**Backend AdonisJS Documentation**

**Menyiapkan Project**

1. pindah ke folder tempat projek akan dibuat

cd C:\Users\ASUS\Desktop\Project

1. Install adonis project

npm init adonis-ts-app@latest backend-adonisjs

“backend-adonisjs” = nama project

setup eslint false aja biar gak galak

node ace –help 🡪 melihat pilihan perintah

1. Start server nya

node ace serve --watch

buka di 127.0.0.1:3333

**Membuat Controller**

Sebagai logic utama yang berisi logic setelah menerima request API

1. node ace make:controller Venue

“Venue” = nama controller, harus dalam bentuk singular

node ace list:routes 🡪 melihat routes yang ada

1. Has many, oleh/di sebuah venue (parent) membuat field (child):

const venue = await Venue.findOrFail(1) // tentukan venue pembuat fieldnya, akan tercatat primary key (id) nya pada tabel field

const newField = new Field()

newField.name = request.input('name')

newField.type = request.input('type')

await venue.related('fields').save(newField) // membuat field baru by a certain venue. 'fields' adalah nama foreign key yg dideclare di hasMany model Venue

1. Belongsto, menampilkan punya bookings (child) apa saja sebuah field (parent), dan ada di venue (grandparent) apa:

const fields = await Field.query().preload('bookings').preload('venue') // . 'bookings' adalah nama foreign key yg dideclare di hasMany model Field. 'venue' adalah nama foreign key yg dideclare di belongsTo model Field

1. Many to many, Mengsinkronkan pivot table antara seorang user dan sebuah booking

const loginUserId = auth.user?.id //user id

const booking = await Booking.findOrFail(params.id) //booking id

await booking.related('users').sync([loginUserId]) //sinkronkan pivot table kolom id user dan id booking

1. Many to many, melihat seorang user sinkron dengan rows apa saja di table bookingconst loginUserId = auth.user?.id //user id

const schedule = await Booking.query().whereHas('users', usersQuery => usersQuery.wherePivot('user\_id', loginUserId)) //”users” adalah foreign key yg dideclare di manyToMany model Booking

**Seputar API**

1. params

params = url tambahan pada GET, ditambah ':' pada route

“params” mengecek parameter bagian dari url, biasanya “id” pada “venues/:id”

1. request

request.qs() 🡪 query parameter pada request, biasanya pada GET

request.input('nama key') 🡪 body pada request, biasanya pada POST

1. response

response.status(201).json({message: “success”, data: variableData})

1. auth

const loginUserId = auth.user?.id

**Membuat validator**

Untuk mengecek apakah request input body API sudah memenuhi syarat validasi

1. node ace make:validator User

“User” = nama validator

**Membuat ORM**

ORM = Object Relation Manager. Menjadikan 1 row/instance data sebagai 1 object. Di express Namanya sequelize, di adonis namanya lucid

1. npm i @adonisjs/lucid
2. configure

node ace configure @adonisjs/lucid

pilih mysql (untuk development pakai mysql xampp) dan pg (untuk production pakai pq heroku)

pilih tampilkan informasi yang harus dicopy di terminal

1. copy text di terminal ke projek

ke file env.ts:

DB\_CONNECTION: Env.schema.string(),

MYSQL\_HOST: Env.schema.string({ format: 'host' }),

MYSQL\_PORT: Env.schema.number(),

MYSQL\_USER: Env.schema.string(),

MYSQL\_PASSWORD: Env.schema.string.optional(),

MYSQL\_DB\_NAME: Env.schema.string(),

PG\_HOST: Env.schema.string({ format: 'host' }),

PG\_PORT: Env.schema.number(),

PG\_USER: Env.schema.string(),

PG\_PASSWORD: Env.schema.string.optional(),

PG\_DB\_NAME: Env.schema.string(),

Ke file config/database.ts:

connection: Application.inDev ? 'mysql' : 'pg' ,

atur port jika pake mac

node ace configure @adonisjs/lucid

1. install driver mysql dan pq

npm i mysql

npm i pg

1. buat database di phpMyAdmin, tentukan nama databasenya
2. untuk development, buat database di phpmyadmin, tentukan nama databasnya

ke file .env:

MYSQL\_USER=root (sesuai user mysql xampp)

MYSQL\_PASSWORD= (sesuai password mysql xampp)

MYSQL\_DB\_NAME=backend-adonis (sesuai nama database di phpMyAdmin)

**Membuat Table Migration**

Table migration adalah table dan kolom-kolom apa saja yang akan dibuat di database (mysql dan pg)

1. membuat migration/membuat tabel baru bernama table users

node ace make:migration users

“users” = nama table

pada adonis terbaru, pada table, use:

table.timestamps(true, true)

instead of:

table.timestamp('created\_at', { useTz: true })

table.timestamp('updated\_at', { useTz: true })

contoh foreign key : table.integer('user\_id').unsigned().references('users.id').onDelete('CASCADE')

1. push ke database

node ace migration:run

1. hapus push terakhir dari database

node ace migration:rollback

1. mengecek status migration database

node ace migration: status

1. overwrite/menambah kolom table tanpa perlu rollback

node ace make:migration add\_phone\_to\_column --table=users

pada file migration baru, tambahkan add kolom baru pada fungsi up dan remove kolom baru pada fungsi down

1. untuk membuat relasi antar table, pastikan urutan pembuatan table benar

kita bisa merekayasa itu dengan mengedit nama file migration yang dibuat berdasarkan waktu

**Membuat Model Object Data**

Pada ORM, table data = object model. 1 row data = 1 instance object model.

1. membuat model

node ace make:model Venue

“Venue” = nama model dari table “venues”. awali dengan kapital dan harus singular

akan ada model Venue pada project yang terhubung dengan tabel venues di phpMyAdmin

1. lengkapi setiap model

tambahkan deklarasi property objek model sesuai dengan attribute tabel di migration table

tentukan property-properti has one, has many, belongto dsb di model ini

**Membuat Auth**

Auth berfungsi sebagai kunci untuk masuk suatu request API. Biasanya dipasang dalam table users, sehingga setiap user memiliki 1 auth bearer

1. menginstall modul auth

npm i @adonisjs/auth

1. konfigurasi auth

node ace configure @adonisjs/auth

provider: lucid

guard: api

model name: User

create migration for User: y (kalau belum bikin migration table users)

provider: database

create migration for provider: y

1. run model yg baru dibuat

pada migration api\_token tambahkan:

table.timestamp('expires\_at', { useTz: true}).nullable()

node ace migration:run

akan ada model baru bernama User yang di dalamnya ada method hash untuk password.

saat bikin fungsi register, maka User.password akan terinkripsi di database

1. install bycript sebagai modul enkripsi password

npm install phc-bcrypt

di file .env:

HASH\_DRIVER=bcrypt

1. menambahkan auth sebagai middleware

pada file start/kernel.ts:

Server.middleware.registerNamed({

auth: () => import('App/Middleware/Auth')

})

1. pada route tambahkan

pada file start/routes.ts:

Route......middleware('auth')

1. pada saat menembak api, tambahkan token hasil login

auth --> bearer token --> paste token

**ORM relation**

One to one, one to many, many to many. Atur hubungan ORM

1. Jika one to one atau one to many, tabel/model Venue has many tabel/model Field:

di migration parent:

table.increments('id').primary()

di migration child:

table.integer('venue\_id').unsigned().references('id').inTable('venues').onDelete('CASCADE')

di model parent (Venue):

import Field from 'App/Models/Field'

@hasMany(() => Field, {

foreignKey: 'venueId', // defaults to venueId

})

public field: HasMany<typeof Field>

di model child (Field):

import Venue from 'App/Models/Venue'

@column()

public venueId: number

@belongsTo(() => Venue

public venue: BelongsTo<typeof Venue

1. Jika many has many:

buat migration baru pivot table table1\_table2, didalamnya:

tambahkan 2 foreign key masing2 reference ke masing2 tabel

table.integer('user\_id').unsigned().references('users.id')

table.integer('skill\_id').unsigned().references('skills.id')

table.unique(['user\_id', 'skill\_id']) //opsional

tidak usah bikin model pivot table. di parent tabel yg dihubungkan:

@manyToMany(() => Skill)

public skills: ManyToMany<typeof Skill>

1. untuk one to one or one to many, preload untuk load data dari tabel lain yg berelasi

const venue = await Venue.query().preload('field').where('id', params.id)

1. untuk many to many, pakai attach

const user = await User.find(1)

const skill = await Skill.find(1)

await user.related('skills').attach([skill.id])

**Membuat Adonis Mail**

Adonis mail biasanya untuk mengirim kode OTP

1. menginstall modul mail

npm i @adonisjs/mail

node ace configure @adonisjs/mail --> pilih SMTP, terminal

ke file env.ts;

SMTP\_HOST: Env.schema.string({ format: 'host' }),

SMTP\_PORT: Env.schema.number(),

SMTP\_USERNAME: Env.schema.string(),

SMTP\_PASSWORD: Env.schema.string(),

1. configura mailtrap, copykan host, port, user, password ke file .env
2. menginstall adonis view, terpakai untuk menampilkan halaman setelah mengirim mail

npm i @adonisjs/view

node ace configure @adonisjs/view

membuat view utk email:

node ace make:view emails/otp\_verification

**Membuat Middleware**

Middleware sebagai gerbang diantara routes, penyambung antara request api dengan modul controller. Contohnya auth. Middleware bisa juga untuk menangkap ip address dari request API. Middleware juga bisa untuk mengecek role user mana yang sebagai admin, user, dsb, sehingga bisa diberi perlakuan khusus.

1. menginstall modul mail

node ace make:middleware verify

“verify” = nama middleware yang nanti dibubuhkan di akhir route.

**Membuat dokumentasi API dengan Swagger**

Swagger bisa digunakan untuk membuat dokumntasi API.

1. menginstall modul swagger.

npm i --save adonis5-swagger

1. Compile your code (jika belum)

node ace serve --watch

1. Connect all dependences:

node ace invoke adonis5-swagger

1. Tambahkan dokumentasi swagger pada setiap controller dan modul

Jika ada middleware auth, tambahkan ke file config/swagger.ts:

securitySchemes: {

bearerAuth: {

type: 'http',

scheme: 'bearer'

}

}

1. Sebelum deploy ke production, generate dokumentasi swagger:

Bikin folder “docs” di root folder projek

Ke file config/swagger.ts:

const swaggerConfig = {

specFilePath: 'docs/swagger.json'

}

1. Generate dokumentasi swagger:

node ace swagger:generate

nanti ada file swagger.json tergenerate di folder docs.

Bisa dilihat di local/docs atau Heroku/docs

1. Sebelum deploy Heroku, update file

ke file config/swagger.ts:

const swaggerConfig = {

mode: process.env.NODE\_ENV === 'production' ? 'PRODUCTION' : 'RUNTIME',

specFilePath: 'docs/swagger.json'

}

ke file package.json:

{

"scripts": {

"build": "npm run compile",

"postbuild": "node ace swagger:generate && cp -a docs/ build/docs"

}

}

**Deploy ke Heroku**

Heroku adalah web hosting gratis

1. menginstall Heroku CLI.

https://devcenter.heroku.com/articles/heroku-cli

1. Buat projek baru di Heroku, beri nama unik contoh: backend-adonisjs

Create new App

1. login Heroku

heroku login

1. arahkan cmd ke projek adonis, pastikan project terpasang Git, set project Adonis agar memiliki alamat remote repository di cloud Heroku:

cd direktoriprojek

heroku git:remote -a backend-adonisjs

1. Update file package.json:

"scripts": {

"build": "node ace build --prod --ignore-ts-errors",

"start": "node ace serve --watch",

"start:prod": "node ./build/server.js",

"postbuild": "cp -a docs/ build/docs"

}

Postbuild untuk swagger docs, --ignore-ts-errors utk ignore ts yg teralu strict

Perlu diingat bahwa projek adonis ditulis dana ts, sementara yg dideploy harus dalam js.

Maka prosesnya adalah projek ts dibuild, lalu yg dibuild dideploy dan diaktifkan pada server

1. Buat file procfile (tanpa extension) berisi:

release: ENV\_SILENT=true node build/ace migration:run --force && node build/ace db:seed

web: npm run start:prod

atau

web: node build/server.js

release: node build/ace migration:run –force

1. Install add-ons "Heroku Postgres" di dashboard heroku

lihat setting postgre Heroku dan copykan data2 ke .env, contohnya:

PG\_HOST=ec2-34-205-14-168.compute-1.amazonaws.com

PG\_PORT=5432

PG\_USER=bbngigemodogzq

PG\_PASSWORD=a9c54c4371bfaf7e613d915cad6a2f8b7a3f572378a8920deaa86b2419b7e613

PG\_DB\_NAME=dq8gof7fi8bek

1. Update isi file config/database.ts

import Application from '@ioc:Adonis/Core/Application'

connection: Application.inDev ? 'mysql' : 'pg'

pg: {

client: 'pg',

connection: {

// ....

ssl: {

rejectUnauthorized: false

}

}

}

1. Tambahkan Convig vars pad Heroku dengan value sesuai file .env

Copykan semua, kecuali HOST tidak usah diisi atau diisi 80

APP\_KEY bisa degenerate ulang dengan perintah: node ace generate:key

1. Deploy

git add .

git commit -m "ready to deploy"

git push origin main

git push heroku main

1. Run server yang sudah dideploy

cd build

npm ci –production

node server.js