/src/controllers/catsContrller.js

```
const { HttpCode } = require("../helpers/constants");
const { CatsService } = require("../services");

const catsService = new CatsService()

const getAll = (req, res, next) => {
  try{
    const cats = catsService.getAll()
    res.status(HttpCode.OK).json({
      status: 'success',
      code: HttpCode.OK,
      data: {
        cats
      }
    })
  } catch(e) {
    next(e)
  }
};

const getById = (req, res, next) => {
  try{
    const cat = catsService.getById(req.params)
    if(cat){
      return res.status(HttpCode.OK).json({
        status: 'success',
        code: HttpCode.OK,
        data: {
          cat
        }
      })
    }else{
      // пробрасываю в app -> app.use((err, req, res, next) => {....
      return next({
        status: HttpCode.NOT_FOUND,
        code: HttpCode.NOT_FOUND,
        message: 'Cat not found',
        data: 'Not found'
      })
    }
  } catch(e) {
    next(e)
  }
};

const update = (req, res, next) => {
  try{
    const cat = catsService.update(req.params, req.body)
    if(cat.id){
      return res.status(HttpCode.OK).json({
        status: 'success',
        code: HttpCode.OK,
        data: {
          cat
        }
      })
    }else{
      // пробрасываю в app -> app.use((err, req, res, next) => {....
      return next({
        status: HttpCode.NOT_FOUND,
        code: HttpCode.NOT_FOUND,
        message: 'Cat not found',
        data: 'Not found'
      })
    }
  } catch(e) {
    next(e)
  }
};
```

3

```
const create = (req, res, next) => {
  try{
    const cat = catsService.create(req.body)
    res.status(HttpCode.CREATED).json({
      status: 'success',
      code: HttpCode.CREATED,
      data: {
        cat
      }
    })
  } catch(e) {
    next(e)
  }
};

const updateStatus = (req, res, next) => {
  try{
    const cat = catsService.update(req.params, req.body)
    if(cat){
      return res.status(HttpCode.OK).json({
        status: 'success',
        code: HttpCode.OK,
        data: {
          cat
        }
      })
    }else{
      // пробрасываю в app -> app.use((err, req, res, next) => {....
      return next({
        status: HttpCode.NOT_FOUND,
        code: HttpCode.NOT_FOUND,
        message: 'Cat not found',
        data: 'Not found'
      })
    }
  } catch(e) {
    next(e)
  }
};

const remove = (req, res, next) => {
  try{
    const cat = catsService.remove(req.params)
    if(cat){
      return res.status(HttpCode.OK).json({
        status: 'success',
        code: HttpCode.OK,
        data: {
          cat
        }
      })
    }else{
      // пробрасываю в app -> app.use((err, req, res, next) => {....
      return next({
        status: HttpCode.NOT_FOUND,
        code: HttpCode.NOT_FOUND,
        message: 'Cat not found',
        data: 'Not found'
      })
    }
  } catch(e) {
    next(e)
  }
};

module.exports = {
  getAll, getById, create, update, updateStatus, remove,
};
```

/api/cats/index.js

```
const express = require("express");
const controllerCats = require('../../controllers/catsContrller')
const router = express.Router();

const {
  validateCreateCat,
  validateUpdateCat,
  validateUpdateStatusCat
} = require('../../validation/catsValidation')

router
  .get("/", controllerCats.getAll)
  .get("/:id", controllerCats.getById)
  .post("/", validateCreateCat, controllerCats.create)
  .put("/:id", validateUpdateCat, controllerCats.update)
  .patch("/:id/vaccinated", validateUpdateStatusCat, controllerCats.
updateStatus)
  .delete("/:id", controllerCats.remove);
```

2

app.js

```
const express = require('express')
const cors = require('cors')
const { HttpCode } = require('./helpers/constants')
const routerCats = require('./api/cats')

const app = express()

app.use(cors())
app.use(express.json())

app.use('/api/cats', routerCats)

app.use((req, res, next) => {
  res.status(HttpCode.NOT_FOUND).json({
    status: 'error',
    code: HttpCode.NOT_FOUND,
    message: `Use api on routes ${req.baseUrl}/api/cats`,
    data: `Not Found`
  })
})

app.use((err, req, res, next) => {
  err.status = err.status ? err.status : HttpCode.INTERNAL_SERVER_ERROR
  res.status(err.status).json({
    status: err.status === HttpCode.INTERNAL_SERVER_ERROR ? 'fail' :
'error',
    code: err.status,
    message: err.message,
    data: err.status === HttpCode.INTERNAL_SERVER_ERROR
                        ? 'INTERNAL SERVER ERROR' : err.data,
  })
})

const PORT = process.env.PORT || 3000
app.listen(PORT, () => {
```

localhost:3000/api/cats/id-cat

1

/src/validation/catsValidation.js

```
const Joi = require("joi");
const { HttpCode } = require("../helpers/constants");

const schemaCreateCat = Joi.object({
  name: Joi.string().alphanum().min(2).max(30).required(),
  age: Joi.number().integer().min(0).max(25).required(),
  isVaccinated: Joi.boolean().optional(),
});

const schemaUpdateCat = Joi.object({
  name: Joi.string().alphanum().min(2).max(30).optional(),
  age: Joi.number().integer().min(0).max(25).optional(),
  isVaccinated: Joi.boolean().optional(),
});

const schemaUpdateStatusCat = Joi.object({
  isVaccinated: Joi.boolean().required()
});


const validate = (schema, body, next) => {
  const { error } = schema.validate(body)
  if(error){
    const [{message}] = error.details
    console.log(error.details);
    return next({
      status: HttpCode.BAD_REQUEST,
      code: HttpCode.BAD_REQUEST,
      message: `Field: ${message.replace(/"/g, '')}`,
      data: 'Bad request'
    })
  }
  next()
}

module.exports.validateCreateCat = (req, res, next) => {
  return validate(schemaCreateCat, req.body, next)
}
module.exports.validateUpdateCat = (req, res, next) => {
  return validate(schemaUpdateCat, req.body, next)
}
module.exports.validateUpdateStatusCat = (req, res, next) => {
  return validate(schemaUpdateStatusCat, req.body, next)
}
```

id-cat

/src/helpers/constants.js

```
const HttpCode = {
  OK: 200,
  CREATED: 201,
  BAD_REQUEST: 400,
  NOT_FOUND: 404,
  INTERNAL_SERVER_ERROR: 500
}

module.exports = { HttpCode }
```

/src/repositories/index.js

```
const CatsRepo = require('./catsRepo')
```

/src/repositories/catsRepo.js

```
const { v4: uuid } = require("uuid");
const db = require("../db");

class CatsRepo {
  constructor() {}

  getAll() {
    return db.get("cats").value();
  }

  getById(id) {
    return db.get("cats").find({ id }).value();
  }

  create(body) {
    const id = uuid();
    const record = {
      id,
      ...body,
      ...(body.isVaccinated ? {} : { isVaccinated: false }),
    };
    db.get("cats").push(record).write();
    return record;
  }
```

5

```
  update(id, body) {
    const record = db.get("cats").find({ id }).assign({...body}).value();
    db.write();
    return record;
  }

  updateStatus({ id }, body) {
    const record = db.get("cats").find({ id }).assign({...body}).value();
    db.write();
    return record;
  }

  remove(id) {
    const [record] = db.get("cats").remove({ id }).write();
    return record;
  }
}

module.exports = CatsRepo;
```

/src/services/index.js

```
const CatsService = require('./catsService')
module.exports = { CatsService }
```

/src/services/catsService.js

```
const { CatsRepo } = require('../repositories')

class CatsService {
  constructor(){
    this.repo = {
      cats: new CatsRepo()
    }
  }

  getAll(){
    return this.repo.cats.getAll()
  }

  getById({id}){
    return this.repo.cats.getById(id)
  }

  create(body){
    return this.repo.cats.create(body)
  }
```

4

```
  update({id}, body){
    return this.repo.cats.update(id, body)
  }

  updateStatus({id}, body){
    return this.repo.cats.updateStatus(id, body)
  }

  remove({id}){
    return this.repo.cats.remove(id)
  }
}

module.exports = CatsService
```

/src/db/index.js

```
const low = require("lowdb");
const path = require("path");
const FileSync = require("lowdb/adapters/FileSync");

const adapter = new FileSync(
  path.join(__dirname, "..", "..", "data", "db.json")
);
const db = low(adapter);

// Set some defaults (required if your JSON file is empty)
db.defaults({ cats: [] }).write();

module.exports = db;
```

6

```
data
  db.json
src
  api
    cats
      index.js
    users
  controllers
    catsController.js
  db
    index.js
  helpers
    constants.js
  repositories
    catsRepo.js
    index.js
  services
    catsService.js
    index.js
  validation
    catsValidation.js
  app.js
```