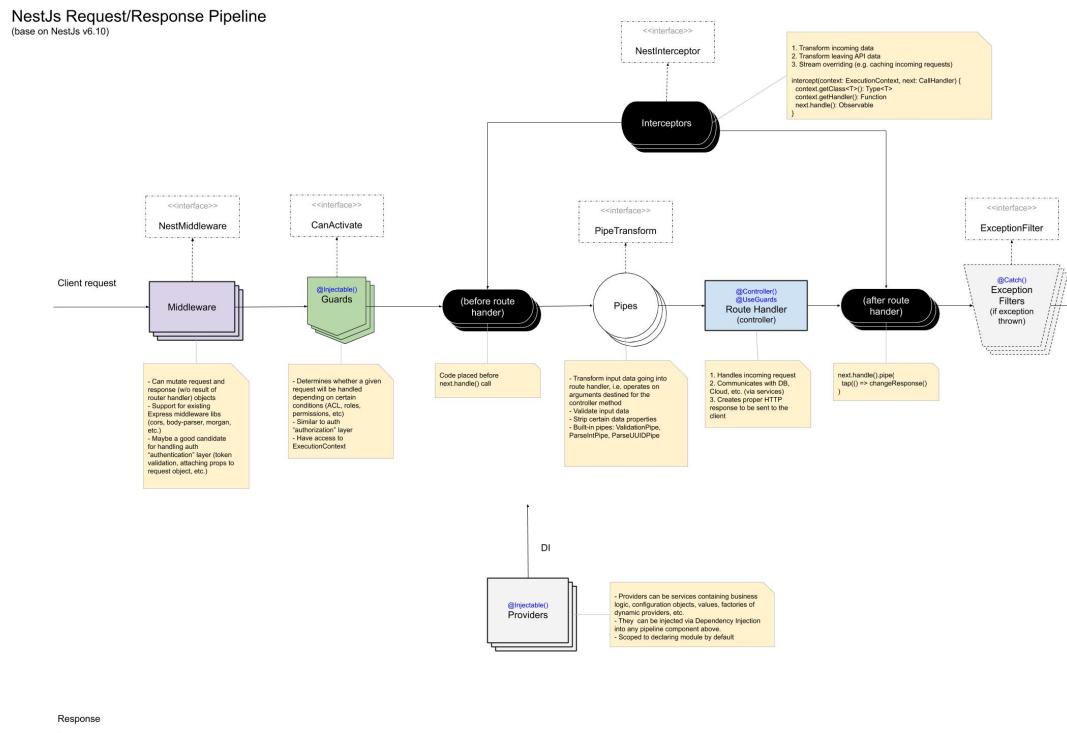




# LN04 - API Loggers

⌚ Created time	@October 22, 2024 2:02 PM
⌘ Type	Documentation
↗ Projects	<a href="#">Advanced Web App</a>
⚡ Status	Proposed
👤 Author	Nguyễn Khang Hy  Minh Huy  callmejippo



## ▼ 1/ Introduction

Sentry là một nền tảng giám sát lỗi và hiệu suất ứng dụng, được thiết kế để nhà phát triển nhanh chóng xác định, phân tích và khắc phục sự cố trong ứng dụng của họ. Nó hoạt động bằng cách thu thập dữ liệu về các lỗi, ngoại lệ và vấn đề hiệu suất từ ứng dụng trong thời gian thực, cung cấp thông tin chi tiết cần thiết để giải quyết vấn đề một cách hiệu quả.

- Sentry cung cấp thông tin đầy đủ về mỗi lỗi, bao gồm stack trace, thông tin về môi trường (trình duyệt, hệ điều hành, thiết bị), phiên bản ứng dụng, breadcrumbs (dòng thời gian các sự kiện dẫn đến lỗi), và nhiều thông tin ngữ cảnh khác.

- Sentry tự động nhóm các lỗi tương tự lại với nhau, giúp bạn tránh bị quá tải bởi hàng loạt thông báo lỗi giống nhau và tập trung vào việc giải quyết nguyên nhân gốc rễ của vấn đề.
- Nhận thông báo ngay lập tức qua email, Slack, Microsoft Teams, hoặc các kênh tích hợp khác khi có lỗi mới xảy ra. Bạn có thể tùy chỉnh các quy tắc thông báo để chỉ nhận thông báo cho các lỗi quan trọng.
- Sentry không chỉ theo dõi lỗi mà còn giám sát hiệu suất ứng dụng, giúp bạn xác định các điểm nghẽn cổ chai và tối ưu hóa tốc độ phản hồi.
- Liên kết lỗi với các bản phát hành cụ thể để dễ dàng theo dõi ảnh hưởng của các thay đổi mã và nhanh chóng xác định phiên bản nào gây ra sự cố.
- Tải lên source maps để Sentry có thể hiển thị mã nguồn gốc của lỗi, giúp bạn dễ dàng gỡ lỗi ngay cả trong môi trường production.
- Sentry cung cấp các dashboards trực quan và báo cáo tùy chỉnh, giúp bạn theo dõi xu hướng lỗi, hiệu suất ứng dụng và các chỉ số quan trọng khác.
- Sentry hỗ trợ hầu hết các ngôn ngữ lập trình phổ biến và framework như JavaScript, Python, Ruby, Java, PHP, Node.js, React, Angular, Vue.js, v.v.

## ▼ 2/ Set up and Configuration

### ▼ 2.1/ Interceptor

Trong NestJS, interceptor là một lớp cung cấp cơ chế để chặn hoặc biến đổi các request và response HTTP. Chúng hoạt động tương tự như middleware, nhưng có thêm khả năng biến đổi dữ liệu trả về từ các handler (controller methods). Interceptor cho phép bạn thực hiện các tác vụ chéo như:

- **Transformation:** Biến đổi dữ liệu response trước khi nó được gửi về client (ví dụ: chuyển đổi dữ liệu sang một định dạng khác).
- **Timing:** Đo thời gian thực thi của các handler.
- **Logging:** Ghi lại thông tin về request và response.
- **Caching:** Lưu trữ kết quả của các request để tăng hiệu suất.
- **Authentication/Authorization:** Xác thực và phân quyền người dùng.
- **Error handling:** Xử lý các lỗi xảy ra trong handler.

Logging interceptor là một loại interceptor cụ thể được sử dụng để ghi lại thông tin về request và response HTTP.

```
import { CallHandler, ExecutionContext, Injectable, Logger, NestIntercep
import { catchError, Observable, tap, throwError } from "rxjs";

@Injectable()
export class LoggingInterceptor implements NestInterceptor {
    private readonly logger = new Logger(LoggingInterceptor.name);

    intercept(context: ExecutionContext, next: CallHandler<any>): Observab
        const request = context.switchToHttp().getRequest();
        const userAgent = request.get('user-agent') || '';
        const { ip, method, path: url } = request;
```

```

    this.logger.log(
      `${method} - ${url} - ${userAgent} - ${ip}: ${context.getClass().name}`
    )

    const now = Date.now();

    return next.handle().pipe(
      tap((res) => {
        const response = context.switchToHttp().getResponse();

        const { statusCode } = response;

        this.logger.log(
          `${method} - ${url} - ${statusCode} - ${userAgent} ${ip} - ${now}`
        );
      }),
      catchError((err) => {
        this.logger.error(err);
        return throwError(() => err);
      }),
    )
  }

}

```

```

async function bootstrap() {

  const app = await NestFactory.create(AppModule);
  app.useGlobalInterceptors(new LoggingInterceptor()); //add this line o

  await app.listen(app.get(ConfigService).get<number>('app.port') ?? 80)
}

```

## ▼ 2.2/ Tạo tài khoản và dự án trên Sentry

- 👉 Truy cập <https://sentry.io/auth/login/> để đăng ký tài khoản
- 👉 Tạo organization

**Create a New Organization**

Organizations represent the top level in your hierarchy. You'll be able to bundle a collection of teams within an organization as well as give organization-wide permissions to users.

Organization Name \*

HCMUS

Data Storage Location \*

Choose where to store your organization's data. Please note, you won't be able to change locations once your organization has been created. [Learn More](#)

eu European Union (EU)

I agree to the [Terms of Service](#) and the [Privacy Policy](#) \*

I agree to let Sentry use my service data for product improvements. [Learn more](#).

Relocating from self-hosted?

**Create Organization**

👉 Chọn platform, tần suất, đặt tên cho project

1 Choose your platform

Popular Browser Server Mobile Desktop Serverless All

Filter Platforms

ANDROID ANGULAR .NET JS DJANGO EXPRESS FASTAPI FLASK GO IOS JAVA LARAVEL

NEST.JS NEXT.JS NODE.JS PHP PYTHON RAILS REACT REACT NATIVE RUBY SPRING BOOT UNITY VUE .NET

2 Set your alert frequency

Alert me on high priority issues

When there are more than 10 occurrences of a unique error in one minute

I'll create my own alerts later

3 Name your project and assign it a team

Project name Team

sakila-project #hcmus Create Project

👉 Hiện ra trang Configure Nest.js SDK, cấu hình theo hướng dẫn!

▼ 2.3/ Cài đặt và cấu hình trên dự án NestJS

### B1. Cài đặt thư viện

```
npm install @sentry/nestjs @sentry/profiling-node --save
```

### B2. Tạo file `instrument.ts` trong thư mục src

```
// Import with `const Sentry = require("@sentry/nestjs");` if you are using ES6 imports
import * as Sentry from '@sentry/nestjs';
```

```

import { nodeProfilingIntegration } from '@sentry/profiling-node';

Sentry.init({
  dsn: 'YOUR_DSN_LINK',
  integrations: [nodeProfilingIntegration()],
  // Tracing
  tracesSampleRate: 1.0, // Capture 100% of the transactions

  // Set sampling rate for profiling - this is relative to tracesSampleRate
  profilesSampleRate: 1.0,
});

```

- **dsn:** Đây là Data Source Name, một chuỗi định danh duy nhất cho dự án Sentry (có được khi tạo project trên sentry)
- **integrations:** Mảng này định nghĩa các tích hợp được sử dụng bởi SDK. nodeProfilingIntegration() cho phép Sentry thu thập dữ liệu profiling, giúp phân tích hiệu suất ứng dụng chi tiết hơn.
- **tracesSampleRate: 1.0:** Đây là tỷ lệ lấy mẫu cho tracing. Giá trị 1.0 nghĩa là 100% các transaction sẽ được ghi lại. Có thể giảm giá trị này (ví dụ: 0.5 cho 50%) để giảm lượng dữ liệu gửi đến Sentry nếu ứng dụng có lưu lượng truy cập rất lớn.
- **profilesSampleRate: 1.0:** Tỷ lệ lấy mẫu cho profiling. Giá trị này là tương đối so với tracesSampleRate. Trong trường hợp này, 100% các transaction được trace cũng sẽ được profile.
- **environment:** Xác định môi trường của ứng dụng (ví dụ: 'production', 'staging', 'development'). Điều này giúp lọc và phân tích lỗi theo môi trường.
- **release:** Xác định phiên bản phát hành của ứng dụng. Giúp bạn theo dõi lỗi theo từng phiên bản.
- **debug:** Bật chế độ debug để xem thông tin chi tiết từ SDK. Chỉ nên sử dụng trong môi trường development.
- **beforeSend:** Một hàm callback cho phép bạn sửa đổi hoặc lọc các sự kiện trước khi chúng được gửi đến Sentry.

### B3. Import `instrument.ts` vào file `main.ts`

```

import './instrument';
import { ConfigService } from '@nestjs/config';
import { NestFactory } from '@nestjs/core';
import { AppModule } from './modules/app/app.module';

async function bootstrap() {

  const app = await NestFactory.create(AppModule);
}

```

```
        await app.listen(app.get(ConfigService).get<number>('app.port') ?? 80)
    }
```

#### B4. Thêm `SentryModule` vào trong `AppModule`

```
import { Module } from '@nestjs/common';
import { SentryModule } from '@sentry/nestjs/setup';
import { AppController } from './app.controller';
import { AppService } from './app.service';

@Module({
  imports: [
    SentryModule.forRoot(),
    // ...other modules
  ],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

#### B5. Upload sourcemaps

```
npx @sentry/wizard@latest -i sourcemaps --saas
```

◇ Where are your build artifacts located?

|

`.\dist`

### ▼ 2.4/ Exception filter

Trong NestJS, Exception Filter là một loại provider chịu trách nhiệm bắt và xử lý các ngoại lệ (exceptions) xảy ra trong ứng dụng. Chúng hoạt động như một lớp trung gian, chặn các ngoại lệ được ném ra bởi các route handler (controller methods) hoặc các provider khác, và cho phép tùy chỉnh cách xử lý các ngoại lệ này trước khi chúng được gửi đến client.

- Thay vì xử lý lỗi trong từng route handler, bạn có thể xử lý chúng một cách tập trung trong một hoặc nhiều Exception Filter. Điều này giúp code của bạn sạch sẽ hơn và dễ bảo trì hơn.
- Exception Filter có thể được sử dụng để ghi lại thông tin về các ngoại lệ, giúp bạn theo dõi và gỡ lỗi ứng dụng.

Khi một ngoại lệ xảy ra trong ứng dụng NestJS, nó sẽ được truyền qua các Exception Filter đã được đăng ký. Nếu một filter "bắt" được ngoại lệ (nghĩa là filter được thiết kế để xử lý loại ngoại lệ đó), nó sẽ xử lý ngoại lệ và trả về một response tùy chỉnh. Nếu không có filter nào bắt được ngoại lệ, NestJS sẽ sử dụng một filter mặc định để trả về một response lỗi chung.



Trong tài liệu của Sentry, đã hướng dẫn chi tiết cách ghi lại toàn bộ exception. Nhóm sẽ demo cách ghi lại những exception có mã lỗi 500.

```

import * as Sentry from '@sentry/node';
import { CaptureContext } from '@sentry/types';

@Catch()
export class AllExceptionsFilter implements ExceptionFilter {
    private readonly logger = new Logger(AllExceptionsFilter.name);

    catch(exception: any, host: ArgumentsHost) {
        const ctx = host.switchToHttp();
        const response = ctx.getResponse<Response>();
        const request = ctx.getRequest<Request>();

        let errorException: ErrorException;
        let httpStatusCode: number;

        if (exception instanceof HttpException) {
            httpStatusCode = exception.getStatus();
            errorException = new ErrorException(
                ErrorCode.HTTP_EXCEPTION,
                exception.message,
                undefined,
            );
        } else if (exception instanceof ErrorException) {
            errorException = exception;
            httpStatusCode = exception.httpStatusCode;
        } else {
            httpStatusCode = HttpStatus.INTERNAL_SERVER_ERROR;
            errorException = new ErrorException(
                ErrorCode.UNDEFINED_ERROR,
                'Internal Server Error',
                exception.message,
            );
        }
    }

    const logData = {
        timestamp: new Date().toISOString(),
        method: request.method,
        path: request.url,
        requestBody: request.body,
        query: request.query,
        params: request.params,
        headers: request.headers,
        responseStatus: httpStatusCode,
        errorCode: errorException.code,
        errorMessage: errorException.message,
    };
}

```

```

        if (httpStatusCode >= 500 && !(exception instanceof HttpException))
            this.logger.error(logData);
            this.logger.error(exception);
        } else if (httpStatusCode >= 400 && httpStatusCode < 500) {
            this.logger.warn(logData);
        }

        const sentryContext: CaptureContext = {
            level: httpStatusCode >= 500 ? 'error' : 'warning',
            extra: logData,
            tags: {
                handled: httpStatusCode < 500 ? 'yes' : 'no',
                url: request.url,
                status_code: errorException.code,
            },
        };
    };

    Sentry.captureException(exception, sentryContext);

    response
        .status(httpStatusCode)
        .setHeader('X-Error-Code', errorException.code)
        .setHeader('X-Error-Message', errorException.message)
        .json(errorException.getErrors());
    }
}

```

```

import { ErrorCode } from './error-code';

export class ErrorException extends Error {
    public readonly code: number;
    public readonly httpStatusCode: number;
    public readonly description: any;

    constructor(code: ErrorCode, message: string | undefined, description: string) {
        super(message);
        const [errorCode, httpStatusCode] = code.split('|');
        this.code = Number(errorCode);
        this.httpStatusCode = Number(httpStatusCode);
        this.description = description;
    }

    public getErrors(): Record<string, any> {
        return {
            error_code: this.code,
            error_message: this.message,
            ...({this.description} && { error_description: this.description }),
        };
    }
}

```

```
    };
}
}
```

```
export enum ErrorCode {
  UNDEFINED_ERROR = '1|500',
  VALIDATION_ERROR = '2|400',
  FORBIDDEN_ERROR = '3|403',
  DATABASE_ERROR = '4|500',
  RESOURCE_NOT_FOUND = '5|404',
  HTTP_EXCEPTION = '6|400',
}
```

```
async function bootstrap() {

  const app = await NestFactory.create(AppModule);

  app.useGlobalFilters(new AllExceptionsFilter());

  await app.listen(app.get(ConfigService).get<number>('app.port') ?? 80)
}
```

### ▼ 3/ Xem Exception Logger trên web

**Issues**

Prioritized 11 For Review 3 Regressed Escalating Archived

All Projects All Envs 14D Custom Search Q is unresolved x issue.priority is high or medium x

Last Seen	GRAPH:	24h 14d	EVENTS	USERS	PRIORITY	ASSIGNEE
Error ActorController.get(actor-controller.ts)	Ongoing	10	0	High		
Error AppController.getError(app-controller.ts)	Ongoing	4	0	High		
NotFoundException callback(@nestjs.core.router:routes-resolver)	Ongoing	8	0	Med		
Error ValidationPipe.exceptionFactory(main.ts)	New	7	0	Med		
TypeError Object.transformFn(actor-dto.ts)	New	7	0	High		
BadRequestException ValidationPipe.exceptionFactory(@nestjs.common.pipes:valid...	Ongoing	4	0	Med		
Error FilmController.createFilmActor(film-controller.ts)	New	1	0	Med		
EntityNotFoundError FilmDatasource.list(film-datasource.ts)	Ongoing	5	0	High		
Error AllExceptionsFilter.catch(all-exceptions-filter.ts)	Ongoing	4	0	High		
TypeError Object.next(logging-interceptor.ts)	Ongoing	4	0	High		
EntityNotFoundError ActorDatasource.list(actor-datasource.ts)	Ongoing	5	0	High		

1-11 of 11 < >

Privacy Policy Terms of Use Service Status API Docs Contribute

## Quản lý lỗi trên hệ thống

Issues > SAKILA-PROJECT-F

Error ActorController.get(actor-controller.ts) Ongoing

Actor not found

Events 9 Users 0 Priority High

Details Activity User Feedback Attachments Tags All Events Merged Issues Similar Issues

Search events...

EVENT ID	TRANSACTION	TITLE	TRACE	TIMESTAMP	RELEASE	ENVIRONMENT	USER	DEVICE	OS
52d11e9	(empty string)	Error: Actor not fou...	7ef2279b	Oct 30, 2024 8:35...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631
a8079ad2	(empty string)	Error: Actor not fou...	e4d60c1b	Oct 30, 2024 8:32...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631
4ad02deb	(empty string)	Error: Actor not fou...	cc29e5d7	Oct 30, 2024 8:32...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631
416d4992	(empty string)	Error: Actor not fou...	b0ff0301	Oct 30, 2024 8:28...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631
3a6eee0e	(empty string)	Error: Actor not fou...	90016245	Oct 30, 2024 8:11...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631
36514da0	(empty string)	Error: Actor not fou...	4bfed61d	Oct 30, 2024 8:09...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631
973f43d7	(empty string)	Error: Actor not fou...	def5dc94	Oct 22, 2024 4:51...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631
680b21a3	(empty string)	Error: Actor not fou...	2c3ed040	Oct 22, 2024 4:47...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631
264e3341	(empty string)	Error: Actor not fou...	c3d2be9c	Oct 22, 2024 4:06...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631
ab068538	(empty string)	Error: Actor not fou...	e52bcd55	Oct 22, 2024 3:09...	(no value)	localhost	(no value)	(empty string)	Windows 10.0.22631

Showing 10 of 10 events < >

Privacy Policy Terms of Use Service Status API Docs Contribute

## Xem danh sách các sự kiện liên quan đến lỗi đó

Issues > SAKILA-PROJECT-F >

**Error ActorController.get(actor-controller.ts)** Ongoing

Actor not found

Event ID: a8079ad2 Oct 30, 8:32 AM

View Full Trace

Event Highlights

handled	yes	transaction	--
level	warning	status_code	101
release	--	Trace: Trace ID	e4d68c1b943349dfb2ed2a1b038979f4
environment	localhost	Runtime: Name	node
url	/api/user/v1/actor/id/1	Runtime: Version	v28.17.0

Stack Trace

Most Relevant Full Stack Trace Newest

Exception - This event

Error: Actor not found

Console

Console

Console

Resources and Possible Solutions

AI SOLUTIONS (EXPERIMENTAL)

You might get lucky, but again, maybe not...

Tags

environment	localhost	runtime	node v28.17.0
handled	yes	name	node
level	warning	server_name	nkhnghyyy
mechanism	generic	status_code	101
os	Windows 10.0.22631	url	/api/user/v1/actor/id/1
name	Windows		

Contexts

App	Operating System
app_memory	Name: Windows
free_memory	Version: 10.0.22631
Start Time	Runtime
2024-10-30T08:32:36.623Z (a few seconds before this event)	Name: node
culture	Version: v28.17.0
locale	
timezone	
Device	Trace Details
Architecture	Span ID: 9e779384871be7b3
Boot Time	Status: unknown
CPU Description	Trace ID: e4d68c1b943349dfb2ed2a1b038979f4
Free Memory	
Memory Size	

Packages

accepts	1.3.8
ansi-styles	4.3.0
app-root-path	3.1.0
append-field	1.0.0

Show More

SDK

Name	sentry.javascript.nestjs
Version	8.34.0

Event Grouping Information

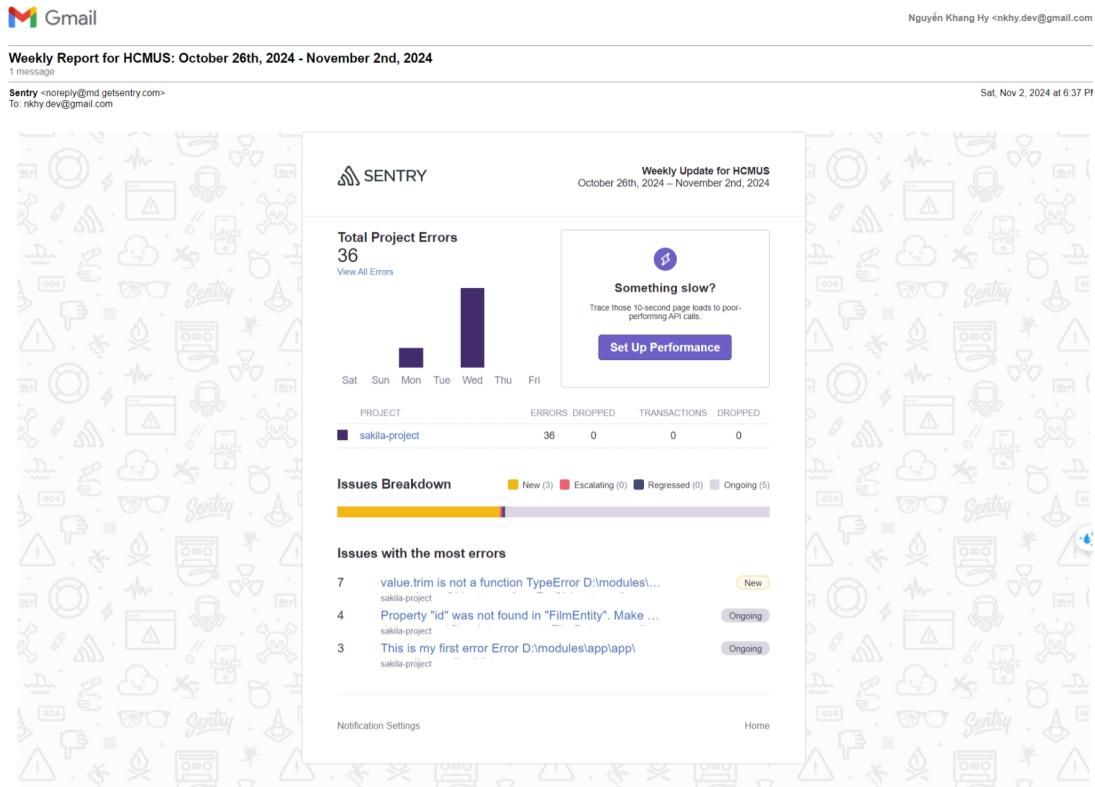
Grouped by: exception stack-trace, in-app exception stack-trace

Show Details

Privacy Policy Terms of Use

Service Status API Docs Contribute

Thông tin chi tiết về lỗi đó



Weekly report về lỗi được gửi về mail của các thành viên trong organization

<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-P - EntityPropertyNotFoundError: Property "created_at" was not found in "FilmEntity". Make sure your query is ... - N	Oct 25
<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-N - Error: Cannot set headers after they are sent to the client - New issue from sakila-project. Sentry Set up ...	Oct 25
<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-M - TypeError: Cannot destructure property 'statusCode' of 'res' as it is undefined. - New issue from sakila...	Oct 25
<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-K - Error: This is my first error - New issue from sakila-project. Sentry Set up in Slack View on Sentry New issue...	Oct 24
<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-J - EntityPropertyNotFoundError: Property "lastUpdate" was not found in "ActorEntity". Make sure your ...	Oct 24
<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-H - BadRequestException: Bad Request Exception - New issue from sakila-project. Sentry Set up in Slack Vie...	Oct 24
<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-G - NotFoundException: Cannot GET /api/user/v1/actors/id/20 - New issue from sakila-project. Sentry Set up ...	Oct 22
<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-F - Error: Actor not found - New issue from sakila-project. Sentry Set up in Slack View on Sentry New issue We ...	Oct 22
<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-E - Error: Actor not found - New issue from sakila-project. Sentry Set up in Slack View on Sentry New issue We ...	Oct 22
<input type="checkbox"/>	<span style="color: #f08080;">★</span>	Sentry	SAKILA-PROJECT-D - Error: Actor not found - New issue from sakila-project. Sentry Set up in Slack View on Sentry New issue We ...	Oct 22

Chi tiết mỗi được thông báo về mail cho các thành viên trong organization

## ▼ 4/ Execute

Với điều kiện đã có database và data của sakila

### B1. Extract source-code.zip

B1.1. Config file `.env` cho phù hợp với điều kiện chạy của máy. Ví dụ

```
APP_NAME=Sakila
APP_PORT=3000
APP_DEBUG=true
APP_FALLBACK_LOCALE=en

DB_CONNECTION=pgsql
DB_TYPE=postgres
DB_HOST=localhost
```

```
DB_PORT=5432
DB_DATABASE=sakila
DB_USERNAME=postgres
DB_PASSWORD=postgres
DB_LOGGING=true

SENTRY_DSN="Your -DSN"
```

- B2. `npm install` để cài đặt các gói trong `package.json`
- B3. `npx @sentry/wizard@latest -i sourcemaps --saas` để upload sourcemaps