



TODO

2015270238 컴퓨터정보학과 김진우

목차

1. 서론

1. 목적
2. 범위
3. 개발환경

2. 시스템 구조

1. 아키텍처 설계
2. 분해설명
3. 구성 요소 설계

3. 사용자 인터페이스 설계

1. 사용자 인터페이스 개요
2. 화면 이미지
3. 화면 개체 및 동작

서론

I. 목적

이 PPT의 목적은 Todo 웹 어플리케이션의 아키텍처 및 시스템 설계를 설명합니다.

서론

2. 범위

Todo 소프트웨어의 목적은 사용자가 해야 할 일을 작성 및 저장하여 추후 본인이 해야 할 일을 언제 기록하였는지 그리고 추가적인 설명에 따라 언제 작성하였는지 표시되어있어 어플리케이션을 통해 한 눈에 사용자가 해야 할 일을 쉽고 명료하게 파악할 수 있는 이점이 있습니다.

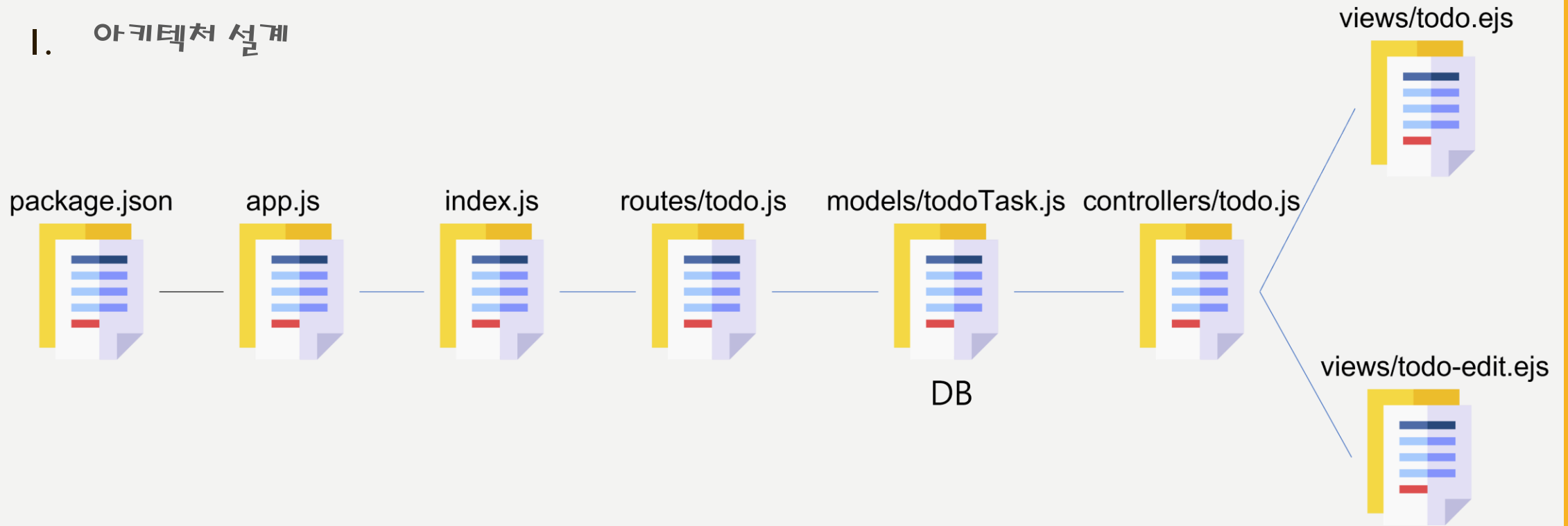
서론

3. 개발환경

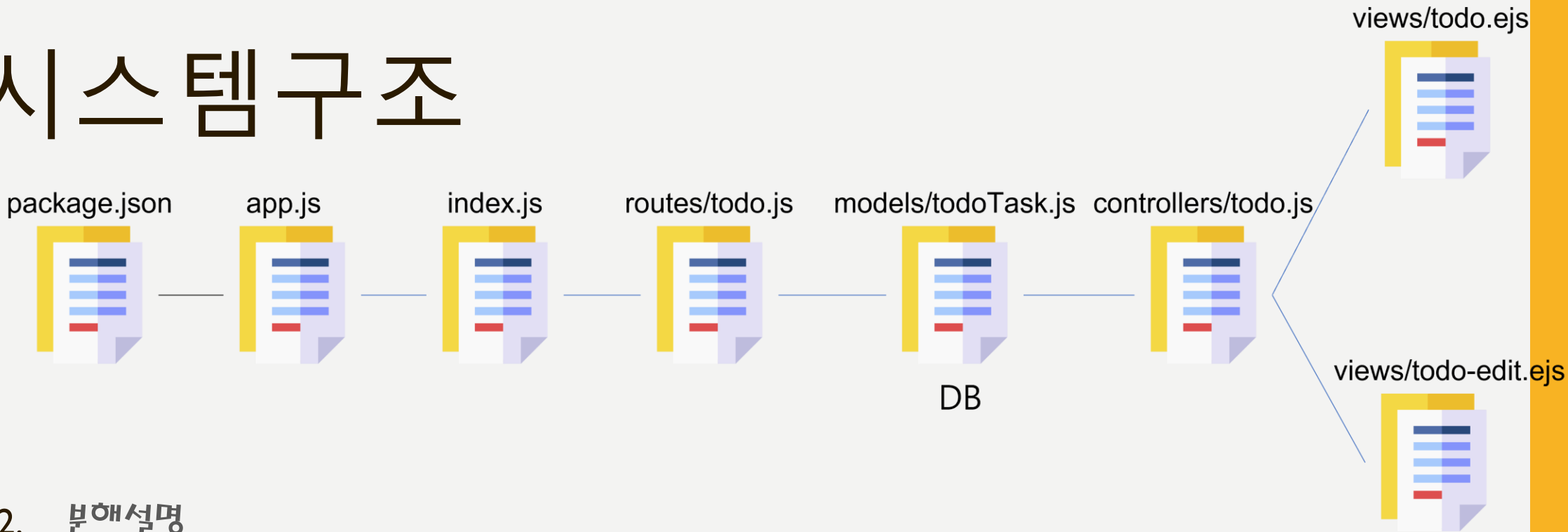
- Node.js
- Visual Code
- MongoDB

시스템구조

I. 아키텍처 설계



시스템구조



2. 본해설명

`package.json`은 `app.js`의 설정파일이고, 라우터 설정을 `routes/index.js`로 설정함으로써 `index`를 모듈처럼 가져와 사용합니다. `routes/index.js`의 라우팅 설정을 `routes/todo.js`로 설정하고, `routes/todo.js`의 라우팅 설정을 `controllers/todo.js`로 하여 `controllers/todo.js`의 작성, 편집, 수정, 삭제 함수들을 이용할 수 있도록 하고, 이때 `models/todoTask.js`의 DB 스키마를 사용합니다. 그리고 결과적으로 `views/todo.ejs` 화면으로 보여주며, 수정할 시에는 `views/todo-edit.ejs` 화면을 거쳐 `views/todo.ejs`로 보여줍니다.

시스템구조

2. 분해설명

<첫 페이지를 보여주는 함수>

```
// 첫 페이지
exports.get = function(req, res){
  console.log("-----!!Todo!!-----");
  TodoTask.find({}, null, {sort: {date: -1}}, (err, tasks) => {
    res.render("todo", { todoTasks: tasks });
  });
};
```

처음 보여질 메인 화면에는 작성했던 TodoList들이 보여져야하기 때문에 TodoTask.find()로 tasks들을 가져와서 “todo” 라는 페이지에 todoTasks 라는 이름으로 보냅니다. models/todoTask.js 에 module.exports = mongoose.model('TodoTask' ,todoTaskSchema);를 작성했기 때문에 TodoTask라는 이름으로 todoTaskSchema에 접근할 수 있다. find라는 메소드는 다음과 같이 사용되는데, “filter: {} 모든 데이터, options:{sort:{date:-1}}정렬 -date를 기준으로 내림차순으로 callback:function(err, tasks){} tasks로 받아와 res.render로 todo 페이지를 보여줘라” 라는 의미입니다.

시스템구조

2. 분해설명

<작성 함수>

해야할 일(todoList)를 작성하기 위해 만들어진 함수입니다. 새로운 todoTask를 만들어서 todoTask에 저장하고, 이 때 내용(입력한 부분)과 현재 시간이 save()함수를 통해 DB에 저장됩니다.

```
// 작성
exports.write = async function(req, res){
  try{
    const todoTask = new TodoTask({
      content: req.body.content,
      date: moment().format("YYYY-MM-DD HH:mm:ss")
    });
    await todoTask.save();
    console.log("==== Success!! Save New TodoTask =====");
    console.table([
      {id: todoTask._id, content: todoTask.content, date: todoTask.date}
    ]);
    res.redirect("/todo");
  }catch(err){
    console.err("==== Fail!! Save TodoTask =====");
    res.redirect("/todo");
  }
};
```

시스템구조

2. 분해설명

<편집 함수>

해야할 일(todoList)들을 db에서 조회해서 todo-edit.ejs에 id와 함께 보내지는 용도로 사용됩니다.

```
// 편집
exports.edit = function(req, res){
  const id = req.params.id;
  TodoTask.find({}, null, {sort: {date: -1}}, (err, tasks) => {
    res.render("todo-edit", { todoTasks: tasks, idTask: id });
  });
};
```

시스템구조

2. 분해설명

<수정 함수>

해당 id값의 해야할 일(content)를 변경하기 위해 사용되는 함수입니다.

```
// 수정
exports.update = function(req, res){
  const id = req.params.id;
  TodoTask.findByIdAndUpdate(id, { content: req.body.content }, err => {
    if(err){
      console.log("==== Fail!! Update TodoTask ====");
      console.error(err);
    }
    console.log("==== Success!! Update TodoTask ====");
    console.log("id: " + id + "\nchanged content: " + req.body.content);
    res.redirect("/todo");
  });
}
```

시스템구조

2. 분해설명

<삭제 함수>

해당 id값의 데이터를 삭제하기 위해 사용되는 함수입니다.

```
//삭제
exports.remove = function(req, res){
  const id = req.params.id;
  TodoTask.findByIdAndRemove(id, err => {
    if(err){
      console.log("==== Fail!! Remove TodoTask ====")
      console.error(err);
    }
    console.log("==== Success!! Remove TodoTask ====");
    console.log("id: " + id);
    res.redirect("/todo");
  });
};
```

시스템구조

3. 구성 요소 설계

//백엔드 부분

controllers – 실제 데이터를 처리하는 로직이 담겨있는 모듈이 들어있는 폴더

models – mongoDB에서 사용할 스키마 파일들이 담겨있는 모듈이 들어있는 폴더

routes – url을 보고 controller의 메소드를 연결해주는 모듈이 들어있는 폴더

// 프론트엔드

public – fonts, css, images 파일들이 담겨있는 폴더

views – 웹 페이지 ejs,html 등을 사용하는 파일들이 들어있는 폴더

//실행 파일












app.js – 실행 서버 파일

package.json – dependencies 리스트가 들어있어서 이 폴더 경로에서 npm install시 모듈이 전부 설치됨.

사용자 인터페이스 설계

I. 사용자 인터페이스 개요


사용자는 웹 어플리케이션을 실행함으로 아래의 화면 이미지들을 보게됩니다. 먼저 빈 화면의 Enter todo라고 적혀있는 빈 칸에 사용자가 해야할 일을 적고, “Write” 버튼을 눌러 추가합니다. 해야할 일을 추가하면 아래 칸에 추가된 날짜와 해야할 일 목록이 저장됩니다. 해야할 일 목록 옆의 버튼 2개를 통해 수정 및 삭제가 가능한데, 먼저 왼쪽 버튼은 해야할 일을 수정하는 버튼이고, 수정 후 다시 눌러주면 수정된 내용이 저장됩니다. 오른쪽 버튼은 해야할 일을 삭제하는 버튼으로 버튼을 누르면 해야할 일이 삭제되며, 삭제된 인터페이스가 자동으로 반영됩니다.

				
TODO LIST	TODO LIST	TODO LIST	TODO LIST	TODO LIST
<input type="text" value="Enter todo"/> <input type="button" value="Write"/>	<input type="text" value="test2"/> <input type="button" value="Write"/>	<input type="text" value="Enter todo"/> <input type="button" value="Write"/>	<input type="text" value="Enter todo"/> <input type="button" value="Write"/>	<input type="text" value="Enter todo"/> <input type="button" value="Write"/>
	2021-12-17 test1  	2021-12-17 test1  	2021-12-17 todo2  	
처음 화면	작성 화면	수정 화면	수정완료 화면	삭제 화면

사용자 인터페이스 설계

2. 화면이미지

처음 화면



TODO LIST

작성 화면



TODO LIST

2021-12-17

test1

수정 화면




TODO LIST

2021-12-17



수정완료 화면




TODO LIST

2021-12-17

todo2

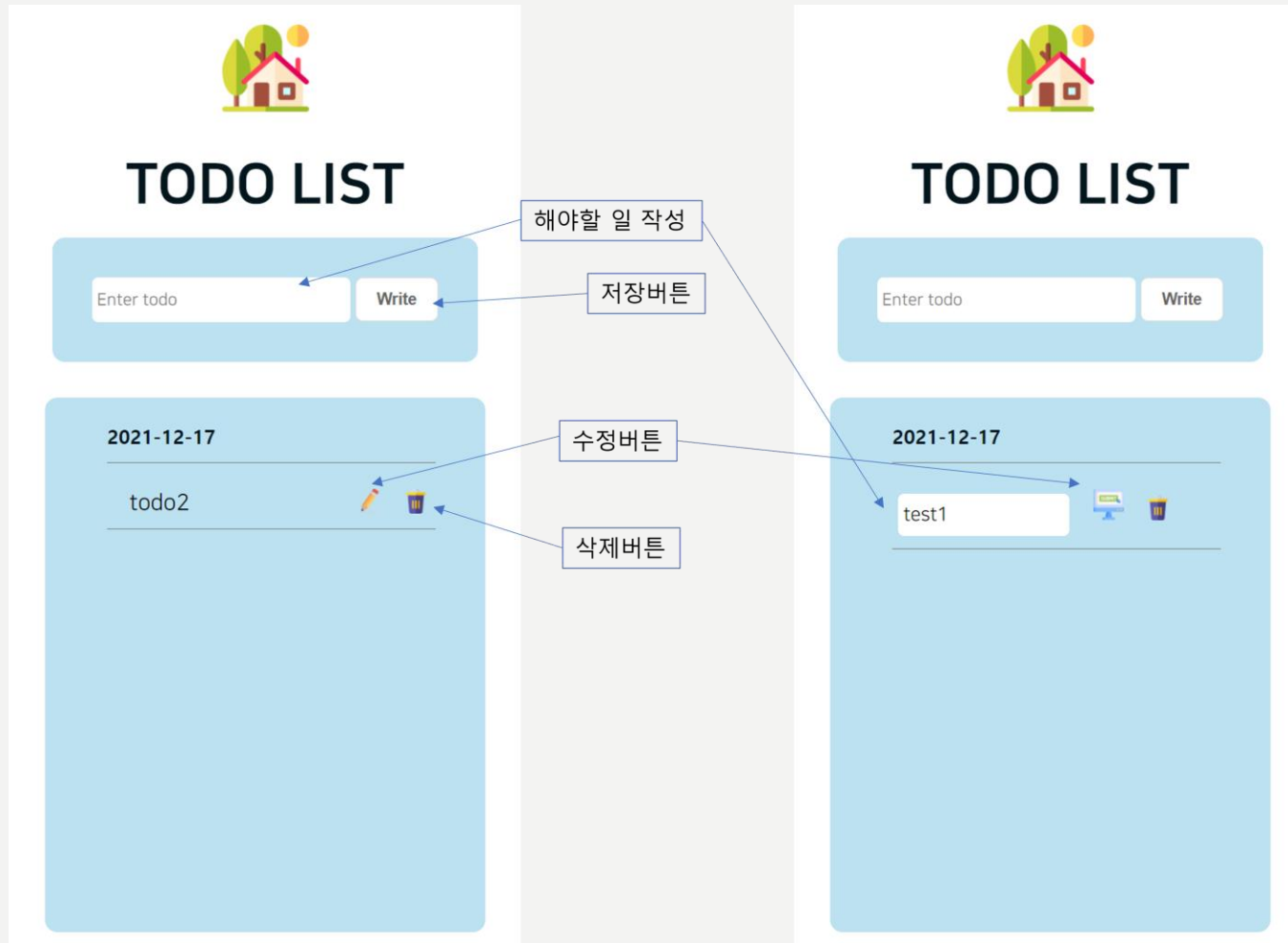
삭제 화면



TODO LIST

사용자 인터페이스 설계

3. 화면 개체 및 동작





THANK YOU