REVEALING FRAMEWORK FUNDAMENTALS: NESTJS BEHIND THE CURTAIN

KAMMYSLIWIEC

KAMIL MYŚLIWIEC

SOFTWARE ENGINEER | CREATOR OF NESTJS

@KAMMYSLIWIEC

KAMMYSLIWIEC

Kamil Mysliwiec

kamilmysliwiec

Creator of **@nestjs**. Senior Software Engineer at **@scalio**. Master of JavaScript, TypeScript, OpenSource (OSS) enthusiast.

Edit bio

@nestjs
Poland
mail@kamilmysliwiec.com
https://kamilmysliwiec.com

Overview | Repositories 26 | Stars 114 | Followers 572 | Following 3

Pinned repositories

Customize your pinned repositories

nestjs/nest

A progressive Node.js framework for building efficient and scalable server-side applications on top of TypeScript & JavaScript (ES6, ES7, ES8) heavily inspired by Angular 😻 🚀

TypeScript ★ 10.3k 666

nestjs/typescript-starter

Nest framework TypeScript starter ☕

TypeScript ★ 268 94

nestjs/swagger

Swagger module for Nest framework (node.js) 🌎

TypeScript ★ 169 50

nestjs/docs.nestjs.com

The official documentation https://docs.nestjs.com 📕

HTML ★ 82 102

nestjs/graphql

GraphQL (Apollo) module for Nest framework (node.js) 🍷

TypeScript ★ 149 24

nestjs/nest-cli

CLI tool for Nest applications 🍹

TypeScript ★ 297 49

NESTJS

WWW.NESTJS.COM

@NESTFRAMEWORK

★ 10 000

KAMMYSLIWIEC

NEST IS
A PLATFORM

BEST PRACTICES

SUITABLE ABSTRACTIONS


OPINIONATED

EXTREMELY REUSABLE

ANGULAR

KAMMYSLIWIEC

BUILDING BLOCKS

1 MODULES

CONTROLLERS

IMPORTS

PROVIDERS

EXPORTS

CONTROLLERS

IMPORTS

PROVIDERS

EXPORTS

CONTROLLERS

IMPORTS

PROVIDERS

EXPORTS

CONTROLLERS

IMPORTS

PROVIDERS

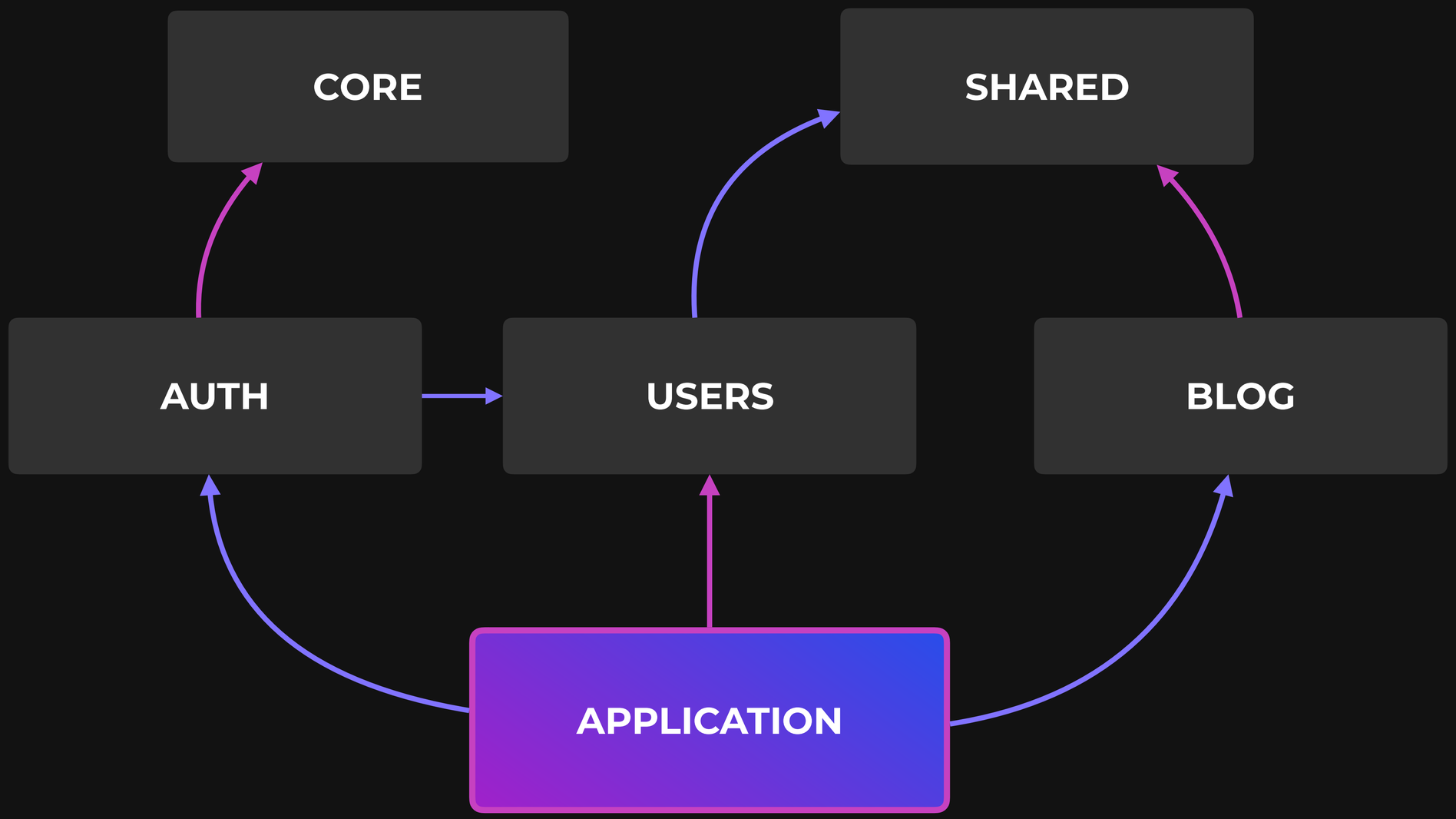
EXPORTS

```
@Module({
  imports: [CommonModule],
  controllers: [AppController],
  providers: [AppService],
  exports: [],
})
class ApplicationModule {}
```

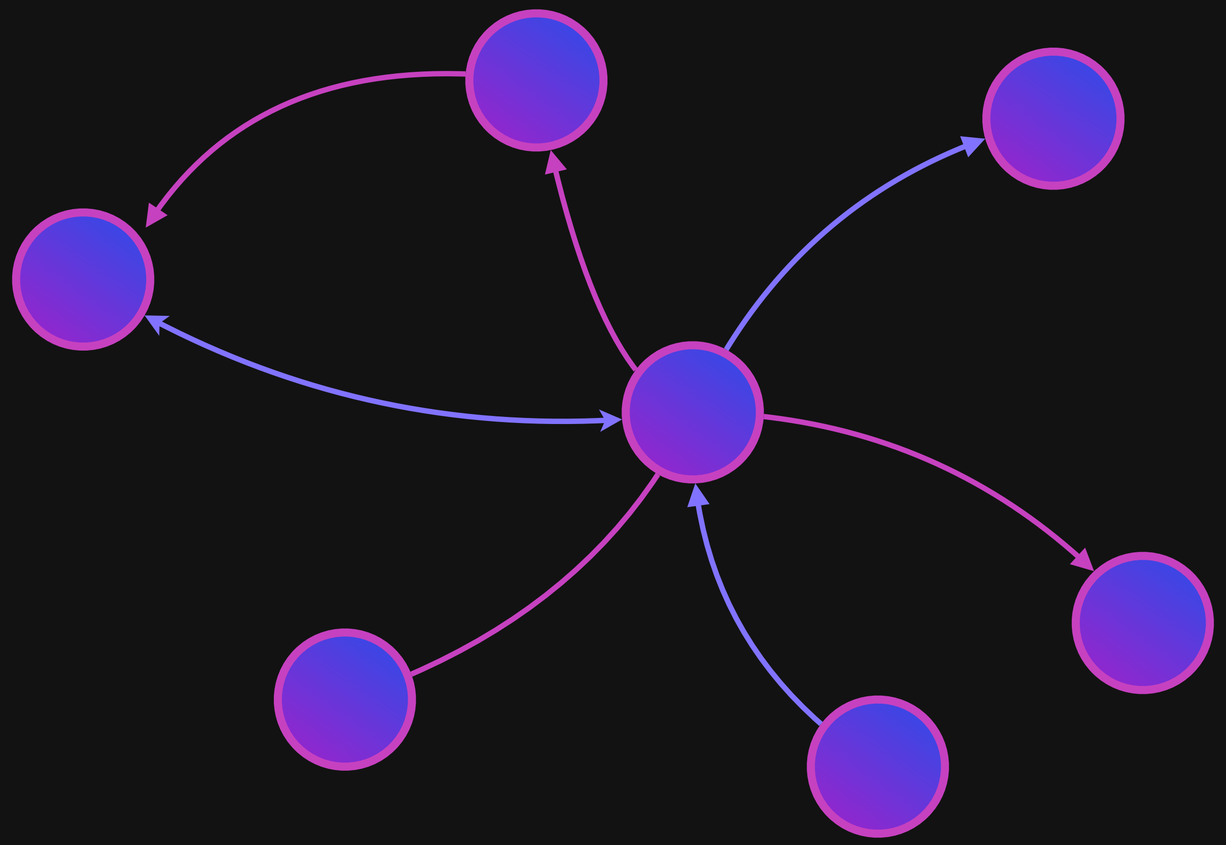
```
@Module({
  imports: [CommonModule],
  controllers: [AppController],
  providers: [AppService],
  exports: [],
})
class ApplicationModule {}
```

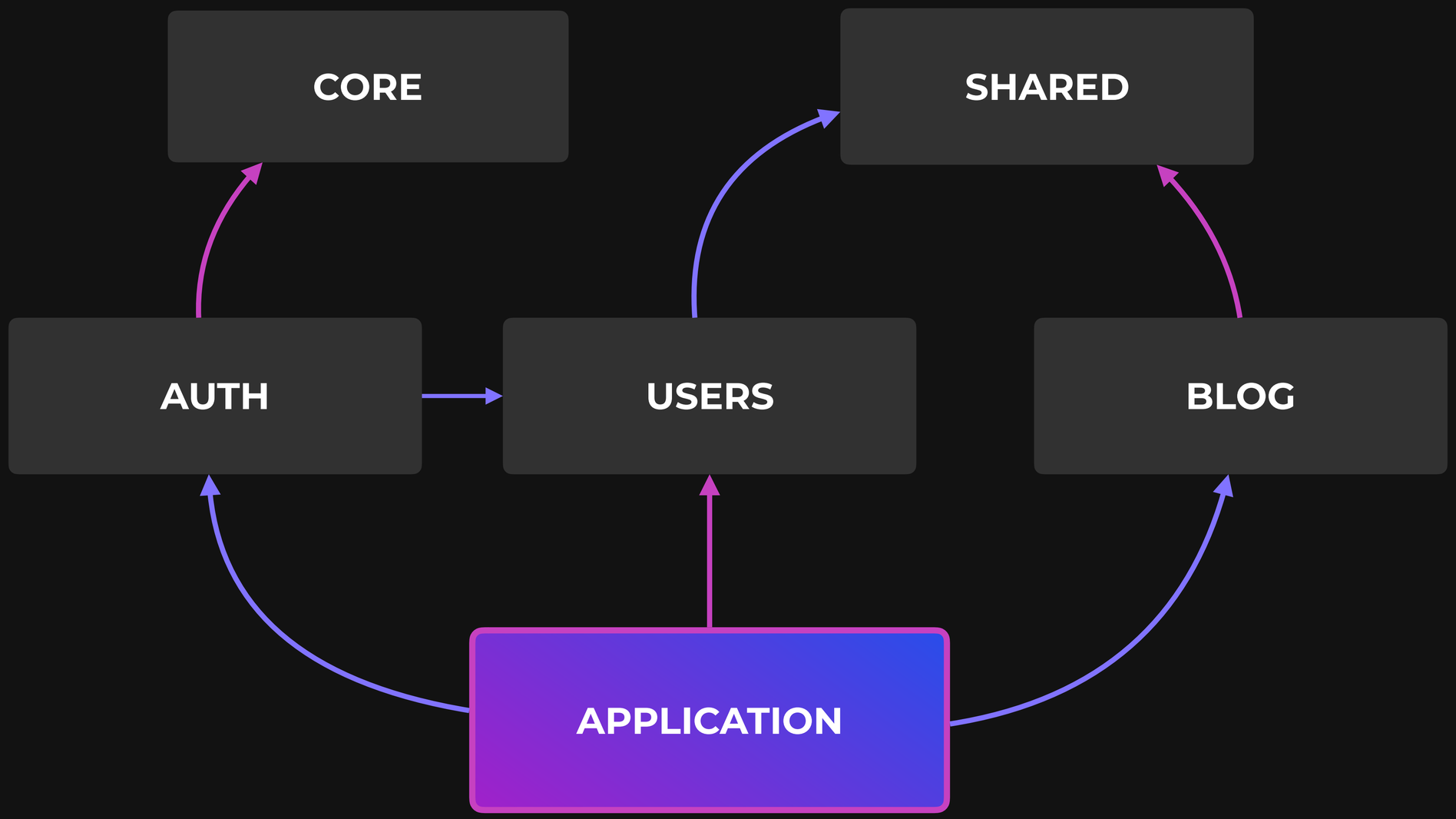
```
@Module({
  imports: [CommonModule],
  controllers: [AppController],
  providers: [AppService],
  exports: [],
})
class ApplicationModule {}
```

```
@Module({
  imports: [CommonModule],
  controllers: [AppController],
  providers: [AppService],
  exports: [],
})
class ApplicationModule {}
```

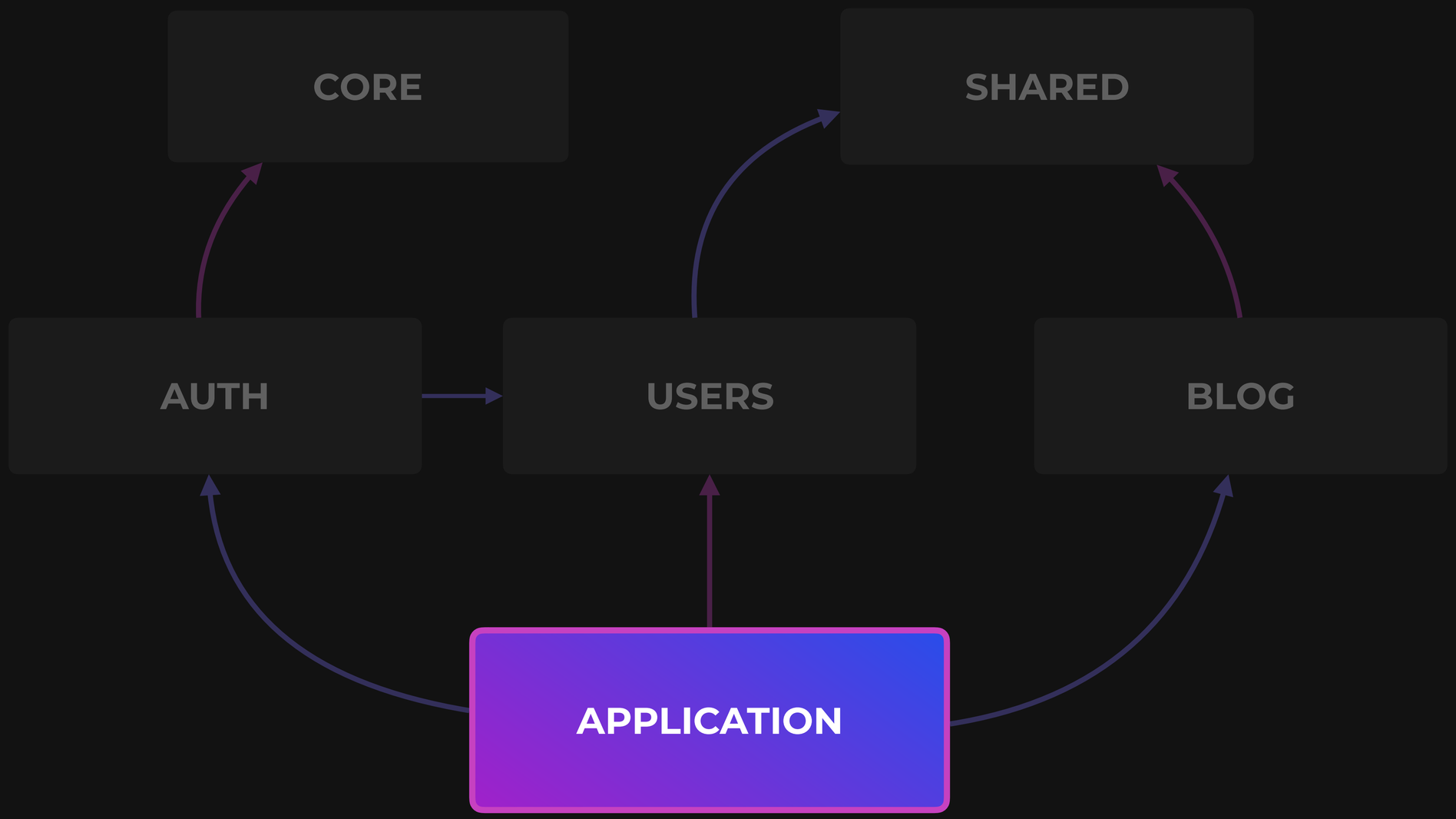


KAMMYSLIWIEC



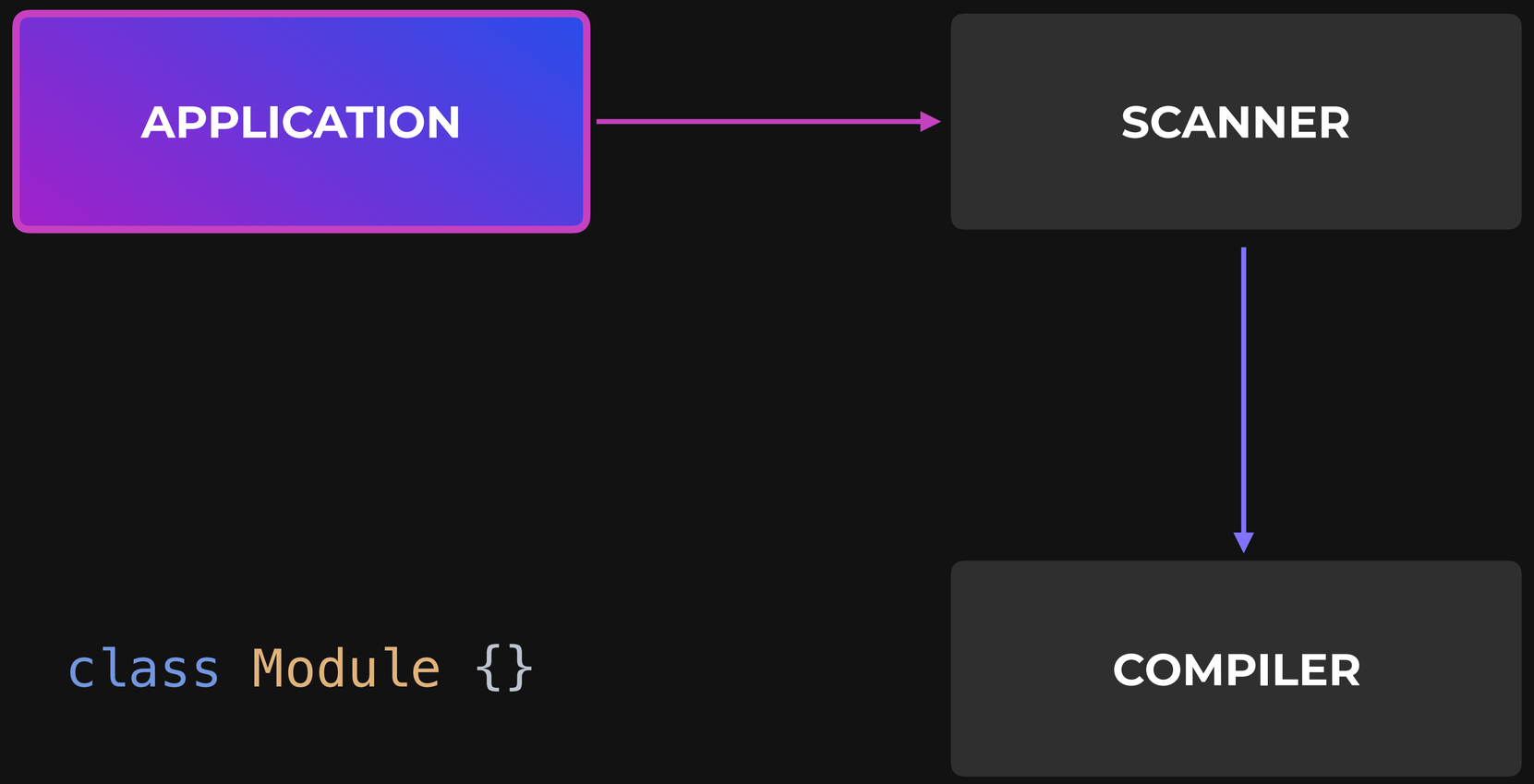


KAMMYSLIWIEC



KAMMYSLIWIEC

APPLICATION




```
class Module {}
```

2 PROVIDERS

```
@Injectable()
class CatsService {
  constructor(
    private httpService: HttpService,
  ) {}
}
```

```
@Injectable()
class CatsService {
  constructor(
    private httpService: HttpService,
  ) {}
}
```

```
@Injectable()
class CatsService {
  constructor(
    private httpService: HttpService,
  ) {}
}
```

```
@Injectable()
class CatsService {
  constructor(
    private httpService: HttpService,
  ) {}
}
```

```
@Injectable()
class CatsService {
  constructor(
    private httpService: HttpService,
    private catsRepository: CatsRepository,
  ) {}
}
```

```
@Injectable()
class CatsService {
  constructor(
    private httpService: HttpService,
    private catsRepository: CatsRepository,
  ) {}
}
```

```
Reflect.getMetadata('design:paramtypes', CatsService);
```

```
Reflect.getMetadata('design:paramtypes', CatsService);
```

```
Reflect.getMetadata('design:paramtypes', CatsService);
```

```
const metadata = Reflect.getMetadata(
  ‘design:paramtypes',
  CatsService,
);
// metadata = [HttpService, CatsRepository]
```

REFLECTION API CHALLENGES

INTERFACES

```
@Injectable()
class CatsService {
  constructor(
    private httpService: HttpService,
  ) {}
}
```

```
@Injectable()
class CatsService {
  constructor(
    private httpService: IHttpService,
  ) {}
}
```

```
const metadata = Reflect.getMetadata(
  ‘design:paramtypes',
  CatsService,
);
// metadata = [Object]
// IHttpService === Object
```

GENERICS

```
@Injectable()
class CatsService {
  constructor(
    private httpService: HttpService,
    private catsRepository: CatsRepository,
  ) {}
}
```

```
@Injectable()
class CatsService {
  constructor(
    private httpService: HttpService,
    private catsRepository: Repository<Cat>,
  ) {}
}
```

```
const metadata = Reflect.getMetadata(
  ‘design:paramtypes',
  CatsService,
);
// metadata = [HttpService, Repository]
// what about Cat? :(
```

CIRCULAR DEPENDENCIES

```
@Injectable()
class CatsService {
  constructor(
    private dogsService: DogsService,
  ) {}
}
```

```
@Injectable()
class DogsService {
  constructor(
    private catsService: CatsService,
  ) {}
}
```

```
const metadata = Reflect.getMetadata(
  ‘design:paramtypes',
  CatsService,
);
// metadata = [undefined]
// DogsService === undefined
```

CUSTOM PROVIDERS

```
{
    provide: 'HTTP_SERVICE',
    useValue: new MockHttpService(),
}
```

```
{
  provide: 'HTTP_SERVICE',
  useValue: new MockHttpService(),
}
```

```
{
    provide: HttpService,
    useValue: new MockHttpService(),
}
```

```
{
  provide: 'HTTP_SERVICE',
  useValue: new MockHttpService(),
}
```

```
{
  provide: 'HTTP_SERVICE',
  useFactory: () => new MockHttpService(),
}
```

```
{
  provide: 'HTTP_SERVICE',
  useClass: MockHttpService,
}
```

```
@Injectable()
class CatsService {
  constructor(
    @Inject('HTTP_SERVICE')
    private httpService: HttpService,
  ) {}
}
```

```
@Injectable()
class CatsService {
  constructor(
    @Optional()
    @Inject('HTTP_SERVICE')
    private httpService: HttpService,
  ) {}
}
```

3 CONTROLLERS


CLIENT
CONTROLLER

GENERAL APPROACH

```
const app = express();
const recipesController = express.Router();

recipesController.post(‘/', (req, res) => {
  const createRecipeDto = req.body;
  res.send('This action adds a new recipe');
});
app.use(recipesController);
```

```
const app = express();
const recipesController = express.Router();

recipesController.post(‘/‘, (req, res) => {
  const createRecipeDto = req.body;
  res.send('This action adds a new recipe');
});
app.use(recipesController);
```

```
const app = express();
const recipesController = express.Router();

recipesController.post('/', (req, res) => {
  const createRecipeDto = req.body;
  res.send('This action adds a new recipe');
});
app.use(recipesController);
```

```
const app = express();
const recipesController = express.Router();

recipesController.post(‘/', (req, res) => {
  const createRecipeDto = req.body;
  res.send('This action adds a new recipe');
});
app.use(recipesController);
```

```
const app = express();
const recipesController = express.Router();

recipesController.post(‘/', (req, res) => {
  const createRecipeDto = req.body;
  res.send('This action adds a new recipe');
});
app.use(recipesController);
```

OUR APPROACH

```
@Controller(‘recipes')
class RecipesController {
  @Post()
  async create(
    @Body() createDto: CreateRecipeDto,
  ): Promise<string> {
    return 'This action adds a new recipe';
  }
}
```

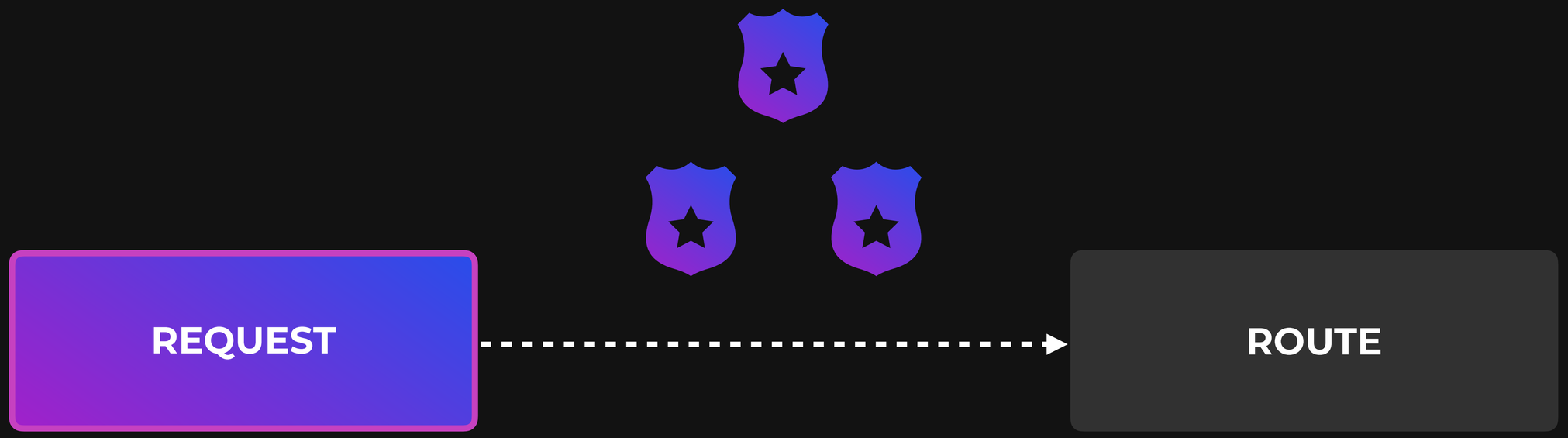
```
@Controller(‘recipes')
class RecipesController {
  @Post()
  async create(
    @Body() createDto: CreateRecipeDto,
  ): Promise<string> {
    return 'This action adds a new recipe';
  }
}
```

```
@Controller(‘recipes')
class RecipesController {
  @Post()
  async create(
    @Body() createDto: CreateRecipeDto,
  ): Promise<string> {
    return 'This action adds a new recipe';
  }
}
```

```
@Controller(‘recipes')
class RecipesController {
  @Post()
  async create(
    @Body() createDto: CreateRecipeDto,
  ): Promise<string> {
    return 'This action adds a new recipe';
  }
}
```

```
@Controller(‘recipes')
class RecipesController {
  @Post()
  async create(
    @Body() createDto: CreateRecipeDto,
  ): Promise<string> {
    return 'This action adds a new recipe';
  }
}
```

4 GUARDS


REQUEST
ROUTE

```
@Injectable()
class AuthGuard implements CanActivate {
  canActivate(...): boolean {
    return true;
  }
}
```

```
@Injectable()
class AuthGuard implements CanActivate {
  canActivate(
    context: ExecutionContext,
  ): boolean {
    return true;
  }
}
```

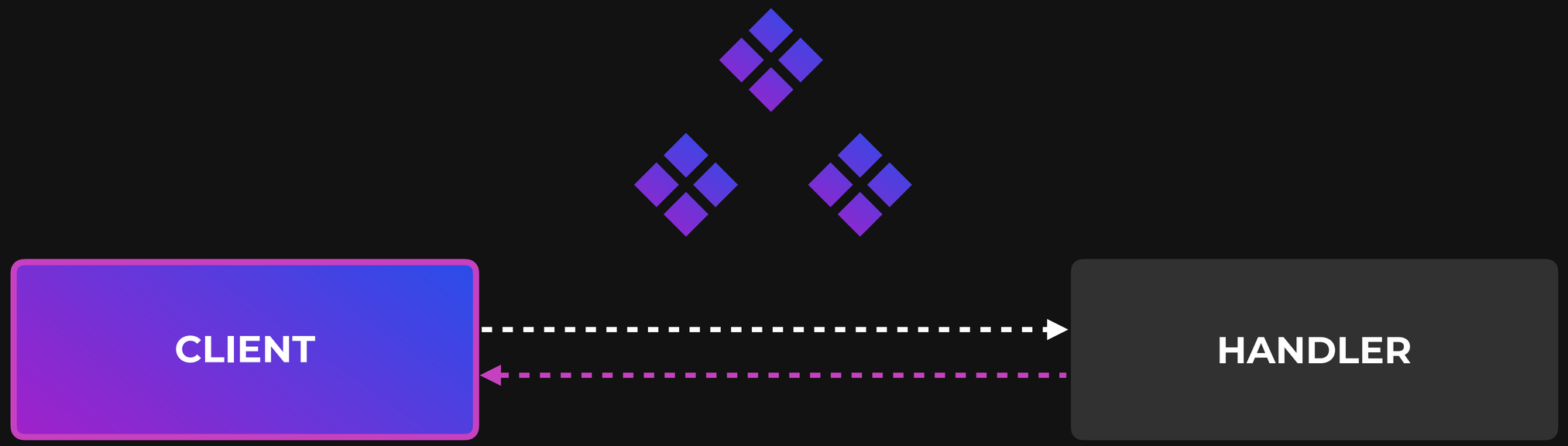
DECLARATIVE BINDING

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

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

```
@Controller(‘cats')
@UseGuards(AuthGuard)
export class CatsController {}
```

5 INTERCEPTORS


CLIENT
HANDLER

```
@Injectable()
class NoopInterceptor<T> implements NestInterceptor {
  intercept(...): Observable<T> {
    return next$;
  }
}
```

```
@Injectable()
export class NoopInterceptor<T> implements NestInterceptor {
  intercept(
    context: ExecutionContext,
    next$: Observable<T>,
  ): Observable<T> {
    return next$;
  }
}
```

MORE

PIPES

```
@Injectable()
class ExponentialStrengthPipe implements PipeTransform {
  transform(
    value: number,
    metadata: ArgumentMetadata,
  ): number {
    ///
  }
}
```

```
@Injectable()
class ExponentialStrengthPipe implements PipeTransform {
  transform(
    value: number,
    metadata: ArgumentMetadata,
  ): number {
    ///
  }
}
```

```
@Injectable()
class ExponentialStrengthPipe implements PipeTransform {
  transform(
    value: number,
    metadata: ArgumentMetadata,
  ): number {
    ///
  }
}
```

```
@Injectable()
class ExponentialStrengthPipe implements PipeTransform {
  transform(
    value: number,
    metadata: ArgumentMetadata,
  ): number {
    ///
  }
}
```

```
@Controller('recipes')
class RecipesController {
  @Post()
  async create(
    @Body() createDto: CreateRecipeDto,
  ): Promise<string> {
    return 'This action adds a new recipe';
  }
}
```

METATYPE

TYPE

DATA

```
@Controller('recipes')
class RecipesController {
  @Post()
  async create(
    @Body() createDto: CreateRecipeDto,
  ): Promise<string> {
    return 'This action adds a new recipe';
  }
}
```

METATYPE

TYPE

DATA

```
const metadata = Reflect.getMetadata(
  ‘design:paramtypes',
  new CatsService(),
  ‘create’
);
// metadata = [CreateRecipeDto]
```

EXCEPTION FILTERS

SWAGGER

swagger

Cats example 1.0

The cats API description

Schemes

HTTP

Authorize

cats

POST /cats

GET /cats/{id}

Models

WEB SOCKETS

```
@SubscribeMessage('events')
handleEvent(client: WsClient, data: string): string {
  return data;
}
```

MICROSERVICES

```
@MessagePattern({ cmd: 'sum' })
sum(data: number[]): Observable<number> {
  return from([1, 2, 3]);
}
```

GRAPHQL

DEVELOPER WORKFLOW

NEST CLI

SCHEMATICS

LATEST NEWS

Strona główna | Powiadomienia | Tweetnij

Eric Simons
@ericsimons40

Obserwuj

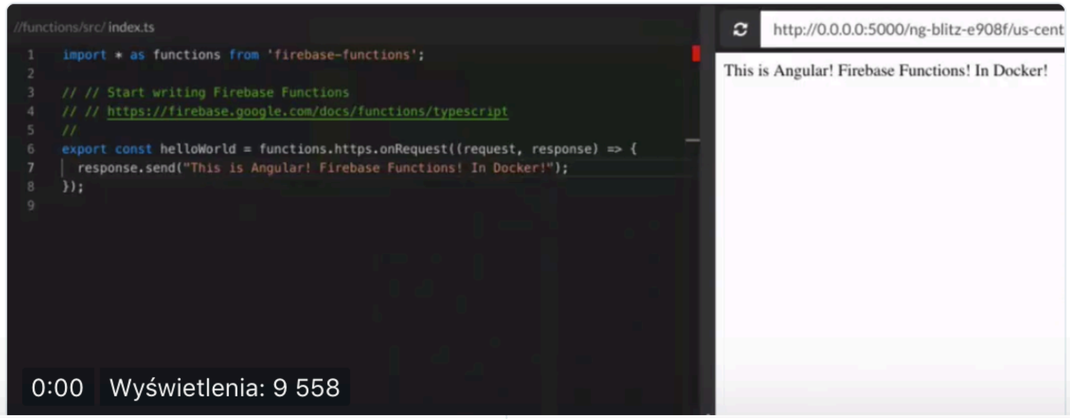
Sneak peek of what's coming to @StackBlitz thanks to @beeman_nl's AMAZING work! 😍

- @nestframework, next.js, @gatsbyjs, rails, django
- @Angular Universal
- @Firebase & GCP Functions

...and anything else that will run in Docker 🤯

Show Bram some ❤️ y'all!!

Przetłumacz tweeta



0:00 Wyświetlenia: 9 558

Kamil Myśliwiec
@kammysliwiec Obserwuje Cię

Creator of @nestframework ❤️ Senior Software Engineer at @scaliolabs. Master of JavaScript, TypeScript, #opensource (OSS) enthusiast.

Polska

Sandbox Editor | Fork | Share | Save

Search sandboxes

Kamil Myśliwiec
kamilmysliwiec

File Editor

- Open Tabs
- Files
  - src
    - app.controller.ts
    - app.module.ts
    - app.service.ts
    - main.ts
  - .gitignore
  - .prettierrc
  - README.md
  - index.html
  - nodemon-debug.json
  - nodemon.json
  - package.json
  - tsconfig.json
  - tsconfig.spec.json
  - tslint.json
- Dependencies

PROD-1541799550-1e6ec17

TS main.ts

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  await app.listen(3000);
}
bootstrap();

```

https://jjo90y00xw.sse.codesandbox.io

Hello world!

Console 0 | Terminal 2

OPEN SOURCE

xtremis
Enjoy Software

SWINGD

neoteric
INNOVATIONS THAT PROFIT

Gojob

ArchitectNow

HostPresto!

scal.io

GENUINEBEE

QUANDER
ENGAGE • CAPTURE • ANALYSE

KAMMYSLIWIEC

THANK YOU

@KAMMYSLIWIEC

KAMMYSLIWIEC