

src\charts\queries\ChartQuery.ts:

```
import { IsISO8601, IsEnum, IsInt, IsOptional, Min } from 'class-validator';
import { Type } from 'class-transformer';
import { Granularity } from '../Granularity';

export class ChartQuery {
  @IsISO8601()
  startDate!: string;

  @IsISO8601()
  endDate!: string;

  @Type(() => Number)
  @IsInt()
  @Min(1)
  page: number = 1;

  @Type(() => Number)
  @IsInt()
  @Min(1)
  pageSize: number = 100;

  // If provided, it must be one of the three values
  @IsOptional()
  @IsEnum(Granularity)
  granularity?: Granularity;
}
```

src\charts\queries\Granularity.ts

```
export enum Granularity {
  Daily = 'daily',
  Monthly = 'monthly',
  Yearly = 'yearly',
}
```

src\charts\charts.controller.spec.ts:

```
import { Test, TestingModule } from '@nestjs/testing';
import { ChartsController } from '../charts.controller';

describe('ChartsController', () => {
  let controller: ChartsController;

  beforeEach(async () => {
    const module: TestingModule = await Test.createTestingModule({
      controllers: [ChartsController],
    }).compile();

    controller = module.get<ChartsController>(ChartsController);
  });
});
```

```

});

it('should be defined', () => {
  expect(controller).toBeDefined();
});
});

```

src\charts\charts.controller.ts:

```

import { Controller, Get, Query } from '@nestjs/common';
import { ChartsService } from '../charts.service';
import { ChartQuery } from '../queries/ChartQuery';

@Controller('charts')
export class ChartsController {
  constructor(private readonly chartsService: ChartsService) {}

  @Get('median-duration')
  medianDuration(@Query() query: ChartQuery) {
    return this.chartsService.medianDuration(query);
  }

  @Get('handling-overview')
  handlingOverview(@Query() query: ChartQuery) {
    return this.chartsService.handlingOverview(query);
  }

  @Get('ai-operation-breakdown')
  aiOperationBreakdown(@Query() query: ChartQuery) {
    return this.chartsService.aiOperationBreakdown(query);
  }

  @Get('call-volume')
  callVolume(@Query() query: ChartQuery) {
    return this.chartsService.callVolume(query);
  }
}

```

src\charts\charts.module.ts

```

import { Module } from '@nestjs/common';
import { ChartsService } from '../charts.service';
import { ChartsController } from '../charts.controller';

@Module({
  providers: [ChartsService],
  controllers: [ChartsController]
})
export class ChartsModule {}

```

src\charts\charts.service.spec.ts

```
import { Test, TestingModule } from '@nestjs/testing';
import { ChartsService } from './charts.service';

describe('ChartsService', () => {
  let service: ChartsService;

  beforeEach(async () => {
    const module: TestingModule = await Test.createTestingModule({
      providers: [ChartsService],
    }).compile();

    service = module.get<ChartsService>(ChartsService);
  });

  it('should be defined', () => {
    expect(service).toBeDefined();
  });
});
```

src\charts\charts.service.ts

```
import { Injectable } from '@nestjs/common';
import { ChartQuery } from './queries/ChartQuery';

@Injectable()
export class ChartsService {
  private readonly backendBaseUrl = 'http://localhost:5000';

  private toQueryRecord(query: ChartQuery): Record<string, string> {
    const record: Record<string, string> = {
      startDate: query.startDate,
      endDate: query.endDate,
      page: String(query.page),
      pageSize: String(query.pageSize),
    };
    if (query.granularity) {
      record.granularity = String(query.granularity);
    }
    return record;
  }

  private async forwardToBackend(
    path: string,
    chartQuery: ChartQuery,
  ): Promise<any> {
    const url = new URL(path, this.backendBaseUrl);
    const params = this.toQueryRecord(chartQuery);
```

```

    for (const [key, value] of Object.entries(params)) {
      url.searchParams.set(key, value);
    }

    const response = await fetch(url.toString());
    if (!response.ok) {
      const text = await response.text().catch(() => "");
      throw new Error(`Backend returned ${response.status}: ${text}`);
    }
    return response.json();
  }

  medianDuration(query: ChartQuery) {
    return this.forwardToBackend(
      '/api/call-records/aggregated/median-duration',
      query,
    );
  }

  handlingOverview(query: ChartQuery) {
    return this.forwardToBackend(
      '/api/call-records/aggregated/handling-overview',
      query,
    );
  }

  aiOperationBreakdown(query: ChartQuery) {
    return this.forwardToBackend(
      '/api/call-records/aggregated/ai-operation-breakdown',
      query,
    );
  }

  callVolume(query: ChartQuery) {
    return this.forwardToBackend(
      '/api/call-records/aggregated/call-volume',
      query,
    );
  }
}

```

src\app.controller.spec.ts

```

import { Test, TestingModule } from '@nestjs/testing';
import { AppController } from './app.controller';
import { AppService } from './app.service';

describe('AppController', () => {

```

```

let appController: AppController;

beforeEach(async () => {
  const app: TestingModule = await Test.createTestingModule({
    controllers: [AppController],
    providers: [AppService],
  }).compile();

  appController = app.get<AppController>(AppController);
});

describe('root', () => {
  it('should return "Hello World!"', () => {
    expect(appController.getHello()).toBe('Hello World!');
  });
});

```

src\app.controller.ts

```

import { Controller, Get } from '@nestjs/common';
import { AppService } from './app.service';

@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  getHello(): string {
    return this.appService.getHello();
  }
}

```

src\app.module.ts

```

import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { ChartsModule } from './charts/charts.module';

@Module({
  imports: [ChartsModule],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}

```

src\app.service.ts

```

import { Injectable } from '@nestjs/common';

```

```

@Injectable()
export class AppService {
  getHello(): string {
    return 'Hello World!';
  }
}

```

src/main.ts

```

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

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

  // Turn on validation
  app.useGlobalPipes(
    new ValidationPipe({
      transform: true,
      whitelist: true,
      forbidNonWhitelisted: false,
    }),
  );

  await app.listen(3000);
}
bootstrap();

```

package.json

```

{
  "name": "charts-proxy",
  "version": "0.0.1",
  "description": "",
  "author": "",
  "private": true,
  "license": "UNLICENSED",
  "scripts": {
    "build": "nest build",
    "format": "prettier --write \"src/**/*.ts\" \"test/**/*.ts\"",
    "start": "nest start",
    "start:dev": "nest start --watch",
    "start:debug": "nest start --debug --watch",
    "start:prod": "node dist/main",
    "lint": "eslint \"{src,apps,libs,test}/**/*.ts\" --fix",
    "test": "jest",
    "test:watch": "jest --watch",
    "test:cov": "jest --coverage",

```

```

    "test:debug": "node --inspect-brk -r tsconfig-paths/register -r ts-node/register
node_modules/.bin/jest --runInBand",
    "test:e2e": "jest --config ./test/jest-e2e.json"
  },
  "dependencies": {
    "@nestjs/common": "^11.0.1",
    "@nestjs/core": "^11.0.1",
    "@nestjs/platform-express": "^11.0.1",
    "class-transformer": "^0.5.1",
    "class-validator": "^0.14.2",
    "reflect-metadata": "^0.2.2",
    "rxjs": "^7.8.1"
  },
  "devDependencies": {
    "@eslint/eslintrc": "^3.2.0",
    "@eslint/js": "^9.18.0",
    "@nestjs/cli": "^11.0.0",
    "@nestjs/schematics": "^11.0.0",
    "@nestjs/testing": "^11.0.1",
    "@types/express": "^5.0.0",
    "@types/jest": "^30.0.0",
    "@types/node": "^22.10.7",
    "@types/supertest": "^6.0.2",
    "eslint": "^9.18.0",
    "eslint-config-prettier": "^10.0.1",
    "eslint-plugin-prettier": "^5.2.2",
    "globals": "^16.0.0",
    "jest": "^30.0.0",
    "prettier": "^3.4.2",
    "source-map-support": "^0.5.21",
    "supertest": "^7.0.0",
    "ts-jest": "^29.2.5",
    "ts-loader": "^9.5.2",
    "ts-node": "^10.9.2",
    "tsconfig-paths": "^4.2.0",
    "typescript": "^5.7.3",
    "typescript-eslint": "^8.20.0"
  },
  "jest": {
    "moduleFileExtensions": [
      "js",
      "json",
      "ts"
    ],
    "rootDir": "src",
    "testRegex": ".*\\.spec\\.ts$",
    "transform": {
      "^.+\\.?(t|j)s$": "ts-jest"
    }
  }
}

```

```

    },
    "collectCoverageFrom": [
      "**/*.(t|j)s"
    ],
    "coverageDirectory": "../coverage",
    "testEnvironment": "node"
  }
}

```

GRAPHQL MIGRATION — CURRENT STRUCTURE (from graphql-migration.pdf, v0)

Project name (package.json): charts-proxy

Runtime: NestJS v11 (HTTP proxy layer in front of an analytics backend)

Purpose (in this stage): Expose HTTP endpoints for charts, validate query params, and forward requests to the legacy analytics backend. Global validation is enabled.

1) FILE TREE (reconstructed)

```

src/
  app.controller.ts
  app.controller.spec.ts
  app.module.ts
  app.service.ts
  charts/
    charts.controller.ts
    charts.controller.spec.ts
    charts.module.ts
    charts.service.ts
    charts.service.spec.ts
  queries/
    ChartQuery.ts
    Granularity.ts
  main.ts
package.json

```

Refs:

- DTO and enum: ChartQuery.ts, Granularity.ts :contentReference[oaicite:0]{index=0}
- Controller endpoints: charts.controller.ts :contentReference[oaicite:1]{index=1}
- Module wiring: charts.module.ts :contentReference[oaicite:2]{index=2}
- Service forwarding logic: charts.service.ts :contentReference[oaicite:3]{index=3}
- Global ValidationPipe and bootstrap: main.ts :contentReference[oaicite:4]{index=4}
- Package scripts and deps: package.json :contentReference[oaicite:5]{index=5}

2) CORE COMPONENTS (roles)

A) Query DTOs (validation at the edge)

- src/charts/queries/ChartQuery.ts
 - startDate, endDate: ISO 8601 strings (required).
 - page: integer ≥ 1 , default 1.
 - pageSize: integer ≥ 1 , default 100.
 - granularity?: enum { daily | monthly | yearly }.
 - Uses class-validator and class-transformer (number coercion).

:contentReference[oaicite:6]{index=6}

- src/charts/queries/Granularity.ts

- export enum Granularity { Daily, Monthly, Yearly }.

:contentReference[oaicite:7]{index=7}

B) HTTP Controller (public endpoints)

- src/charts/charts.controller.ts
 - Base path: /charts
 - Routes:
 - GET /charts/median-duration
 - GET /charts/handling-overview
 - GET /charts/ai-operation-breakdown
 - GET /charts/call-volume
 - Binds @Query() to ChartQuery DTO for validation.

:contentReference[oaicite:8]{index=8}

C) Module wiring

- src/charts/charts.module.ts
 - Registers ChartsService and ChartsController.

:contentReference[oaicite:9]{index=9}

D) Service (backend proxy)

- src/charts/charts.service.ts
 - backendBaseUrl: "http://localhost:5000"
 - toQueryRecord(): maps DTO \rightarrow URLSearchParams
 - startDate, endDate, page, pageSize, granularity?
 - forwardToBackend(path, ChartQuery): builds URL with params, fetch(), JSON.
 - Endpoint mappings (path on legacy backend):
 - medianDuration \rightarrow /api/call-records/aggregated/median-duration
 - handlingOverview \rightarrow /api/call-records/aggregated/handling-overview
 - aiOperationBreakdown \rightarrow /api/call-records/aggregated/ai-operation-breakdown
 - callVolume \rightarrow /api/call-records/aggregated/call-volume

(Note: the word "aggregated" appears with a double "g" in the paths.)

:contentReference[oaicite:10]{index=10}

E) Application bootstrap

- src/main.ts

- Global ValidationPipe({ transform: true, whitelist: true }).
- App listens on port 3000. :contentReference[oaicite:11]{index=11}

F) Project config

- package.json

- Scripts: build, start, start:dev, test (jest), test:e2e, etc.
- Deps: @nestjs/common/core/platform-express, class-validator, class-transformer, rxjs.
- Dev deps: @nestjs/cli, @nestjs/testing, jest, ts-jest, eslint, prettier, typescript, etc.

:contentReference[oaicite:12]{index=12}

3) HTTP → Legacy Backend Mapping (current)

/charts/median-duration → GET
{backendBaseUrl}/api/call-records/aggregated/median-duration
/charts/handling-overview → GET
{backendBaseUrl}/api/call-records/aggregated/handling-overview
/charts/ai-operation-breakdown → GET
{backendBaseUrl}/api/call-records/aggregated/ai-operation-breakdown
/charts/call-volume → GET
{backendBaseUrl}/api/call-records/aggregated/call-volume
(Query params forwarded: startDate, endDate, page, pageSize, granularity?)
:contentReference[oaicite:13]{index=13}

4) VALIDATION MODEL (edge)

- Global ValidationPipe ensures:

- Type transformation (e.g., page → number),
- Whitelisting of declared DTO fields,
- ISO validation on dates via ChartQuery.

:contentReference[oaicite:14]{index=14}

5) TEST STUBS PRESENT

- app.controller.spec.ts, charts.controller.spec.ts, charts.service.spec.ts contain simple “should be defined” tests using @nestjs/testing. :contentReference[oaicite:15]{index=15}

END OF CURRENT STRUCTURE SNAPSHOT

