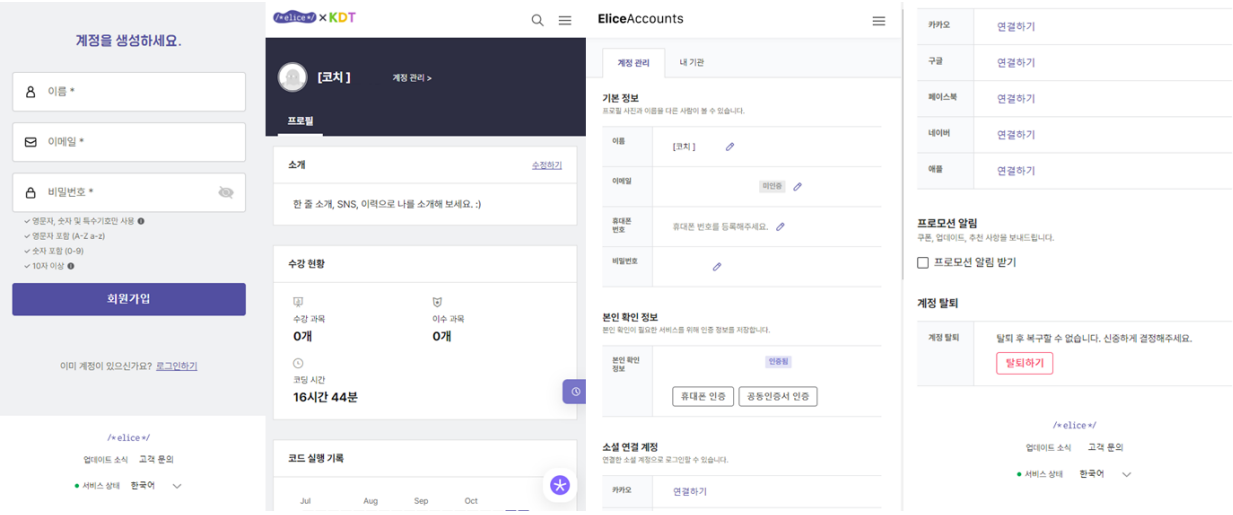
2022-10-24 2반 실습

학습 포인트

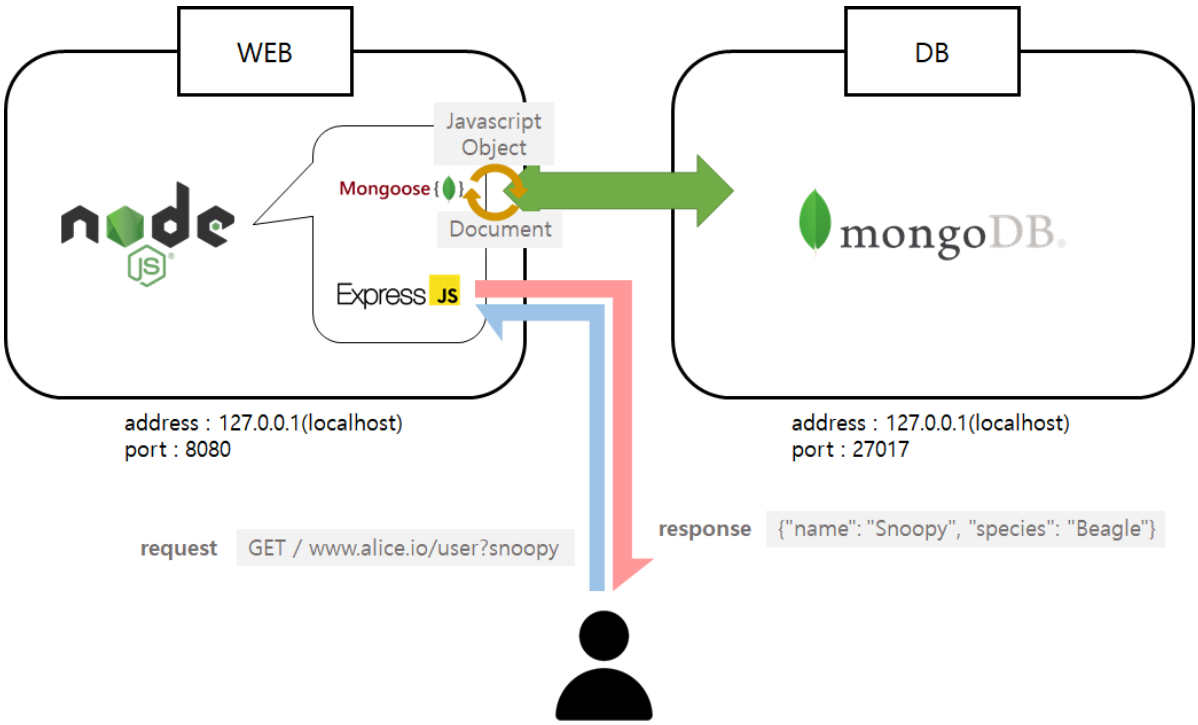
- **무엇**을 개발할 것인가?

  → 웹을 통해 사용자 정보를 DB에 정보를 `CRUD`(생성-create, 조회-read, 수정-update, 삭제-delete)하는 시스템 개발



- **어떻게** 개발할 것인가?

  → `Node js`, `Express`, `MongoDB`, `Mongoose`를 사용하여 개발



과정

1. 준비 : Configuration, Prerequisites, Dependencies
   - 개발 전 필요한 환경설정, 요구사항, 프레임워크, 라이브러리 등을 준비(설치, 설정 등) 한다.
     - node js
     - express

```
npm install express
```

     - mongodb
     - mongoose

```
npm install mongoose
```

2. nodejs에서 exptess, mongoose 모듈 불러오기
   - const express = require('express');

```
const { Router } = require('express');
const { Router } = require('mongoose');
...
```

3. DB 연결

- DB URI를 설정해서 연결(db사용자명:주소:포트/db명)

```
mongoose
  .connect(MONGO_URI)
  .then(() => console.log('Successfully connected to mongodb'))
  .catch(e => console.error(e));
```

4. Schema 정의

- MongoDB에는 명시적인 구조가 없기 때문에 데이터의 타입이 알기 어렵기 때문에 mongoose에서 스키마를 사용하여 보완
- 기본 데이터 타입

Data Type	Description
String	표준 자바스크립트와 Node.js의 문자열 type
Number	표준 자바스크립트와 Node.js의 숫자 type
Boolean	표준 자바스크립트와 Node.js의 불리언 type
Buffer	Node.js의 binary type(이미지, PDF, 아카이브 등)
Date	ISODate format data type(2016-08-08T12:52:30:009Z)
Array	표준 자바스크립트와 Node.js의 배열 type
Schema.types.ObjectId	12byte binary 숫자에 대한 MongoDB 24자리 hex 문자열 (501d86090d371bab2c0341c5)
Schema.types.Mixed	모든 유형의 데이터

- 스키마 속성

Data Type	Description
required	필수 입력
unique	다른 행과 중복되면 비허용

Data Type	Description
trim	공백 제거(문자열 타입에 사용)
default	문서가 생성되면 기본값으로 저장
lowercase	대문자를 소문자로 저장
match	정규식으로 저장하려는 값과 비교
validate	함수로 개발자가 조건을 설정
set	값을 입력할 때 함수로 조건을 설정
get	값을 출력할 때 함수로 조건을 설정
ref	해당하는 모델을 참조할 때 사용

- 예제

```
const { Schema } = require('mongoose');
const shortId = require('./types/short-id');

const PostSchema = new Schema({
 shortId,
  title: {
    type: String,
    required: true,
  },
  content: {
    type: String,
    required: true,
  },
  author: {
    type: String,
    default: '작성자',
  }
}, {
  timestamps: true,
});

module.exports = PostSchema;
```

5. 기능 구현
    - node js에서 mongoose 내장 함수를 사용하여 데이터의 생성, 조회, 수정, 삭제 기능을 구현
    - Mongoose 내장 함수

Data Type	Description
create()	Model로부터 call하는 메소드, model.create()
save()	instance로부터 call하는 메소드, instance.save()
find()	조건에 해당하는 모든 documents 조회, 메소드의 인자로 필터링할 조건
findOne()	조건에 해당하는 하나의 document만 조회, 메소드의 인자로 필터링할 조건
findById()	request의 파라미터로 넘어오는 id를 통해 document를 조회, 메소드의 인자로 필터링할 조건
updateOne()	조건에 해당하는 하나의 데이터만 수정
updateMany()	조건에 해당하는 모든 데이터를 수정

```
//create
await Character.create({ name: 'Jean-Luc Picard' });
await Character.create([{ name: 'Will Riker' }, { name: 'Geordi LaForge' }]);

//find
// find all documents
await MyModel.find({});
// find all documents named john and at least 18
await MyModel.find({ name: 'john', age: { $gte: 18 } }).exec();
// executes, passing results to callback
MyModel.find({ name: 'john', age: { $gte: 18 }}, function (err, docs) {});
// Find one adventure whose `country` is 'Croatia', otherwise `null`
await Adventure.findOne({ country: 'Croatia' }).exec();
//findById()
async (req, res) => {
  const { id } = req.params;
  const post = await Post.findById(id);
  return res.render("write", { pageTitle: post.title, post });
};

//update
User.updateOne({name: "cozups"},{age: 20});
User.updateMany({필터},{업데이트 할 내용});


//delete
User.deleteOne({name: "cozups"},{age: 20});
User.deleteMany({필터},{삭제 할 내용});
```

- `exec()` : 사용 시 유사 프로미스가 아닌 온전한 프로미스를 반환값으로 얻을 수 있으며, 에러가 났을 때 stack trace에 오류가 발생한 코드의 위치가 포함되기 때문에 공식 문서에서도 exec()을 사용할 것을 권장하고 있다.
- `req.params` 주소에 포함된 변수를 담는다.
- `req.query` 주소 바깥의 ? 이후의 변수를 담는다.
- `req.body` XML, JSON, Multi Form 등의 데이터를 담는다.