Лекция 12: NodeJS

Web-программирование / ПГНИУ

JavaScript

- Мульти-парадигменный (объектно-ориентированный, императивный, функциональный)
- Интерпретируемый (или JIT-компилируемый)
- Прототипно-ориентированный
- Событийно-ориентированный
- Динамическая слабая типизация
- Автоматическое управление памятью
- С-подобный синтаксис

NodeJS

- **Node** или **Node.js** — программная платформа, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. (wikipedia)
- **NodeJS** – среда исполнения JavaScript и API для взаимодействия с ОС
- Позволяет писать на **JS** консольные утилиты, серверную часть, десктопные приложения, serverless и т.д.

NodeJS для Full-Stack разработка

- Один язык
- Одна экосистема (npm, eslint и т.д.)
- Переиспользование кода
- SSR - Server-Side Rendering
- **Back-End For Frontend**

NodeJS в нагруженном приложении

- За счёт асинхронности держит высокую нагрузку на I/O
- Отлично справляется с realtime приложениями
- Лёгкий runtime, легко масштабируется экземплярами приложения

Проблемы NodeJS на Back-End

- Не для тяжёлых операций
- Молодой, экосистема развивается
  - Нет одного "default" фреймворка
  - Нет фреймворка, где всё, что можно из коробки
  - Все ORM не идеальные
  - Язык развивается быстрее библиотек

Основные фреймворки

- Микрофреймворки
  - **Express**
  - Koa
  - Fastify
- Фреймворки
  - **Nest**
  - Loopback
  - Adonis

Express

- Fast, unopinionated, minimalist web framework
- Самое популярное решение под NodeJS
- Микрофреймворк
- Основа - роутинг + посредники
- Посредник меняет объект запроса и ответа

Express и экосистема

- express
- body-parser, multer
- express-session, cookie-parser
- passport
- csurf, helmet, morgan

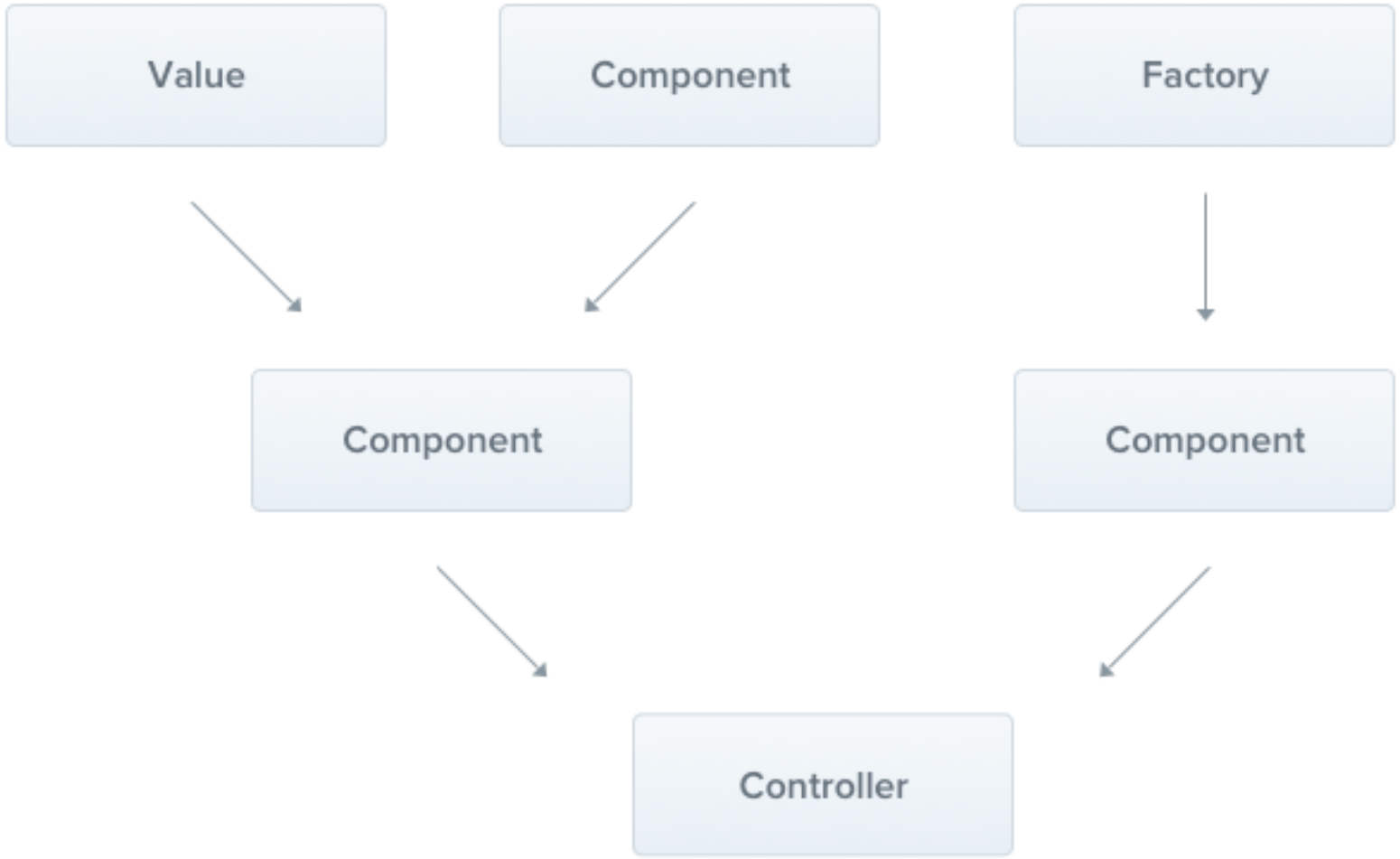
```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(3000, () => {
  console.log(`Listening at http://localhost:${port}`);
});
```

Nest

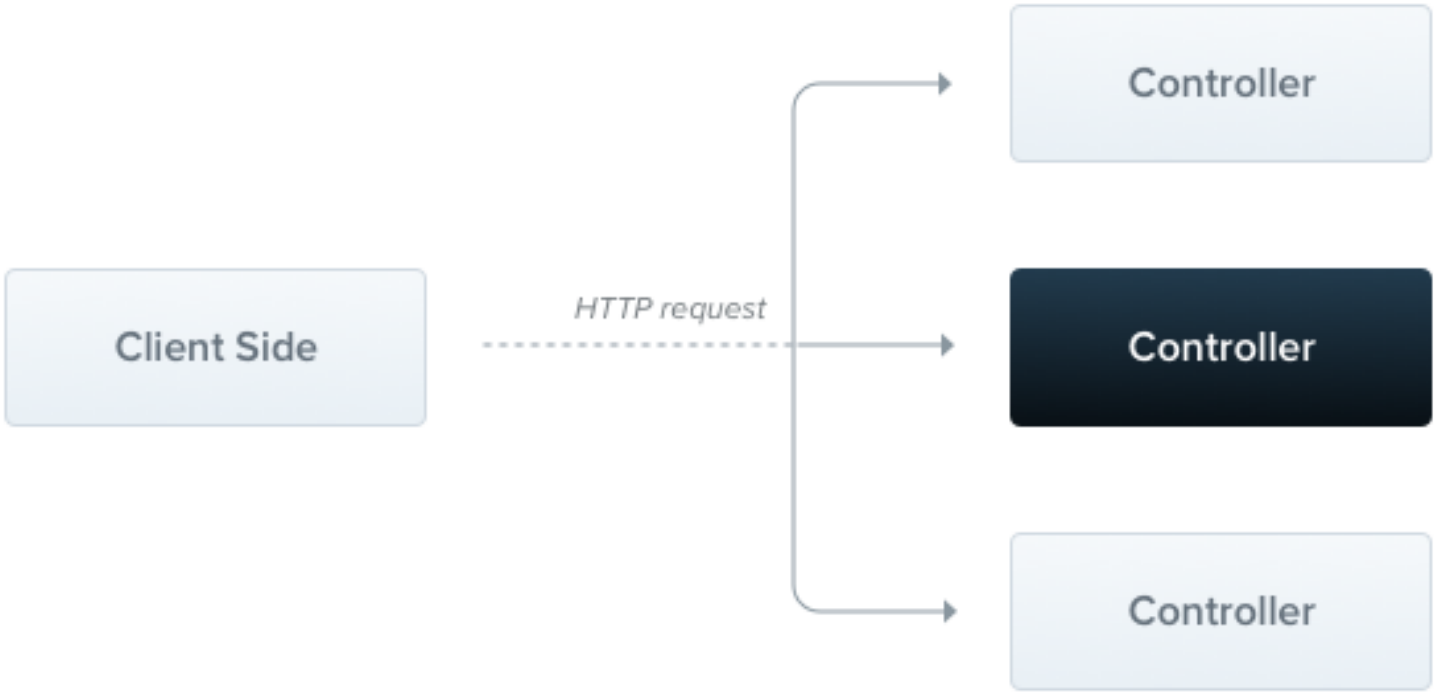
- Фреймворк, предоставляющий из коробки архитектуру для построения тестируемых, масштабируемых слабо-связанных и хорошо поддерживаемых приложений.
- Вдохновлён Angular (и переиспользует его модули)
- Работает как поверх Express, так и поверх Fastify
- DataBase Agnostic
- CLI
- TypeScript
- Rx.js

Nest: основные компоненты

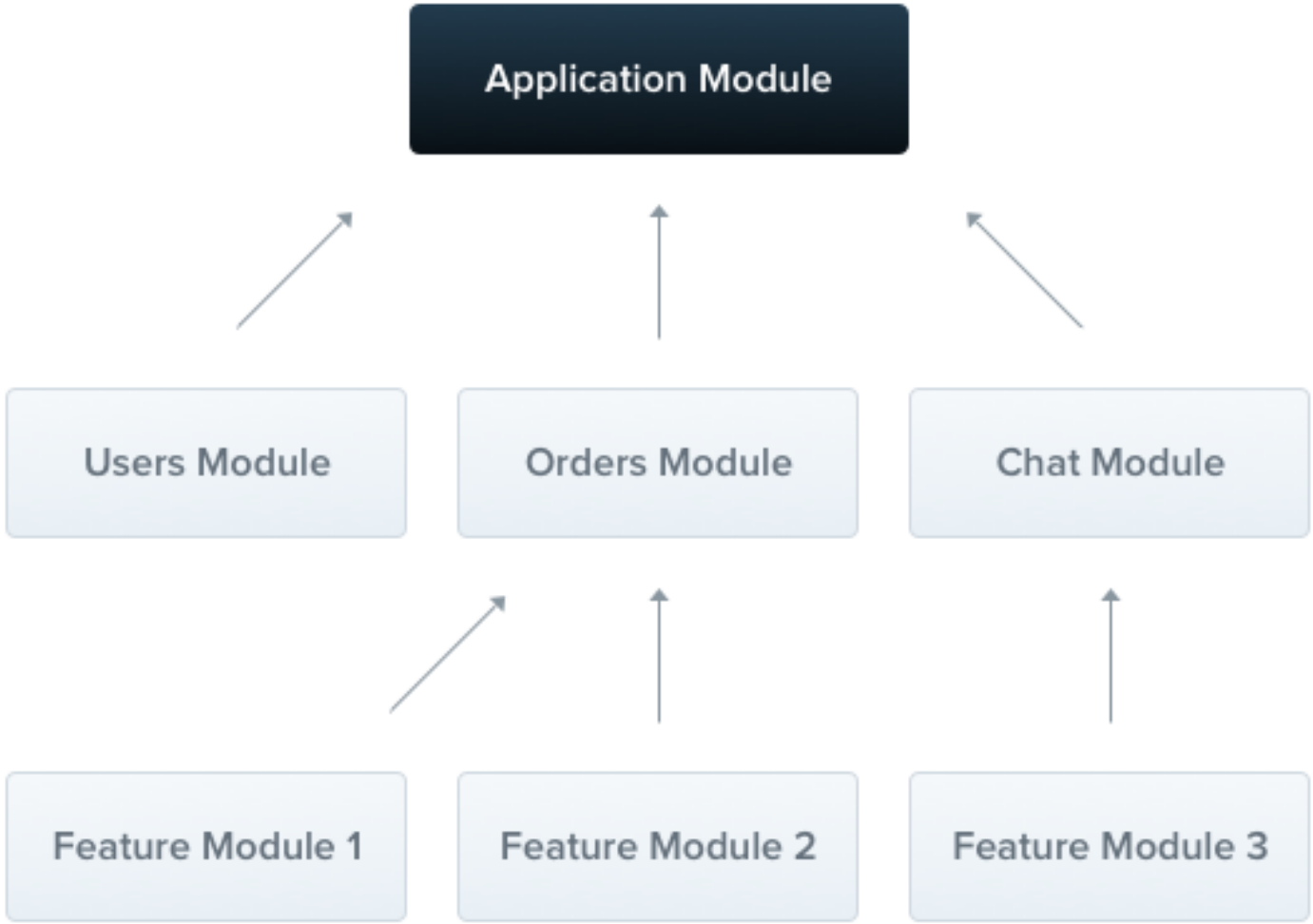
- **Контроллер** - HTTP роутинг
- **Сервис** - компонент, решающий бизнес-задачи
- **Провайдер** - поставщик услуги, внедряемый в другие компоненты приложения (DI)
- **Модуль** - основная единица структуры приложения


Value
Component
Factory
Component
Component
Controller

```ts
import { Injectable } from '@nestjs/common';
import { Cat } from './interfaces/cat.interface';

@Injectable()
export class CatsService {
  private readonly cats: Cat[] = [];

  create(cat: Cat) {
    this.cats.push(cat);
  }

  findAll(): Cat[] {
    return this.cats;
  }
}
```


Client Side
HTTP request
Controller
Controller
Controller

```ts
import { Controller, Get, Post, Body } from '@nestjs/common';
import { CreateCatDto } from './dto/create-cat.dto';
import { CatsService } from './cats.service';
import { Cat } from './interfaces/cat.interface';

@Controller('cats')
export class CatsController {
  constructor(private catsService: CatsService) {}

  @Post()
  async create(@Body() createCatDto: CreateCatDto) {
    this.catsService.create(createCatDto);
  }

  @Get()
  async findAll(): Promise<Cat[]> {
    return this.catsService.findAll();
  }
}
```


Application Module
Users Module
Orders Module
Chat Module
Feature Module 1
Feature Module 2
Feature Module 3

```ts
import { Module } from '@nestjs/common';
import { CatsController } from './cats.controller';
import { CatsService } from './cats.service';

@Module({
  controllers: [CatsController],
  providers: [CatsService],
})
export class CatsModule {}
```

Nest: другие компоненты

- Модули для организации аутентификации и авторизации
- Кэширование, валидация, конфигурация, логгирование
- GraphQL
- WebSockets (обёртки над socket.io и ws)
- Микросервисы: gRPC, Redis, MQTT, NATS, RabbitMQ, Kafka
- Генерация документации приложения
- Генерация Swagger документации API

my-awesome-app documentation

Type to search

- Getting started
  - Overview
  - README
  - Dependencies
- Modules
- Miscellaneous
- Documentation coverage

Documentation generated using

compodoc

A progressive Node.js framework for building efficient and scalable server-side applications, heavily inspired by Angular.

npm v5.5.0 | license MIT | downloads 230k/m | build passing | linux passing | coverage 94%
chat on gitter | backers 84 | sponsors 23 | Donate PayPal | Follow 3k

Description

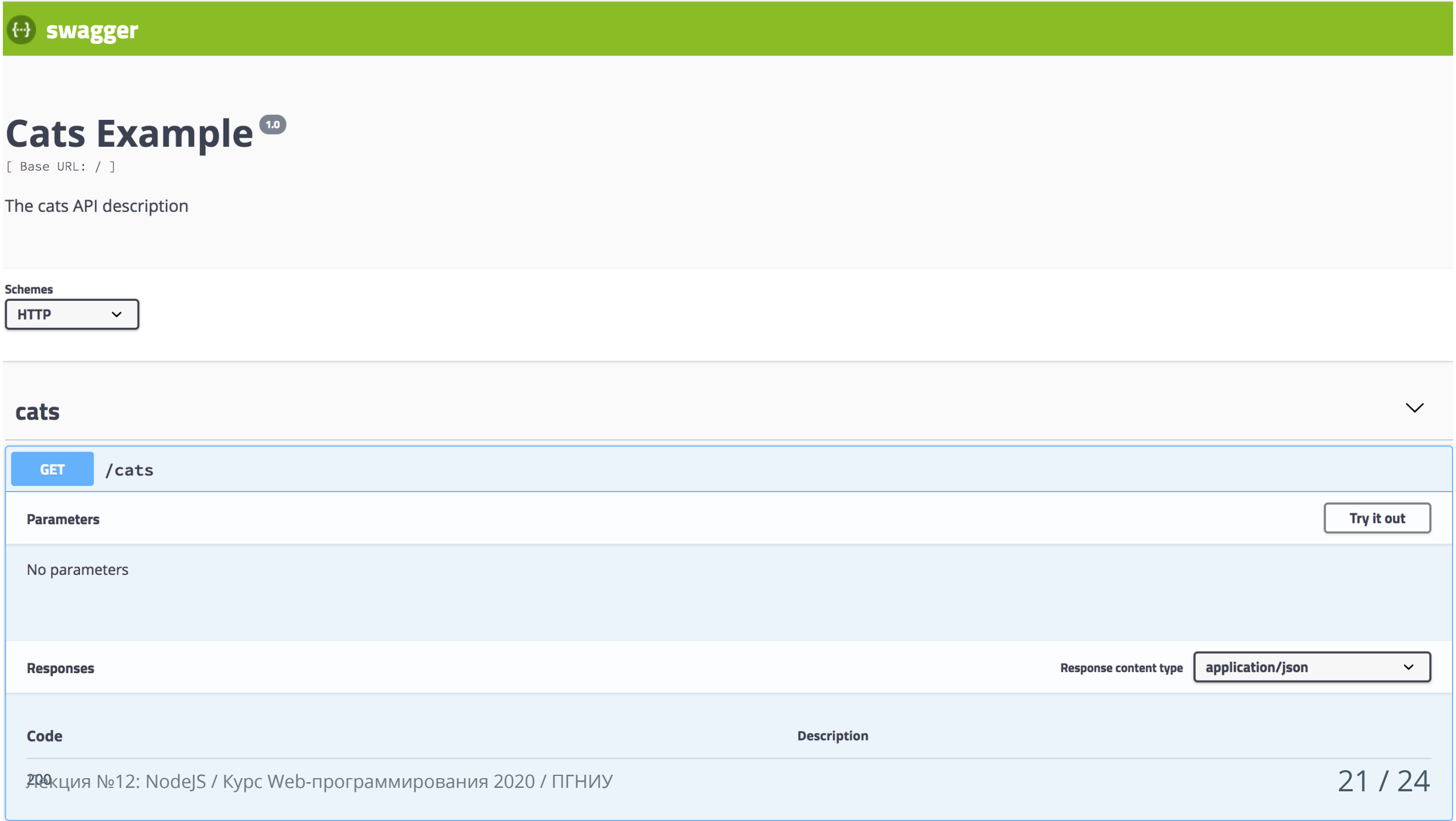
Nest framework TypeScript starter repository.

Installation

```
$ npm install
```

Running the app

```
# development
$ npm run start

# watch mode
$ npm run start:dev

# incremental rebuild (webpack)
$ npm run webpack
$ npm run start:hmr

# production mode
$ npm run start:prod
```


swagger
Cats Example
1.0
[Base URL: /]
The cats API description
Schemes
HTTP
cats
GET
/cats
Parameters
Try it out
No parameters
Responses
Response content type
application/json
Code
Description
200

ORM

- Sequelize
- TypeORM
- Objection
- Mikro-ORM

Headless CMS

- Strapi
- Keystone

Ссылки

- NodeJS: https://nodejs.org/en/
- The Back-end for Front-end Pattern (BFF)
- Express: http://expressjs.com
- NestJS: https://nestjs.com
- HolyJS 2018 Moscow | Kamil Myśliwiec — Revealing framework fundamentals: NestJS behind the curtain
- Sequelize: https://sequelize.org
- Objection: https://vincit.github.io/objection.js/
- Mikro-ORM: http://mikro-orm.io