ذيفنت مرعف تبتتلا نم ةيادبلا إلى ةيانهلا

.4□□□□□□□□□ بمساعدة المدونة هذه كتابة تمت

طلب كل تتبع إمكانية لضمان النهاية إلى البداية من □□□ □□□□□□ التتبع لمرعف شامل حل على عمل تم لقد الأخطاء، تصحيح في الحل هذا يساعد والخلفية. الأمامية الواجهة عبر متسق بشكل نظامنا في واستجابة عمل لكيفية مفصل شرح يلي فيما فريد. تتبع بمرعف عملية كل ربط خلال من الأحداث وتسجيل والمراقبة، الكود. على أمثلة مع الحل، هذا

الحل يعمل كيف

1. **□□□ □□□□□□ التتبع مرعف إنشاء**:
   - □□□□ عن عبارة المرعف هذا يكون أن يمكن النظام. إلى يصل طلب لكل فريد تتبع مرعف إنشاء يتم أخرى. فريدة قيمة أي أو
   - الطلب. معها يتفاعل التي والخدمات الطبقات جميع عبر المرعف هذا تمرير يتم
2. **الطبقات: عبر التتبع مرعف تمرير**
   - الخادم. إلى إرساله يتم طلب لكل مع التتبع مرعف إرفاق يتم الأمامية، الواجهة في
   - الطلب. تعالج التي والمكونات الخدمات جميع عبر التتبع مرعف تمرير يتم الخلفية، في
3. **الأحداث: تسجيل**
   - الاستجابات، الطلبات، يشمل هذا بها. الخاص التتبع مرعف مع عملية أو حدث كل تسجيل يتم أخرى. عمليات وأي الأخطاء،
   - فيه. البحث يمكن مركزي تسجيل نظام في السجلات هذه تخزين يتم
4. **والمراقبة: الأخطاء تصحيح**
   - أين وفهم النظام عبر الطلب مسار لتتبع التتبع مرعف استخدام يمكن مشكلة، أو خطأ حدوث عند المشكلة. حدثت
   - المستخدم. سلوك وتحليل النظام أداء لمراقبة التتبع مرعف استخدام أيضًا يمكن

الكود على أمثلة

الأمامية الواجهة في التتبع مرعف إنشاء

```
function generateTraceId() {
    return 'trace-' + Math.random().toString(36).substr(2, 9);
}

const traceId = generateTraceId();

fetch('/api/endpoint', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
        'X-Trace-Id': traceId
    },
    body: JSON.stringify({ data: 'example' })
});
```

ريمت فرعم عبتتال يف ةيفلخلا □□□□□.□□ □لاثم

```
const express = require('express');
const { v4: uuidv4 } = require('uuid');

const app = express();

app.use((req, res, next) => {
    const traceId = req.headers['x-trace-id'] || uuidv4();
    req.traceId = traceId;
    res.setHeader('X-Trace-Id', traceId);
    next();
});

app.post('/api/endpoint', (req, res) => {
    const { traceId } = req;
    console.log(`Processing request with Trace ID: ${traceId}`);
    //
    res.json({ status: 'success', traceId });
});
```

```
app.listen(3000, () => {
    console.log('Server is running on port 3000');
});
```

الت تبع مع رف مع الأحداث تسجيل

```
const winston = require('winston');

const logger = winston.createLogger({
    level: 'info',
    format: winston.format.json(),
    transports: [
        new winston.transports.File({ filename: 'combined.log' })
    ]
});

function logEvent(traceId, message) {
    logger.info({ traceId, message });
}

//
logEvent('trace-12345', 'Request received');
```

الخلاصة

عملية جعل مما بأكمله، النظام عبر واستجابة طلب كل تتبع يمكن الفريد، التتبع معرف باستخدام تحليل والتحليل، للتتبع النظام قابلية من يعزز الحل هذا وفعالية. كفاءة أكثر والمراقبة الأخطاء تصحيح عملية في تحسين وتوفير الأداء تجربة لضمان المستخدمين.

كيف يعمل

الأمامية الواجهة [illegible]

إلى العميل مع وإرساله طلب لكل [illegible] تتبع معرف إنشاء الحل هذا من الأمامي الجزء يتضمن الخلفية. على المعالجة من مختلفة مراحل عبر الطلب لتتبع المعرف هذا يُستخدم الخلفية.

1. جمع المعلومات الضرورية من العميل، مثل نوع الشاشة، أبعاد الصورة، ونظام التشغيل. تأكد من جمع جميع المعلومات ذات الصلة قبل إرسال أي طلبات. سيساعد هذا في تقليل عدد الطلبات المطلوبة.
2. إنشاء معرف تتبع فريد لكل طلب. يجب أن يتضمن هذا المعرف معلومات حول العميل والمعلومات المطلوبة. يجب أن يكون هذا المعرف فريدًا لكل طلب ويجب أن يتم إنشاؤه في بداية دورة حياة الطلب.
3. استخدم دالة apiFetch لإجراء المكالمات إلى واجهة برمجة التطبيقات. تأكد من أن كل طلب يتضمن معرف التتبع ومعلومات العميل ذات الصلة المطلوبة.

التعامل مع الأخطاء في واجهة برمجة التطبيقات

يجب أن يتضمن كل طلب يتم إرساله إلى واجهة برمجة التطبيقات معرف التتبع الفريد الخاص به. يجب أن يتضمن كل رد من واجهة برمجة التطبيقات نفس معرف التتبع. يسمح هذا بمطابقة الردود مع الطلبات الأصلية وتسجيل الأخطاء بشكل صحيح.

1. **التعامل مع الأخطاء**: يجب أن تكون واجهة برمجة التطبيقات قادرة على التعامل مع الأخطاء بشكل صحيح. يجب أن ترسل رسائل خطأ واضحة ومفيدة مع معرف التتبع الخاص بالطلب. يجب أن تكون رسائل الخطأ قادرة على مساعدة المطور في تحديد المشكلة وإصلاحها.
2. **التسجيل**: يجب أن تستخدم واجهة برمجة التطبيقات نظام تسجيل قوي يسجل جميع الأخطاء والطلبات. يجب أن يتضمن كل سجل معرف التتبع الخاص بالطلب المقابل. يجب أن يكون نظام التسجيل قادرًا على التعامل مع كميات كبيرة من البيانات وأن يكون قابلاً للبحث.
3. **الاستجابة**: يجب أن ترسل واجهة برمجة التطبيقات استجابة في حالة حدوث خطأ. يجب أن تتضمن الاستجابة معرف التتبع الخاص بالطلب. يجب أن تكون الاستجابة بتنسيق قابل للقراءة بسهولة ويجب أن تتضمن معلومات كافية لمساعدة المطور في تحديد المشكلة وإصلاحها.

البيانات

يجب أن يتم تخزين جميع البيانات ذات الصلة في قاعدة بيانات قوية وقابلة للتطوير. يجب أن تكون قاعدة البيانات قادرة على التعامل مع كميات كبيرة من البيانات وأن تكون قابلة للبحث. يجب أن يكون نظام قاعدة البيانات قادرًا على التعامل مع الاستعلامات المعقدة وأن يكون قادرًا على إرجاع النتائج بسرعة.

يجب أن يتضمن كل سجل في قاعدة البيانات معرف التتبع الخاص بالطلب المقابل. يجب أن يكون معرف التتبع مفهرسًا لتحسين أداء الاستعلامات. يجب أن يكون نظام قاعدة البيانات قادرًا على التعامل مع الاستعلامات التي تبحث عن السجلات بناءً على معرف التتبع الخاص بها.

على سبيل المثال، يمكن استخدام الاستعلام التالي للبحث عن جميع السجلات المتعلقة بمعرف التتبع الخاص بالطلب:

```
trace_id:"Lc6t"
```

بتتبع لك يسمح مما ،□□□6□□ التتبع عرف على تحتوي التي السجل إدخالات جميع سيعيد الاستعلام هذا
نتائج نطاق لتضييق أخرى معايير مع الاستعلام هذا دمج يمكنك ذلك، إلى بالإضافة النظام. عبر الطلب مسار
السجل. رسائل داخل محددة كلمات أو الزمني، الطابع السجل، مستوى حسب التصفية مثل البحث،

بناءً والاتجاهات المقاييس تعرض تحكم لوحات إنشاء أيضًا يمكنك ،□□□□□□ في التصور إمكانيات باستخدام
أوقات متوسط المعالجة، الطلبات عدد تصور يمكنك المثال، سبيل على .□□□□□ □□□□□ على التتبع عرفات
الأنماط تحديد في دعاسي هذه بها. الخاصة التتبع عرفات بمتطلبات وكلها الأخطاء، ومعدلات الاستجابة،
تطبيقك. وموثوقية أداء في المحتملة والمشكلات

سلوك وتحليل وتصحيح لمراقبة شاملاً نهجًا □□ □□□□□ حل مع بالاقتران □□□□□□ استخدام
فيه. والتحقيق فعال بشكل طلب كل تتبع إمكانية يضمن مما نظامك،

الأمامية الواجهة □□□□□□□□□□

api.js

```
const BASE_URL = process.env.REACT_APP_BASE_URL;

//
const getClientInfo = () => {
    const { language, platform, cookieEnabled, doNotTrack, onLine } = navigator;
    const { width, height } = window.screen;
    const connection = navigator.connection || navigator.mozConnection || navigator.webkitConnection;
    const networkType = connection ? connection.effectiveType : 'unknown';
    const timeZone = Intl.DateTimeFormat().resolvedOptions().timeZone;
    const referrer = document.referrer;
    const viewportWidth = window.innerWidth;
    const viewportHeight = window.innerHeight;
};

return {
    screenWidth: width,
    screenHeight: height,
    networkType,
    timeZone,
    language,
    platform,
    cookieEnabled,
```

```
    doNotTrack,
    onLine,
    referrer,
    viewportWidth,
    viewportHeight
};
```

□□

```
//
export const generateTraceId = (length = 4) => {
    const characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
    let traceId = '';
    for (let i = 0; i < length; i++) {
        const randomIndex = Math.floor(Math.random() * characters.length);
        traceId += characters.charAt(randomIndex);
    }
    return traceId;
};

export const apiFetch = async (endpoint, options = {}) => {
    const url = `${BASE_URL}${endpoint}`;
    const clientInfo = getClientInfo();
```

تم ريدصت ةالد apiFetch يتلا ذخأت :نيلماعم endpoint و options ضارتفاب□ أن options يه نئاك غرافف لكشب□. ضارتفاي□ متي اءشنإ ريغتم url نع قيرط جمد BASE_URL عم endpoint. مث متي اءعدتسا ةالد getClientInfo() لوصحلل ىلع تامولعم ليمعلا اهنيزختو في ريغتم clientInfo.

```
const traceId = options.traceId || generateTraceId();

const headers = {
    'Content-Type': 'application/json',
    'X-Client-Info': JSON.stringify(clientInfo),
    'X-Trace-Id': traceId,
    ...(options.headers || {})
};
```

```
const response = await fetch(url, {
    ...options,
    headers
});

    return response;
};
```

App.js

```
try {
  const response = await apiFetch('api', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(content),
    traceId: traceId
  });
```

ومعالجتها: تنسيق إلى البيانات تحويل يتم ،response.ok نجاحة الاستجابة كانت إذا

```
if (response.ok) {
  const data = await response.json();
  ...//
}
```

الخطأ رسالة استخراج ويتم ،تنسيق إلى الخطأ بيانات تحويل يتم نجاحة، غير الاستجابة كانت إذا أما Trace ID إلى التتبع معرف إضافة يتم ثم محددة. رسالة وجود عدم حالة في افتراضية رسالة استخدام أو كإشعار: وعرضها الخطأ رسالة

```
else {
  const errorData = await response.json();
  const errorMessage = errorData.message || '           ';
  let errorToastMessage = errorMessage;
  errorToastMessage += `    ) :    }$traceId})`;
  toast.error(errorToastMessage, {
```

```
    autoClose: 8000
  });
  setError(errorToastMessage);
}
```

سلسلة إلى تحويله يتم ثم ،Error نوع من الخطأ كان إذا مما التحقق يتم الكود، تنفيذ أثناء خطأ حدوث حالة في
:نصية

```
catch (error) {
  let errorString = error instanceof Error ? error.message : JSON.stringify(error);
}
```

[illegible] 1000

```
if (error.response) {
    //                                  2xx
    errorString += ` (HTTP ${error.response.status}: ${error.response.statusText})`;
    console.error('       :     ', error.response.data);
} else if (error.request) {
    //
    errorString += '  )          (    ';
    console.error('       :   ', error.request);
} else {
    //
    errorString += `   )       :    }$error.message})`;
}

errorString += `    ) :    }$traceId})`;

if (error instanceof Error) {
    errorString += `\nStack: ${error.stack}`;
}
```

إذا كان error عبارة عن كائن من نوع Error، يتم إضافة سلسلة نصية تحتوي على \nStack: متبوعة بمحتوى error.stack إلى errorString.

[illegible]

[illegible] :) ${duration} (

```
toast.error(`:   }$errorString}`, {
    autoClose: 8000
  });
  setError(errorString);
} finally {
  toast.dismiss(toastId);
}
```

□□□□□□□□□□ الخلفية

__init__.py

```
# -*- encoding: utf-8 -*-

import os
import json
import time
import uuid
import string
import random

from flask import Flask, request, Response, g, has_request_context
from flask_cors import CORS

from .routes import initialize_routes
from .models import db, insert_default_config
import logging
from logging.handlers import RotatingFileHandler
from prometheus_client import Counter, generate_latest, Gauge
from flask_migrate import Migrate
from logstash_formatter import LogstashFormatterV1
```

تم ترجمة الكود التالي إلى: أعلاه

```
from .routes import initialize_routes
from .models import db, insert_default_config
import logging
```

```
from logging.handlers import RotatingFileHandler
from prometheus_client import Counter, generate_latest, Gauge
from flask_migrate import Migrate
from logstash_formatter import LogstashFormatterV1
```

هي. كما تبقى أن يجب محددة ودوال مكتبات أسماء على يحتوي لأنه ترجمته يتم لم الكود ملاحظة:

```
app = Flask(__name__)

app.config.from_object('api.config.BaseConfig')

db.init_app(app)
initialize_routes(app)

CORS(app)

migrate = Migrate(app, db)

class RequestFormatter(logging.Formatter):
    def format(self, record):
        if has_request_context():
            record.trace_id = getattr(g, 'trace_id', 'unknown')
        else:
            record.trace_id = 'unknown'
        return super().format(record)
```

إلى: أعلاه الكود ترجمة تمت

```
class RequestFormatter(logging.Formatter):
    def format(self, record):
        if has_request_context():
            record.trace_id = getattr(g, 'trace_id', 'unknown')
        else:
            record.trace_id = 'unknown'
        return super().format(record)
```

طريقة تعديل يتم .logging.Formatter من ترث التي RequestFormatter فئة تعريف يتم أعلاه، الكود في من trace_id تعيين يتم متاحًا، كان إذا متاح. request context طلب سياق هناك كان إذا ما لفحص format الكائن g إلى record.trace_id. إذا طلب، سياق هناك يكن لم إذا 'unknown'. القيمة إلى trace_id تعيين يتم التنسيق. لإكمال الأم الفئة من format الأصلية الطريقة استدعاء يتم أخيرًا،

```
class CustomLogstashFormatter(LogstashFormatterV1):
    def format(self, record):
        if has_request_context():
            record.trace_id = getattr(g, 'trace_id', 'unknown')
        else:
            record.trace_id = 'unknown'
        return super().format(record)
```

إلى: أعلاه الكود ترجمة تمت

```
class CustomLogstashFormatter(LogstashFormatterV1):
    def format(self, record):
        if has_request_context():
            record.trace_id = getattr(g, 'trace_id', 'unknown')
        else:
            record.trace_id = 'unknown'
        return super().format(record)
```

تبقى ما عادةً والتي بالإنجليزية، ومتغيرات دوال أسماء على يحتوي لأنه هو كما الكود على الحفاظ تم ملاحظة: الترجمة. في هي كما

```
def setup_loggers():
    logstash_handler = RotatingFileHandler(
        'app.log', maxBytes=100000000, backupCount=1)
    logstash_handler.setLevel(logging.DEBUG)
    logstash_formatter = CustomLogstashFormatter()
    logstash_handler.setFormatter(logstash_formatter)

txt_handler = RotatingFileHandler(
    'plain.log', maxBytes=100000000, backupCount=1)
txt_handler.setLevel(logging.DEBUG)
txt_formatter = RequestFormatter(
    '%(asctime)s %(levelname)s: %(message)s [in %(pathname)s:%(lineno)d] [trace_id: %(trace_id)s]')
txt_handler.setFormatter(txt_formatter)

root_logger = logging.getLogger()
root_logger.setLevel(logging.DEBUG)
```

```
root_logger.addHandler(logstash_handler)
root_logger.addHandler(txt_handler)
```

تم الحفاظ على الكود كما هو لأنه يحتوي على أسماء متغيرات ودوال بالإنجليزية. إلى يحتاج ولا ترجمة.

```
app.logger.addHandler(logstash_handler)
app.logger.addHandler(txt_handler)

werkzeug_logger = logging.getLogger('werkzeug')
werkzeug_logger.setLevel(logging.DEBUG)
werkzeug_logger.addHandler(logstash_handler)
werkzeug_logger.addHandler(txt_handler)
```

تم الحفاظ على الكود كما هو لأنه يحتوي على أسماء متغيرات ومكتبات. يجب لا ترجمتها.

```
setup_loggers()

def generate_trace_id(length=4):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(length))
```

ترجمة:

```
def generate_trace_id(length=4):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(length))
```

ملاحظة: الكود أعلاه كما هو سيبقى بدون ترجمة. يتم ولا [illegible] برمجة بلغة مكتوب لأنه generate_trace_id الوظيفة ترجمته.

تتبع عرف بإنشاء تقوم [illegible] عشوائي استخدام بأحرف أبجدية أحرف وأرقام.

```
@app.before_request
def before_request():
    request.start_time = time.time()
    trace_id = request.headers.get('X-Trace-Id', generate_trace_id())
    g.trace_id = trace_id
```

تمت ترجمة الكود أعلاه إلى:

```
@app.before_request
def before_request():
    request.start_time = time.time()
    trace_id = request.headers.get('X-Trace-Id', generate_trace_id())
    g.trace_id = trace_id
```

هذا في الكود، يتم تعيين وقت بداية الطلب باستخدام time.time()، ثم يتم الحصول على start_time باستخدام headers الم فتاح X-Trace-Id. إذا لم يتم العثور على X-Trace-Id في trace_id من رأس الطلب باستخدام الدالة generate_trace_id(). أخيرًا، يتم تعيين trace_id في الرأس، إنشاء trace_id جديد باستخدام g الذي يمكن الوصول إليه في جميع أنحاء التطبيق.

```
    client_info = request.headers.get('X-Client-Info')
    if client_info:
        try:
            client_info_json = json.loads(client_info)
            logging.info(f"     :     }client_info_json}")
        except json.JSONDecodeError:
            logging.warning(   " JSON          X-Client-Info")

@app.after_request
def after_request(response):
    response.headers['X-Trace-Id'] = g.trace_id

    if response.status_code != 200:
        logging.error(f'      :      }response.status_code}')
        logging.error(f'   :      }response.get_data(as_text=True)}')

    if response.content_type == 'application/json':
        try:
            response_json = response.get_json()
            response_json['trace_id'] = g.trace_id
            response.set_data(json.dumps(response_json))
        except Exception as e:
            logging.error(f"        trace_id    :      {e}")
```

##

:

□□□□□□□□□:□□□6□□

```json
{
  "_index": "flask-logs-2024.07.05",
  "_type": "_doc",
  "_id": "Ae9zgZABqOMSOpxCZC5X",
  "_version": 1,
  "_score": 1,
  "_source": {
    "tags": [
      "_grokparsefailure"
    ],
    "filename": "generate.py",
    "funcName": "post",
    "message":  "          "   ,
    "@version": 1,
    "name": "root",
    "host": "ip-172-31-35-xxx.ec2.internal",
    "relativeCreated": 685817.8744316101,
    "levelname": "INFO",
    "created": 1720158740.894831,
    "thread": 139715118360128,
    "threadName": "Thread-5",
    "levelno": 20,
    "pathname": "/home/project/project-name/api/routes/generate.py",
    "msecs": 894.8309421539307,
    "processName": "MainProcess",
    "lineno": 287,
    "path": "/home/project/project-name/app.log",
    "args": [],
    "source_host": "ip-172-31-35-xxx.ec2.internal",
    "module": "generate",
```

```
  "trace_id": "Lc6t",
  "stack_info": null,
  "process": 107613,
  "@timestamp": "2024-07-05T05:52:20.894Z"
},
"fields": {
  "levelname.keyword": [
    "INFO"
  ],
  "tags.keyword": [
    "_grokparsefailure"
  ],
  "relativeCreated": [
    685817.9
  ],
  "processName.keyword": [
    "MainProcess"
  ],
  "filename.keyword": [
    "generate.py"
  ],
  "funcName": [
    "post"
  ],
  "path": [
    "/home/project/project-name/app.log"
  ],
  "processName": [
    "MainProcess"
  ],
  "@version": [
    1
  ],
  "host": [
    "ip-172-31-35-xxx.ec2.internal"
  ],
```

```
"msecs": [
  894.83093
],
"source_host.keyword": [
  "ip-172-31-35-xxx.ec2.internal"
],
"host.keyword": [
  "ip-172-31-35-xxx.ec2.internal"
],
"levelname": [
  "INFO"
],
"process": [
  107613
],
"threadName.keyword": [
  "Thread-5"
],
"trace_id": [
  "Lc6t"
],
"source_host": [
  "ip-172-31-35-xxx.ec2.internal"
],
"created": [
  1720158700
],
"module": [
  "generate"
],
"module.keyword": [
  "generate"
],
"name.keyword": [
  "root"
],
```

```
"thread": [
  139715118360128
],
"message": [
    "         "
[,
"levelno": [
  20
],
"trace_id.keyword": [
  "Lc6t"
],
"threadName": [
  "Thread-5"
],
"pathname": [
  "/home/project/project-name/api/routes/generate.py"
],
"tags": [
  "_grokparsefailure"
],
"pathname.keyword": [
  "/home/project/project-name/api/routes/generate.py"
],
"@timestamp": [
  "2024-07-05T05:52:20.894Z"
],
"filename": [
  "generate.py"
],
"lineno": [
  287
],
"message.keyword": [
    "         "
[,
```

```
    "name": [
      "root"
    ],
    "funcName.keyword": [
      "post"
    ],
    "path.keyword": [
      "/home/project/project-name/app.log"
    ]
  }
}
```

كما هو موضح أعلاه، يمكنك رؤية معرف التتبع في السجل.