi-model	1039.46	-21.89	-110.89
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	807.65	-13.91	-113.95
4.	4.	4.	PostgreSQL	Relational, Multi-model	638.91	+2.66	+21.08
5.	5.	5.	MongoDB	Document, Multi-model	429.83	+8.75	-5.67
6.	6.	6.	Redis	Key-value, Multi-model	156.77	+0.82	-7.00
7.	↑ 8.	↑ 11.	Snowflake	Relational	136.53	+6.17	+18.84
8.	↓ 7.	8.	Elasticsearch	Search engine, Multi-model	130.82	-2.01	-8.77
9.	9.	↓ 7.	IBM Db2	Relational, Multi-model	124.40	-1.50	-15.41
10.	10.	10.	SQLite	Relational	109.95	-1.46	-20.25

PostgreSQL

설치

- https://www.postgresql.org/download/
- Windows 다운로드
- 설치 완료 후에 작업 표시줄에서 psql 검색



PostgreSQL

데이터베이스 생성 및 유저 생성

```
// ch10_01.sql

1 drop database tut08;
2 sudo -u postgres psql
3 create database tut08;
4 create user admin with encrypted password 'admin1234';
5 grant all privileges on database tut08 to admin;
6
```

❑ [C]-[nodejs]-[project]-[10]-[ch10_01.sql]

Seoul woman up

테스트 프레임워크

테스트 프레임워크

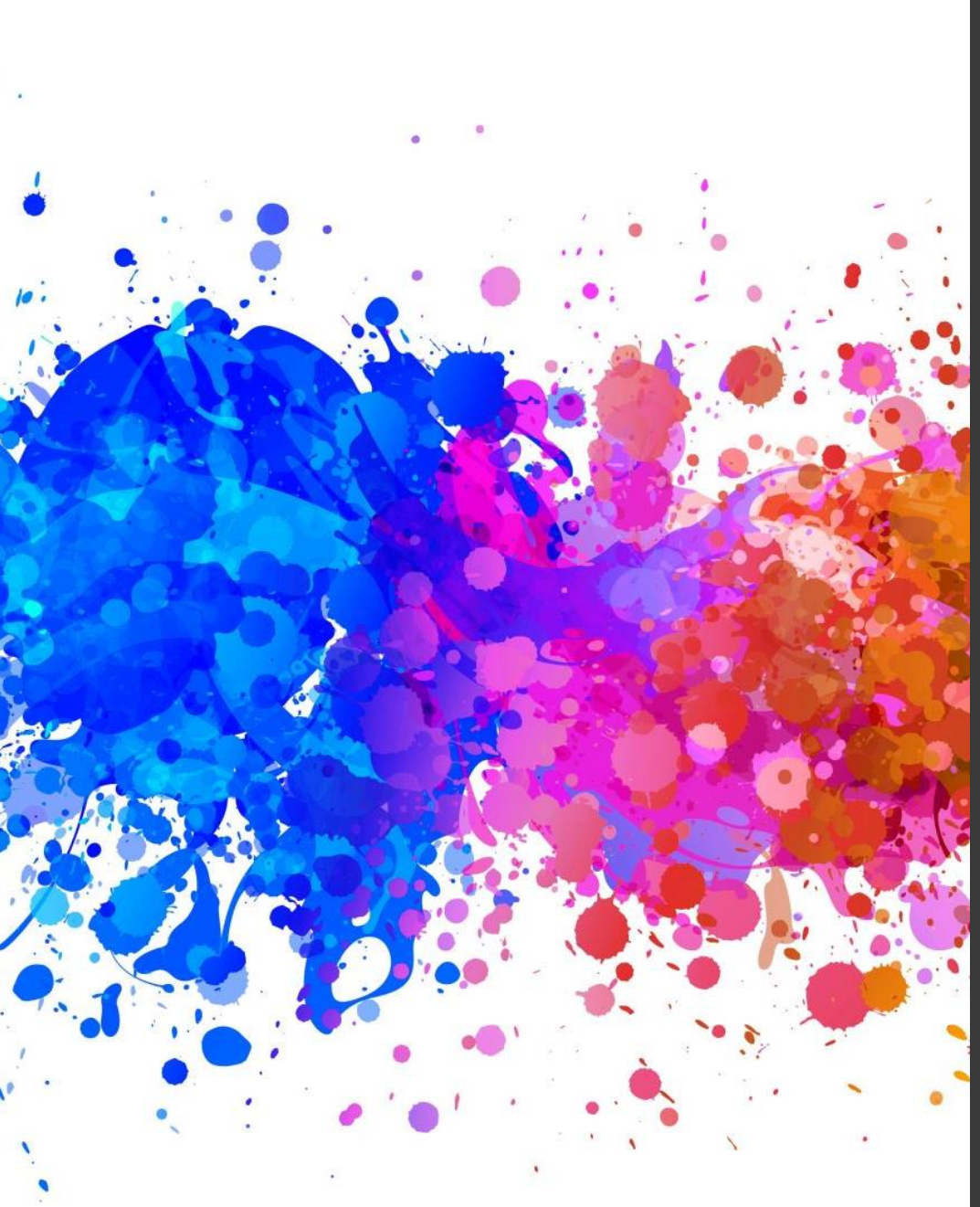
소개

- 자동화된 테스트. 실행
- assertion 라이브러리 지원
- 테스트 리포팅
- 비동기 코드 지원
- 모킹 및 스파이 기능
- 병렬 실행

테스트 프레임워크

종류

- Junit – Java
- PHPUnit – PHP
- Mocha – Node.js
- Jest – Node.js
- Pytest - Python



Seoul woman up

Jest

Jest

소개

- 페이스북에서 개발한 자바스크립트 및 Node.js 환경에서 사용되는 테스트 프레임워크
- 제로 설정
- 스냅샷 테스트
- 병렬 실행

Jest

프로젝트 생성

```
npm init -y
npm install jest

// package.json
{
  "scripts": {
    "test": "jest"
  }
}
```

- [C]-[nodejs]-[project]-[10]-[ch10_02]

Jest

예제

// math.js

```
1  // math.js
2
3  function sum(a, b) {
4      return a + b;
5  }
6
7  function subtract(a, b) {
8      return a - b;
9  }
10
11 module.exports = { sum, subtract };
12
```

// math.test.js

```
1  // math.test.js
2
3  const { sum, subtract } = require('./math');
4
5  test('sum 함수가 정확히 더하는지', () => {
6    expect(sum(1, 2)).toBe(3);
7    expect(sum(-1, 1)).toBe(0);
8  });
9
10 test('subtract 함수가 정확히 빼는지', () => {
11   expect(subtract(3, 1)).toBe(2);
12   expect(subtract(1, -1)).toBe(2);
13 });
```

Jest

express 테스트

```
npm install express supertest
```

- [C]-[nodejs]-[project]-[10]-[ch10_02]

Jest

express 테스트

// app.js

```
1  // app.js
2
3  const express = require('express');
4  const app = express();
5
6  app.get('/', (req, res) => {
7    res.send('Hello World!');
8  });
9
10 // app.listen(3000, () => {
11 //     console.log(`server is listening`);
12 // });
13 module.exports = app;
14
```

// app.test.js

```
1  // app.test.js
2
3  const request = require('supertest');
4  const app = require('./app');
5
6  describe('GET /', () => {
7    it('should respond with "Hello World!"', async () => {
8      const response = await request(app).get('/');
9      expect(response.status).toBe(200);
10     expect(response.text).toBe('Hello World!');
11   });
12 });
13
```

Seoul woman up

MVC 패턴

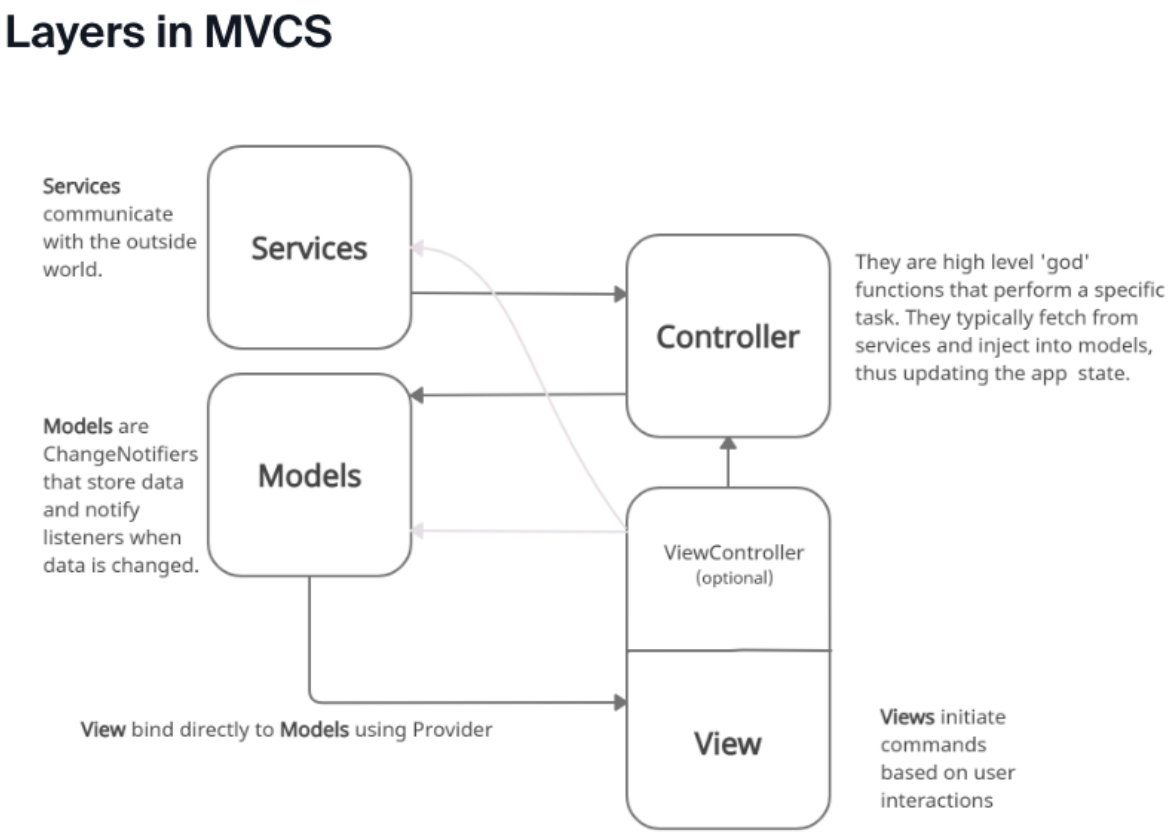
MVC 패턴

소개

- MVC(Model – View – Controller)
- Model (모델)
- View (뷰)
- Controller (컨트롤러)

MVC 패턴

MVCS 소개

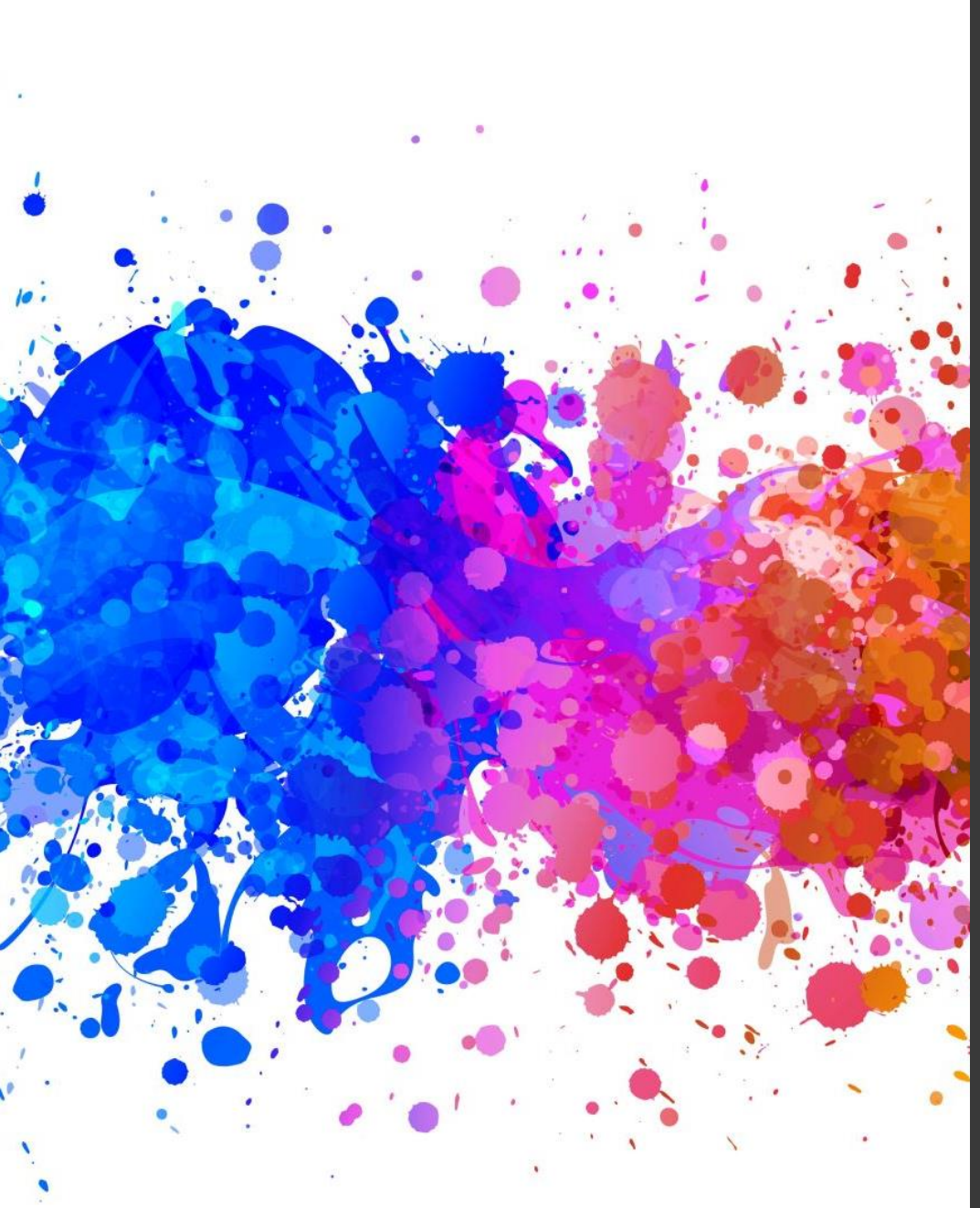


- ❑ Models
- ❑ Services
- ❑ Controller
- ❑ View

MVC 패턴

Service, Controller, DAO 패턴

- Controller 패턴
  - 사용자 요청 수신
  - 요청을 Service 전달 및 반환 결과 사용자 반환
- Service 패턴
  - 비즈니스 로직 처리
  - DAO 호출로 데이터 접근
- DAO(Data Access Object) 패턴
  - 데이터베이스 상호작용
  - CRUD



Seoul woman up

회원

회원

프로젝트 생성

```
npm init -y
npm install express jest nodemon pg
sequelize sequelize-cli
npx sequelize init
```

```
"devDependencies": {
  "jest": "^29.7.0",
  "@babel/preset-env": "^7.15.6",
  "@babel/preset-typescript": "^7.15.0",
  "@babel/plugin-transform-modules-commonjs": "^7.15.0"
},
"jest": {
  "transform": {
    "^.+\\.jsx?$": "babel-jest"
  }
}
```

- [C]-[nodejs]-[project]-[10]-[ch10_03]
- jest 관련 설정 : package.json 에 좌측 내용추가
- [illegible] 생성 후 아래 내용 입력

```
{
  "presets": ["@babel/preset-env"],
  "plugins": ["@babel/plugin-transform-modules-commonjs"]
}
```

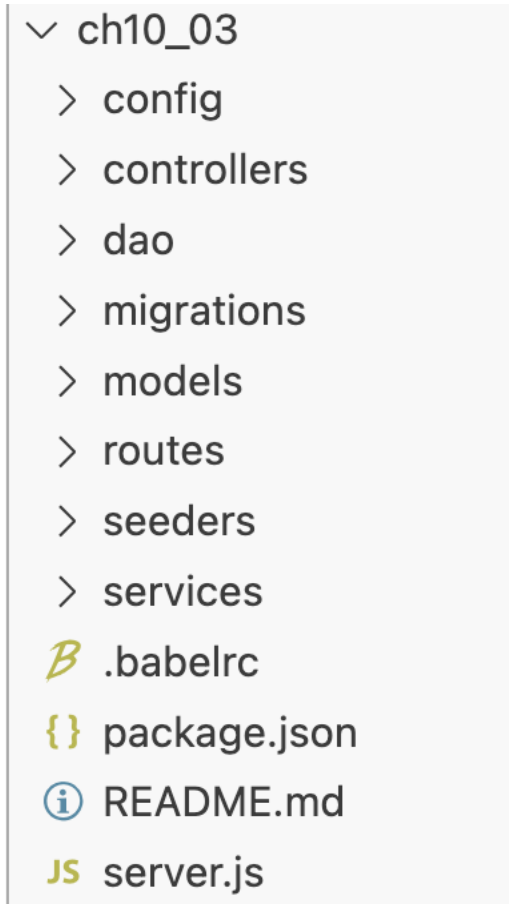
회원

소개

- 회원 모델(테이블) 생성
- 필드
  - 아이디 (id)
  - 이메일 (email)
  - 비밀번호(password)
  - 이름 (name)
  - 주소 (address)
  - 생성일 (create _at)

회원

디렉토리 구조



- ❑ config
- ❑ controller
- ❑ dao
- ❑ models
- ❑ routes
- ❑ services
- ❑ server.js

회원

모델

```
module.exports = (sequelize, DataTypes) => {
    const User = sequelize.define('User', {
        id: {
            type: DataTypes.INTEGER,
            allowNull: false,
            primaryKey: true,
            autoIncrement: true,
        },
        email: {
            type: DataTypes.STRING,
            allowNull: false,
            unique: true
        },
        password: {
            type: DataTypes.STRING,
            allowNull: false,
        },
        name: DataTypes.STRING,
        address: DataTypes.STRING,
        create_at: {
            type:DataTypes.DATE,
            allowNull: false,
            defaultValue: DataTypes.NOW
        }
    });
    return User;
}
```

- ❑ models/user.js
- ❑ id
- ❑ email
- ❑ password
- ❑ name
- ❑ address
- ❑ create_at

회원

DAO

```
1  const models = require('../models');
2
3  const createUser = async (userData) => {
4      return await models.User.create(userData)
5  };
6
7  const getUserById = async (id) => {
8      return await models.User.findByPk(id);
9  };
10
11 const getAllUsers = async () => {
12     return await models.User.findAll();
13 };
14
15 const updateUser = async (id, userData) => {
16     return await models.User.update(userData, {
17         where : {id},
18     });
19 };
20
21 const deleteUser = async (id) => {
22     return await models.User.destory({
23         where : { id }
24     });
25 };
```

- ❑ dao/userDao.js

- ❑ createUser
- ❑ getUserById
- ❑ getAllUsers
- ❑ updateUser
- ❑ deleteUser

회원

DAO - test

```
1  const userDao = require('./userDao');
2
3  describe('User DAO', () => {
4      test('should create a new user', async () => {
5          const userData = {
6              email: 'jane1@example.com',
7              name: 'jane doe',
8              password: 'jane1234'
9          };
10         const createdUser = await userDao.createUser(userData);
11         expect(createdUser.email).toBe(userData.email)
12     });
13 });
14
```

❑ dao/userDao.test.js

회원

Service

```
1  const userDao = require('../dao/userDao');
2
3  const createUser = async (userData) => {
4      return await userDao.createUser(userData);
5  };
6
7  const getUserById = async (id) => {
8      return await userDao.getUserById(id);
9  };
10
11 const getAllUsers = async () => {
12     return await userDao.getAllUsers();
13 };
14
15 const updateUser = async (id, userData) => {
16     return await userDao.updateUser(id, userData);
17 };
18
19 const deleteUser = async (id) => {
20     return await userDao.deleteUser(id);
21 }
```

- ❑ services/userService.js

- ❑ createUser
- ❑ getUserById
- ❑ getAllUsers
- ❑ updateUser
- ❑ deleteUser

회원

Controller

```
1  const userService = require('../services/userService');
2
3  const createUser = async(req, res) => {
4      try{
5          const user = await userService.createUser(req.body);
6          res.status(201).json(user)
7      }catch(error) {
8          res.status(500).json({error: error.message});
9      }
10 };
11
12 const getUserById = async (req, res) => {
13     try{
14         const user = await userService.getUserById(req.params.id);
15         if(user) {
16             res.status(200).json(user)
17         } else {
18             res.status(404).json({message: 'User not found'});
19         }
20     }catch(error) {
21         res.status(500).json({error: error.message});
22     }
23 };
24
25 const getAllUsers = async (req, res) => {
26     try{
27         const users = await userService.getAllUsers();
28         res.status(200).json(users);
29     }catch(error) {
30         res.status(500).json({error: error.message});
31     }
32 };
```

- controllers/userController.js

- createUser
- getUserById
- getAllUsers
- updateUser
- deleteUser

회원

Routes

```
1  const express = require('express');
2  const userController = require('../controllers/userController');
3
4  const router = express.Router();
5
6  router.post('/', userController.createUser);
7  router.get('/:id', userController.getUserById);
8  router.get('/', userController.getAllUsers);
9  router.put('/:id', userController.updateUser);
10 router.delete('/:id', userController.deleteUser);
11
12 module.exports = router;
```

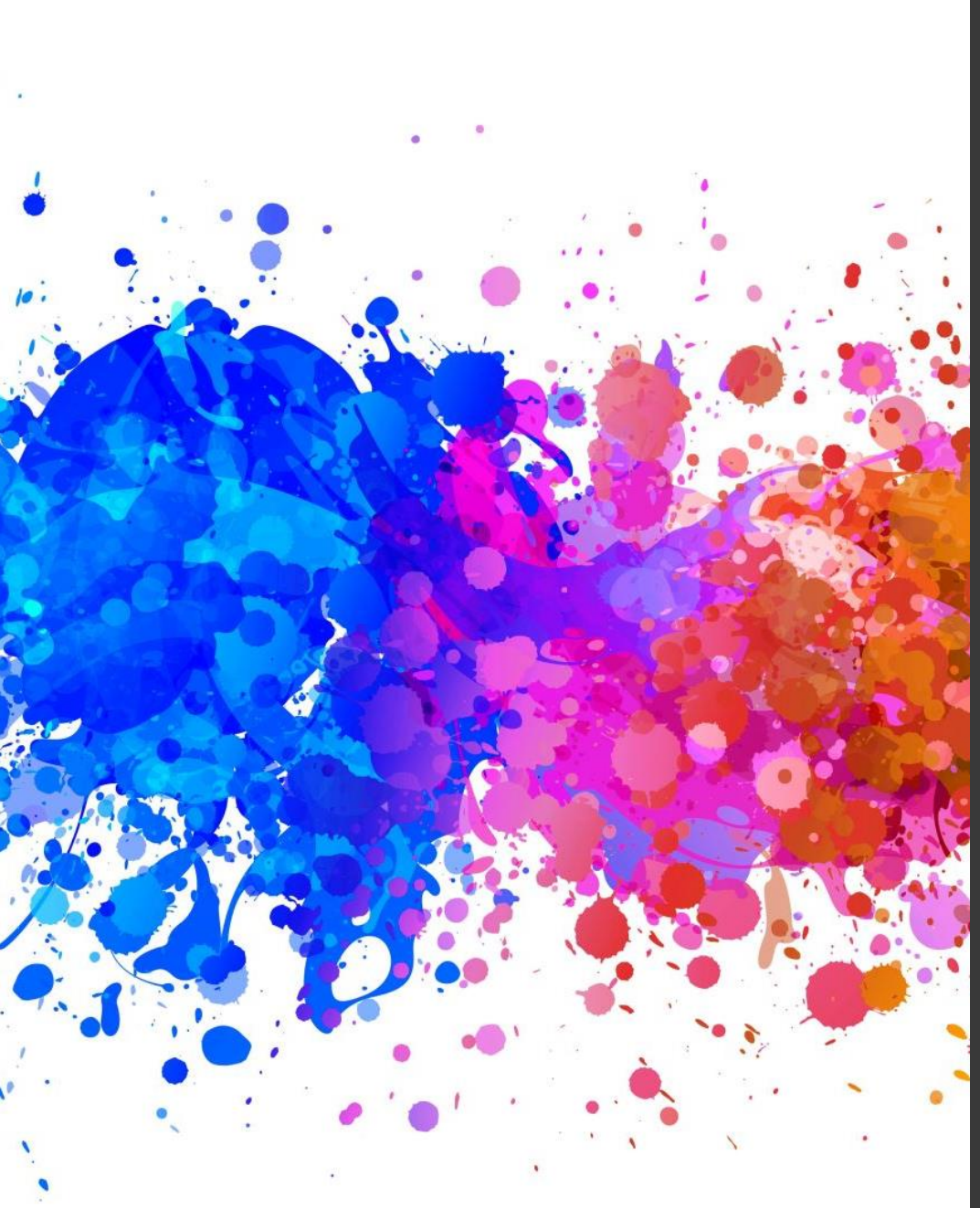
❑ routes/userRoutes.js

회원

server.js

```
1  const express = require('express');
2  const fs = require('fs');
3  const path = require('path');
4  const userRouter = require('./routes/userRoutes');
5  const boardRouter = require('./routes/boardRoutes');
6
7  const models = require('./models');
8  const app = express();
9  const PORT = process.env.PORT || 3000;
10
11 app.use(express.json());
12 app.use('/users', userRouter);
13 app.use('/boards', boardRouter);
14
15 app.listen(PORT, () => {
16     console.log(`Server will be start..`);
17     models.sequelize.sync({force:false}).then(()=> {
18         console.log(`DB 연결 성공`);
19     }).catch((err) => {
20         console.error(`DB 연결 에러 : ${err}`);
21         process.exit();
22     });
23 });
24 console.log(`Server is listening on port 3000`);
```

❑ server.js



Seoul woman up

게시판 추가

게시판 추가

추가 파일

- model/board.js
- dao/boardDao.js
- services/boardService.js
- controllers/boardController.js
- routes/boardRoute.js

Seoul woman up

백문이 불여일타