**MiniLib**

Mini Library for Me

[pdvegaa@gmail.com](mailto:pdvegaa@gmail.com)

[teza.harsony230394@gmail.com](mailto:teza.harsony230394@gmail.com)

**Menu**

**Show Books**

Read – 3 jam

Sort by BookTitle – 2 jam

Association many to many Authors to Books

Table Authors (id, first\_name:string, last\_name:string)

Table Books (id, title:string)

Table AuthorBooks (id, AuthorId:integer, BooksId:integer, genre:string)

Association one to many Publishers to Books

Table Publishers (id, name:string)

**Add Book**

Create – 3 jam

**Update Book**

Update – 3 jam

Delete Book – 1 jam

**Search**

Search Book by Author – 5 jam

**Search Book by Book title – 4 jam**

**Registrasi**

Create Akun – 3 jam

Table Account (id, name:string, phone:string, password:string, salt:string)

**Login**

Authentikasi – 2 jam

Session – 2 jam

Fitur Tambahan

Add book yang udah dibaca

Menyimpan data peminjam buku

Melihat buku yang dipinjam sama peminjam

Melihat data peminjam

**Progress :**

pada terminal di folder minilibme

express –view=ejs

npm install

npm init

npm install --save sequelize

npm install --save sequelize-cli

npm install --save pg pg-hstore

sequelize init

setting config environment development & production on config/config.json. Environment test tetap.

{

"development": {

"username": "postgres",

"password": 1234,

"database": "minilibme",

"host": "127.0.0.1",

"dialect": "postgres",

"logging": "false"

},

"test": {

"username": "root",

"password": null,

"database": "database\_test",

"host": "127.0.0.1",

"dialect": "mysql"

},

"production": {

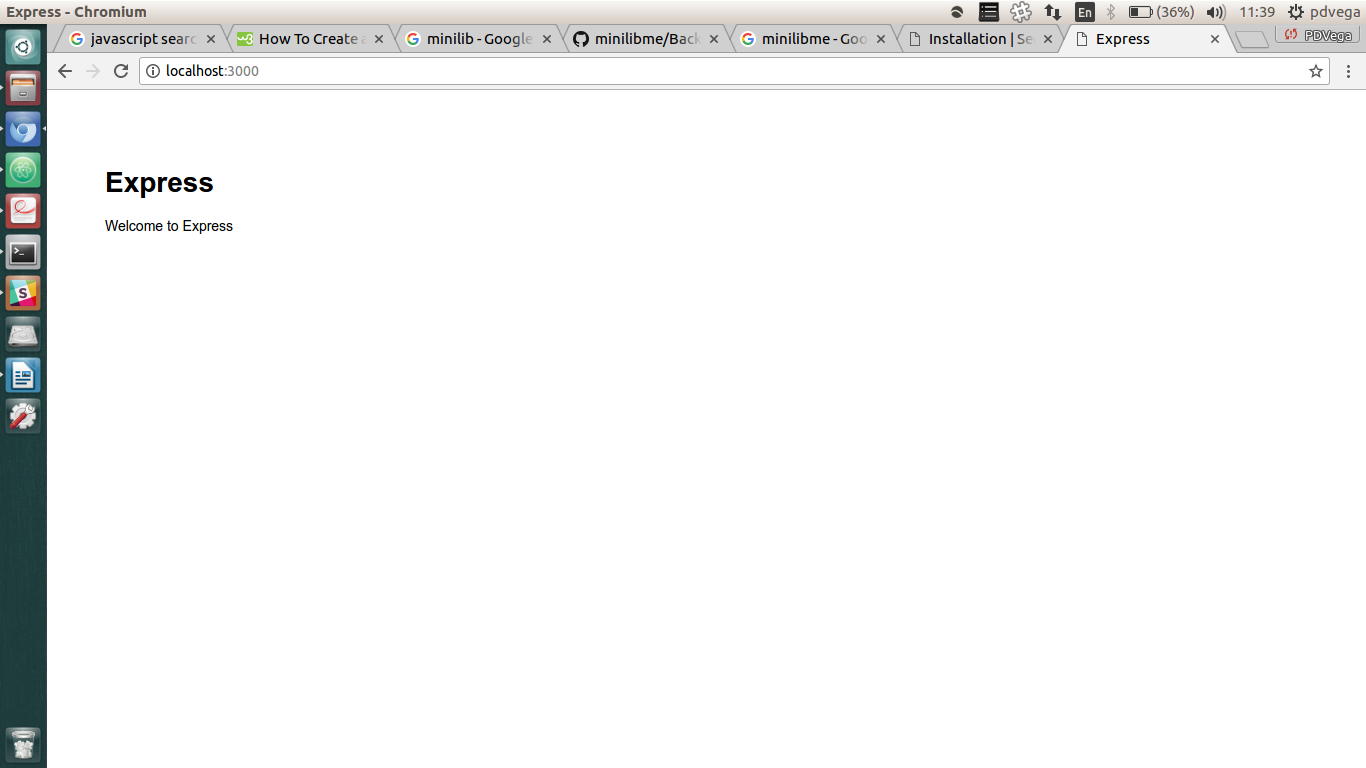
"use\_env\_variable": "DATABASE\_URL"

}

}

npm start to running minilibme

success access localhost:3000



touch .gitignore

write node\_modules inside file .gitignore

update pg 6.4.0 (sebelumnya 7.0.1) karena ga connect ke sequelize on package.json

{

"name": "minilibme",

"version": "0.0.0",

"private": true,

"scripts": {

"start": "node ./bin/www"

},

"dependencies": {

"body-parser": "~1.17.1",

"cookie-parser": "~1.4.3",

"debug": "~2.6.3",

"ejs": "~2.5.6",

"express": "~4.15.2",

"morgan": "~1.8.1",

"pg": "^6.4.1",

"pg-hstore": "^2.3.2",

"sequelize": "^4.3.2",

"sequelize-cli": "^2.7.0",

"serve-favicon": "~2.4.2"

},

"description": "Mini Library for Me - Your Personal Library Team Member : 1. Pella De Vega 2. Teza Harsony",

"main": "app.js",

"devDependencies": {},

"repository": {

"type": "git",

"url": "git+https://github.com/PDVega/minilibme.git"

},

"keywords": [

"pair",

"project"

],

"author": "pdvega & tezaharsony",

"license": "ISC",

"bugs": {

"url": "https://github.com/PDVega/minilibme/issues"

},

"homepage": "https://github.com/PDVega/minilibme#readme"

}

bisa juga dijalankan dari cli

npm remove --save pg

npm install --save pg@6

create database

psql postgres

postgrest=# CREATE DATABASE minilibme;

CREATE DATABASE (akan muncul tulisan seperti ini apabila berhasil create db)

postgrest=#\q (untuk kembali ke working directory kita)

sequelize help

sequelize help:model:create

sequelize model:create --name Author --attributes first\_name:string,last\_name:string

sequelize model:create --name Book --attributes title:string

sequelize model:create --name AuthorBook --attributes AuthorId:integer, BookId:integer,genre:string

sequelize model:create --name Publisher --attributes name:string

sequelize db:migrate

create routing :

/books (to show all books in library)

/routes/touch books.js (create file books in routes)

inside file books.js

const express = require('express')

const router = express.Router()

const db = require('../models')

router.get('/', (req, res) => {

res.send('books page okay')

})

module.exports = router;

add in file app.js

var express = require('express');

var path = require('path');

var favicon = require('serve-favicon');

var logger = require('morgan');

var cookieParser = require('cookie-parser');

var bodyParser = require('body-parser');

var index = require('./routes/index');

var users = require('./routes/users');

var books = require('./routes/books');

var app = express();

// view engine setup

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'ejs');

// uncomment after placing your favicon in /public

//app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico')));

app.use(logger('dev'));

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', index);

app.use('/users', users);

app.use('/books', books);

// catch 404 and forward to error handler

app.use(function(req, res, next) {

var err = new Error('Not Found');

err.status = 404;

next(err);

});

// error handler

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

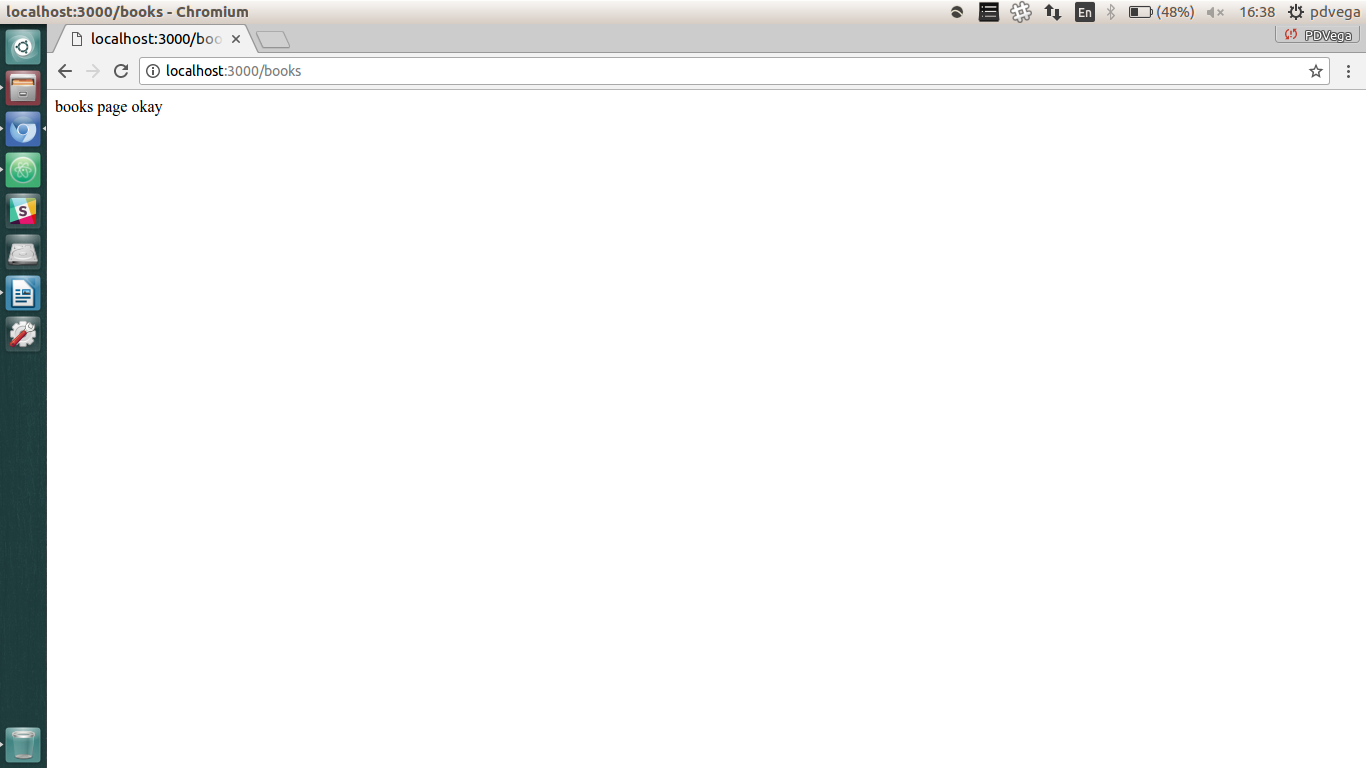
// render the error page

res.status(err.status || 500);

res.render('error');

});

module.exports = app;



add route /showbooks inside file books.js

const express = require('express')

const router = express.Router()

const db = require('../models')

router.get('/', (req, res) => {

res.send('books page okay')

})

router.get('/showbooks', (req, res) => {

db.Book.findAll()

.then(data\_book => {

res.render('showbooks', {data\_book : data\_book})

})

})

module.exports = router;

sequelize seed:create --name seedBooks

insert pada file /seeders/seedBooks.js berdasarkan data buku yang dibuat Teza.

'use strict';

module.exports = {

up: function (queryInterface, Sequelize) {

/\*

Add altering commands here.

Return a promise to correctly handle asynchronicity.

Example:

return queryInterface.bulkInsert('Person', [{

name: 'John Doe',

isBetaMember: false

}], {});

\*/

return queryInterface.bulkInsert('Books', [{

title : 'Anak Rantau',

createdAt : new Date(),

updatedAt : new Date()

},

{

title : 'Matlab Untuk Mahasiswa, Belajar Dari Berbagai Studi Kasus',

createdAt : new Date(),

updatedAt : new Date()

},

{

title : 'Konspirasi Alam Semesta',

createdAt : new Date(),

updatedAt : new Date()

},

{

title : 'Dasar-Dasar Endokrinologi',

createdAt : new Date(),

updatedAt : new Date()

},

{

title : 'Words Their Way: Word Study for Phonics, Vocabulary, and Spelling Instruction',

createdAt : new Date(),

updatedAt : new Date()

},

{

title : 'Harga Sebuah Percaya',

createdAt : new Date(),

updatedAt : new Date()

}], {})

},

down: function (queryInterface, Sequelize) {

/\*

Add reverting commands here.

Return a promise to correctly handle asynchronicity.

Example:

return queryInterface.bulkDelete('Person', null, {});

\*/

return queryInterface.bulkDelete('Books', null, {})

}

};

sequelize db:seed --seed seeders/20170719103326-seedBooks.js

|  |  |  |  |
| --- | --- | --- | --- |
| **Judul Buku** | **Penulis** | **Penerbit** | **Kategori** |
| Anak Rantau | Ahmad Fuadi | Falcon Publishing | Drama |
| ~~Matlab Untuk Mahasiswa, Belajar Dari Berbagai Studi Kasus~~  Feel Real | ~~R.H Sianipar~~  Radin Azkia | ~~Andi Publisher~~  GagasMedia | ~~Tutorial~~  Drama |
| Konspirasi Alam Semesta | Fiersa Besari | MediaKita | Puisi |
| Dasar-Dasar Endokrinologi | Alwi Shahab | Rayyana Komunikasindo | Kedokteran |
| [~~Words Their Way: Word Study for Phonics, Vocabulary, and Spelling Instruction~~](http://opentrolley.co.id/Book_Detail.aspx?EAN=9780133996333)  From Zero To A Pro: Java Script Dan jQuery | [~~Bear, Donald R,~~](http://opentrolley.co.id/author/bear-donald-r-)  [~~Invernizzi, Marcia R,~~](http://opentrolley.co.id/author/invernizzi-marcia-r-)  [~~Templeton, Shane~~](http://opentrolley.co.id/author/templeton-shane)  Abdul Kadir | ~~Pearson~~  Andi Publisher | ~~Edukasi~~  Edukasi |
| Harga Sebuah Percaya | Tere Liye | Republika | Fiksi |

Create file /views/showbooks.ejs dan code untuk show table book

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Show Book</title>

</head>

<body>

<table>

<thead>

<tr>

<th>ID</th>

<th>Judul Buku</th>

<th>Pengarang</th>

<th>Penerbit</th>

</tr>

</thead>

<tbody>

<% data\_book.forEach(book => { %>

<tr>

<td><%= book.id %></td>

<td><%= book.title %></td>

<td>Masih Ngarang</td>

<td>Belum Terbit</td>

</tr>

<% }) %>

</tbody>

</table>

</body>

</html>

http://searchoracle.techtarget.com/answer/Books-and-authors-many-to-many

sequelize seed:create --name seedAuthors

insert pada file /seeders/seedAuthor.js berdasarkan data buku yang dibuat Teza.

'use strict';

module.exports = {

up: function (queryInterface, Sequelize) {

/\*

Add altering commands here.

Return a promise to correctly handle asynchronicity.

Example:

return queryInterface.bulkInsert('Person', [{

name: 'John Doe',

isBetaMember: false

}], {});

\*/

return queryInterface.bulkInsert('Authors', [{

first\_name : 'Ahmad',

last\_name : 'Fuadi',

createdAt : new Date(),

updatedAt : new Date()

},

{

first\_name : 'Radin',

last\_name : 'Azkia',

createdAt : new Date(),

updatedAt : new Date()

},

{

first\_name : 'Fiersa',

last\_name : 'Besari',

createdAt : new Date(),

updatedAt : new Date()

},

{

first\_name : 'Alwi',

last\_name : 'Shahab',

createdAt : new Date(),

updatedAt : new Date()

},

{

first\_name : 'Abdul',

last\_name : 'Kadir',

createdAt : new Date(),

updatedAt : new Date()

},

{

first\_name : 'Tere',

last\_name : 'Liye',

createdAt : new Date(),

updatedAt : new Date()

}], {});

},

down: function (queryInterface, Sequelize) {

/\*

Add reverting commands here.

Return a promise to correctly handle asynchronicity.

Example:

return queryInterface.bulkDelete('Person', null, {});

\*/

return queryInterface.bulkDelete('Authors', null, {});

}

};

sequelize db:seed --seed seeders/20170719121228-seedAuthor.js

sequelize seed:create --name seedPublishers

insert pada file /seeders/seedPublishers.js berdasarkan data buku yang dibuat Teza.

'use strict';

module.exports = {

up: function (queryInterface, Sequelize) {

/\*

Add altering commands here.

Return a promise to correctly handle asynchronicity.

Example:

return queryInterface.bulkInsert('Person', [{

name: 'John Doe',

isBetaMember: false

}], {});

\*/

return queryInterface.bulkInsert('Publishers', [{

name : 'Falcon Publishing',

createdAt : new Date(),

updatedAt : new Date()

},

{

name : 'GagasMedia',

createdAt : new Date(),

updatedAt : new Date()

},

{

name : 'MediaKita',

createdAt : new Date(),

updatedAt : new Date()

},

{

name : 'Rayyana Komunikasindo',

createdAt : new Date(),

updatedAt : new Date()

},

{

name : 'Andi Publisher',

createdAt : new Date(),

updatedAt : new Date()

},

{

name : 'Republika',

createdAt : new Date(),

updatedAt : new Date()

}], {});

},

down: function (queryInterface, Sequelize) {

/\*

Add reverting commands here.

Return a promise to correctly handle asynchronicity.

Example:

return queryInterface.bulkDelete('Person', null, {});

\*/

return queryInterface.bulkDelete('Publishers', null, {});

}

};

sequelize db:seed --seed seeders/20170719122620-seedPublisher.js

setting association in models.

One Publisher has many Book

add in /models/publisher.js

'use strict';

module.exports = function(sequelize, DataTypes) {

var Publisher = sequelize.define('Publisher', {

name: DataTypes.STRING

});

Publisher.associate = models => {

Publisher.hasMany(models.Book);

}

return Publisher;

};

add column PublisherId in /models/book.js

sequelize migration:create --name addColumn\_PublisherId\_in\_Book

edit file /migrations/20170719130819-addColumn\_PublisherId\_in\_Book

'use strict';

module.exports = {

up: function (queryInterface, Sequelize) {

/\*

Add altering commands here.

Return a promise to correctly handle asynchronicity.

Example:

return queryInterface.createTable('users', { id: Sequelize.INTEGER });

\*/

return queryInterface.addColumn('Books', 'PublisherId', Sequelize.INTEGER);

},

down: function (queryInterface, Sequelize) {

/\*

Add reverting commands here.

Return a promise to correctly handle asynchronicity.

Example:

return queryInterface.dropTable('users');

\*/

return queryInterface.removeColumn('Books', 'PublisherId');

}

};

edit file /models/book.js untuk menambahkan PublisherId dan menambahkan association (relation)

'use strict';

module.exports = function(sequelize, DataTypes) {

var Book = sequelize.define('Book', {

title: DataTypes.STRING,

PublisherId : DataTypes.INTEGER

});

Book.associate = models => {

Book.belongsTo(models.Publisher);

}

return Book;

};

lakukan migrate untuk penambahan kolom baru (PublisherId)

sequelize db:migrate --name migrations/20170719130819-addColumn\_PublisherId\_in\_Book

.js

setting relation /models/author.js

'use strict';

module.exports = function(sequelize, DataTypes) {

var Author = sequelize.define('Author', {

first\_name: DataTypes.STRING,

last\_name: DataTypes.STRING

});

Author.associate = models => {

Author.belongsToMany(models.Book, {

through : 'AuthorBook'

})

}

return Author;

};

setting relation /models/authorbookjs

'use strict';

module.exports = function(sequelize, DataTypes) {

var AuthorBook = sequelize.define('AuthorBook', {

AuthorId: DataTypes.INTEGER,

BookId: DataTypes.INTEGER,

genre: DataTypes.STRING

});

AuthorBook.associate = models => {

AuthorBook.belongsTo(models.Author);

AuthorBook.belongsTo(models.Book);

}

return AuthorBook;

};

create routing author

create file /routes/authors.js

const express = require('express');

const router = express.Router();

const db = require('../models')

router.get('/', (req, res) => {

res.send('author page ok')

})

module.exports = router;

create routing authors/author (show author in library)

add code in file /routes/authors.js

router.get('/', (req, res) => {

db.Author.findAll()

.then(data\_authors => {

res.render('author', {data\_authors : data\_authors})

})

})

create views/author.ejs

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Author</title>

</head>

<body>

<table>

<thead>

<tr>

<th>ID</th>

<th>First Name</th>

<th>Last Name</th>

</tr>

</thead>

<tbody>

<% data\_authors.forEach(author =>{ %>

<tr>

<td><%= author.id %></td>

<td><%= author.first\_name %></td>

<td><%= author.last\_name %></td>

</tr>

<% }) %>

</tbody>

</table>

</body>

</html>

create routing authors/addauthor (show form to add author)

add code in file /routes/authors.js

router.post('/addauthor', (req, res, next) => {

let first\_name = req.body.first\_name

let last\_name = req.body.last\_name

db.Author.create({

'first\_name' : first\_name,

'last\_name' : last\_name

})

.then(data\_author => {

res.redirect('/authors')

})

})

create views/addauthor.ejs

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Add Author</title>

</head>

<body>

<form class="" action="/authors/addauthor" method="post">

<label for="">First Name</label>

<input type="text" name="first\_name" value="">

<label for="">Last Name</label>

<input type="text" name="last\_name" value="">

<button type="submit" name="button">Add Author</button>

</form>

</body>

</html>

create routing authors/edit/:id (show data to edit from author take id when clicked)

add code in file /routes/authors.js

router.get('/edit/:id', (req, res, next) => {

let id = req.params.id

db.Author.findById(id)

.then(data\_author => {

res.render('editauthor', {data\_author : data\_author})

})

})

add code views/author.ejs (to adding link to update and delete)

<td><a href="/authors/edit/<%=author.id%>">Update</a> || <a href="/authors/delete/<%= author.id%>">Delete</a></td>

create file views/editauthor.ejs

create form for edit

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Edit Author</title>

</head>

<body>

<form class="" action="/authors/edit/<%= data\_author.id%>" method="post">

<label for="">First Name</label>

<input type="text" name="first\_name" value="<%= data\_author.first\_name%>">

<label for="">Last Name</label>

<input type="text" name="last\_name" value="<%= data\_author.last\_name%>">

<button type="submit" name="button">Update Author</button>

</form>

</body>

</html>

add code in file /routes/authors.js (post form to update author)

//routing post to input database update data input from form editauthor

router.post('/edit/:id', (req, res, next) => {

let id = req.params.id

let first\_name = req.body.first\_name

let last\_name = req.body.last\_name

db.Author.findById(id)

.then(data\_author => {

data\_author.update({

'first\_name' : first\_name,

'last\_name' : last\_name

})

.then(() => {

res.redirect('/authors')

})

})

})