**Documentation Technique pour le Monitorage de l'Application Django**

Métriques Prometheus **:**

* Les middlewares PrometheusBeforeMiddleware et PrometheusAfterMiddleware sont utilisés pour collecter les métriques Prometheus de l’application Django.
* Sentry : Sentry est configuré pour la gestion des erreurs et des alertes.

#### **Liste des Métriques et des Seuils et Valeurs d’Alerte**

1. Temps de Réponse des Requêtes HTTP :
   * Métrique : Temps moyen de réponse des requêtes HTTP collecté par Prometheus.
   * Seuil d’Alerte : Si le temps de réponse dépasse 2000ms, une alerte est déclenchée.
   * Disponible dans /metrics de l’application.
2. Taux d’Erreur des Requêtes HTTP :
   * Métrique : Pourcentage de requêtes HTTP échouées (codes de réponse 4xx et 5xx) collecté par Prometheus.
   * Seuil d’Alerte : Si le taux d’erreur dépasse 5%, une alerte est déclenchée.
   * Diponible dans /metrics de l’application.
3. Erreurs d'Application :
   * Métrique : Nombre d'exceptions non gérées collectées par Sentry.
   * Seuil d’Alerte : Toute exception non gérée déclenche une alerte immédiate.
   * Disponible sur le dashboard de Sentry.

#### Arguments en Faveur des Choix Techniques pour l’Outillage du Monitorage de l’Application

1. Prometheus et Django Prometheus :
   * Choix : Utilisation de Prometheus pour la collecte des métriques et de django-prometheus pour l'intégration avec Django.
   * Poins Forts :
     + Scalabilité : Prometheus est conçu pour être performant à grande échelle.
     + Flexibilité : Permet de définir des métriques personnalisées et de les collecter facilement via des exports.
     + Communauté Active : Bénéficie d'une large communauté et de nombreuses intégrations avec d'autres outils de monitoring.
2. Sentry :
   * Choix : Utilisation de Sentry pour la gestion des erreurs et des alertes.
   * Points forts :
     + Efficacité : Sentry permet de détecter et diagnostiquer les erreurs en temps réel.
     + Intégration : S'intègre facilement avec Django.
     + Analyse des Incidents : Offre des fonctionnalités avancées d’analyse des incidents et des performances.

**Procédure d’Installation et de Configuration des Dépendances pour l’Outillage du Monitorage de l’Application**

Environnement Python **:**

* Python 3,11 et pip nécessaires.

Installer Django et les dépendances du projet **:**

* Installez les dépendances via le fichier requirements.txt

**Installer Prometheus et Django Prometheus :**

Mettre à jour les paramètres Django (settings.py) pour inclure les middlewares et applications Prometheus :

INSTALLED\_APPS = [

# Autres applications...

'django\_prometheus',

]

MIDDLEWARE = [

'django\_prometheus.middleware.PrometheusBeforeMiddleware',

# Autres middlewares...

'django\_prometheus.middleware.PrometheusAfterMiddleware',

]

**Configurez Sentry dans les paramètres Django :**

import sentry\_sdk

from sentry\_sdk.integrations.django import DjangoIntegration

sentry\_sdk.init(

dsn="LE DSN DEPUIS LE COMPTE SENTRY",

integrations=[DjangoIntegration()],

traces\_sample\_rate=1.0,

send\_default\_pii=True

)

Configurer les Logs (dans settings.py) :

LOGGING = {

'version': 1,

'disable\_existing\_loggers': False,

'formatters': {

'verbose': {

'format': '{levelname} {asctime} {module} {message}',

'style': '{',

},

'simple': {

'format': '{levelname} {message}',

'style': '{',

},

},

'handlers': {

'file': {

'level': 'DEBUG',

'class': 'logging.FileHandler',

'filename': os.path.join(BASE\_DIR, 'debug.log'),

'formatter': 'verbose',

},

},

'root': {

'handlers': ['file'],

'level': 'DEBUG',

},

'loggers': {

'django': {

'handlers': ['file'],

'level': 'DEBUG',

'propagate': True,

},

'django.request': {

'handlers': ['file'],

'level': 'ERROR',

'propagate': False,

},

},

}

**Mise en Place et Test du Monitorage :**

Exécutez les migrations pour initialiser la base de données :

python manage.py makemigrations

python manage.py migrate