

# Projet de Fin d'Études de Valentin Hernandez– Bloc 4

---

## Sommaire

1. Introduction
2. Présentation du projet
3. Supervision et alerte
4. Consignation des anomalies
5. Journal des versions
6. Perspectives d'amélioration
7. Difficultés rencontrées
8. Conclusion

## **1. Introduction**

Ce projet de fin d'études consiste à développer un site de musculation qui propose des conseils de nutrition et d'exercices, et qui intègre un chat en temps réel permettant aux utilisateurs de discuter entre eux ou avec un coach.

Le Bloc 4 vise à maintenir l'application en condition opérationnelle. Cela implique : assurer la supervision de l'application, consigner les anomalies détectées et tenir un journal des versions déployées.

## 2. Présentation du projet

Le projet repose sur une architecture Django (backend), utilisant Django Channels pour le chat en temps réel, une base de données SQLite en développement, et une interface simple en HTML/CSS avec un peu de JavaScript intégré dans les templates. Le code est versionné sur GitHub.

Fonctionnalités principales :

- Conseils de nutrition
- Conseils d'exercices
- Chat temps réel entre utilisateurs et coachs

Architecture simplifiée :

- Backend : Django + Channels
- Frontend : HTML/CSS + JS
- Base de données : SQLite
- Gestion du code : Git/GitHub

### 3. Supervision et alerte

Pour superviser l'application, un système de logging a été mis en place dans Django. La configuration LOGGING permet d'enregistrer les erreurs critiques dans un fichier 'logs/errors.log'. Ainsi, toute erreur 500 est automatiquement consignée.

Exemple de configuration dans settings.py :

```
1  
2 LOGGING = {  
3     'version': 1,  
4     'disable_existing_loggers': False,  
5     'handlers': {  
6         'file': {  
7             'level': 'ERROR',  
8             'class': 'logging.FileHandler',  
9             'filename': os.path.join(BASE_DIR, 'logs/errors.log'),  
10        },  
11    },  
12    'loggers': {  
13        'django': {  
14            'handlers': ['file'],  
15            'level': 'ERROR',  
16            'propagate': True,  
17        },  
18    },  
19 }
```

You, 23 hours ago • final ...

Figure 1 – Configuration du logging Django dans settings.py

Capture du fichier logs/errors.log après avoir déclenché une erreur volontaire (ZeroDivisionError). On y voit l'exception et la trace associée.

```
fitness_project > logs > # errors.log
1 Internal Server Error: /test-error/
2 Traceback (most recent call last):
3   File "C:\Users\valen\Desktop\projets\Musculation\.venv\Lib\site-packages\django\core\handlers\exception.py", line 55, in inner
4     response = get_response(request)
5   File "C:\Users\valen\Desktop\projets\Musculation\.venv\Lib\site-packages\django\core\handlers\base.py", line 197, in _get_response
6     response = wrapped_callback(request, *callback_args, **callback_kwargs)
7   File "C:\Users\valen\Desktop\projets\Musculation\fitness_project\training\views.py", line 19, in test_error
8     1 / 0 # division par zéro -> erreur 500
9     ~~~~~
10 ZeroDivisionError: division by zero
11 "GET /test-error/ HTTP/1.1" 500 70377
12
```

Figure 2 – Enregistrement d'une erreur 500 (ZeroDivisionError) dans errors.log

## 4. Consignation des anomalies détectées

Capture du fichier ANOMALIES.md contenant la fiche d'anomalie. La première entrée correspond au ZeroDivisionError, avec date, type, description, conditions, impact et statut. Chaque anomalie comporte : la date, le type d'erreur, la description, l'impact et le statut.

```
1  # Journal des anomalies
2
3  ## Anomalie 1
4  - **Date** : 18/08/2025
5  - **Type** : Erreur 500 - ZeroDivisionError
6  - **Description** : Division par zéro dans la vue `test_error`.
7  - **Conditions d'apparition** : accès à l'URL `/test-error/`.
8  - **Impact** : indisponibilité de la page (Internal Server Error).
9  - **Statut** : Corrigé (vue créée volontairement pour test des logs
10
11  ---
12
13  ## Anomalie 2
14  - **Date** : |
15  - **Type** : ...
16  - **Description** : ...
17  - **Conditions d'apparition** : ...
18  - **Impact** : ...
19  - **Statut** : ...
20
```

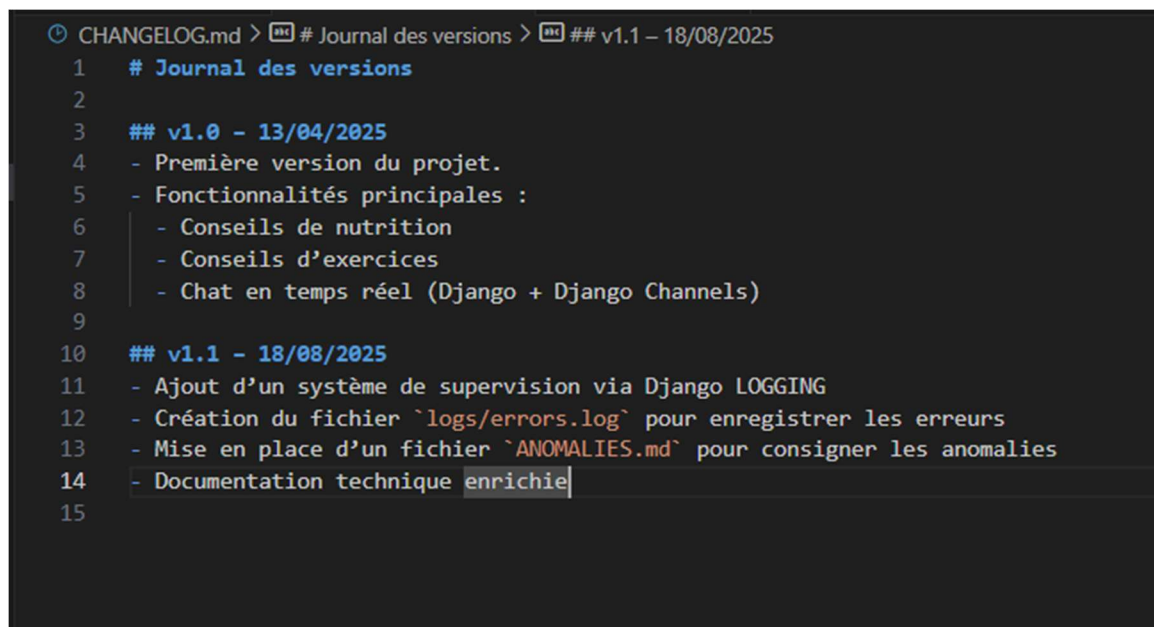
Figure 3 – Exemple de fiche d'anomalie consignée dans ANOMALIES.md

Ce suivi permet d'avoir une traçabilité claire des bugs rencontrés et corrigés, et d'améliorer la maintenance future.

## 5. Journal des versions déployées

Un fichier `CHANGELOG.md` documente les différentes versions de l'application et leurs évolutions.

Exemple de journal :



```
CHANGELOG.md > # Journal des versions > ## v1.1 – 18/08/2025
1  # Journal des versions
2
3  ## v1.0 – 13/04/2025
4  - Première version du projet.
5  - Fonctionnalités principales :
6    - Conseils de nutrition
7    - Conseils d'exercices
8    - Chat en temps réel (Django + Django Channels)
9
10 ## v1.1 – 18/08/2025
11 - Ajout d'un système de supervision via Django LOGGING
12 - Création du fichier `logs/errors.log` pour enregistrer les erreurs
13 - Mise en place d'un fichier `ANOMALIES.md` pour consigner les anomalies
14 - Documentation technique enrichie
15
```

Figure 4 – Journal des versions du projet dans CHANGELOG.md

La figure 5 illustre le comportement de l'application lorsqu'une erreur critique survient. Ici, une erreur volontaire de type `ZeroDivisionError` a été déclenchée dans la vue `/test-error/`.

Cette anomalie est visible côté utilisateur (erreur 500 dans le navigateur) et est simultanément enregistrée dans le fichier `errors.log`.

### ZeroDivisionError at /test-error/

division by zero

Request Method: GET  
Request URL: http://127.0.0.1:8000/test-error/  
Django Version: 5.1.4  
Exception Type: ZeroDivisionError  
Exception Value: division by zero  
Exception Location: C:\Users\valen\Desktop\projets\Musculation\fitness\_project\training\views.py, line 19, in test\_error  
Raised during: training.views.test\_error  
Python Executable: C:\Users\valen\Desktop\projets\Musculation\venv\Scripts\python.exe  
Python Version: 3.13.1  
Python Path: ['C:\\Users\\valen\\Desktop\\projets\\Musculation\\fitness\_project',  
'C:\\Users\\valen\\AppData\\Local\\Programs\\Python\\Python313\\python313.zip',  
'C:\\Users\\valen\\AppData\\Local\\Programs\\Python\\Python313\\DLLs',  
'C:\\Users\\valen\\AppData\\Local\\Programs\\Python\\Python313\\Lib',  
'C:\\Users\\valen\\AppData\\Local\\Programs\\Python\\Python313',  
'C:\\Users\\valen\\Desktop\\projets\\Musculation\\.venv',  
'C:\\Users\\valen\\Desktop\\projets\\Musculation\\.venv\\Lib\\site-packages']  
Server time: Tue, 19 Aug 2025 16:44:28 +0200

### Traceback

[Switch to copy-and-paste view](#)

```
C:\Users\valen\Desktop\projets\Musculation\.venv\Lib\site-packages\django\core\handlers\exception.py, line 55, in inner
55.         response = get_response(request)
                        ~~~~~
▶ Local vars

C:\Users\valen\Desktop\projets\Musculation\.venv\Lib\site-packages\django\core\handlers\base.py, line 197, in _get_response
197.         response = wrapped_callback(request, *callback_args, **callback_kwargs)
                        ~~~~~
▶ Local vars

C:\Users\valen\Desktop\projets\Musculation\fitness_project\training\views.py, line 19, in test_error
19.         1 / 0 # division par zéro => erreur 500
            ~~~~
▶ Local vars
```

### Request information

USER	valen
GET	No GET data
POST	No POST data
FILES	No FILES data

Figure 5 – Affichage d'une erreur 500 dans le navigateur



## 6. Perspectives d'amélioration

À l'avenir, plusieurs améliorations peuvent être envisagées :

- Intégration d'outils de monitoring temps réel (Prometheus, Grafana)
- Mise en place d'un pipeline CI/CD avec GitHub Actions
- Déploiement sur un hébergeur cloud (ex : Render, Heroku, AWS)
- Alertes automatisées envoyées par e-mail ou Slack en cas d'erreur critique
- Suivi utilisateur : ajout d'un formulaire pour signaler directement les bugs depuis le site.

## 7. Difficultés rencontrées

- Apprendre Django + Django Channels en peu de temps.
- Mise en place du chat en temps réel avec Django Channels (gestion des WebSockets, routing, consumers).
- Problèmes de compatibilité avec SQLite (utile en dev mais limité pour un futur déploiement).
- Tests unitaires : écrire des tests pour le chat WebSocket est plus complexe que pour des vues classiques.
- Configuration de Django LOGGING (il a fallu comprendre pourquoi les erreurs 404 n'étaient pas loggées et déclencher une vraie erreur 500).

## 8. Conclusion

Ce Bloc 4 a permis de franchir une étape importante dans la vie du projet en apportant une véritable dimension de suivi et de maintenance. La mise en place d'un système de supervision a montré qu'il est possible de détecter rapidement les incidents techniques et de les documenter, ce qui facilite leur résolution. Le suivi des anomalies et le journal des versions constituent désormais un cadre clair pour comprendre l'évolution de l'application et intervenir efficacement en cas de problème.

Au-delà de la technique, ce travail m'a sensibilisé à la nécessité d'anticiper la maintenance dès la conception d'un projet. Surveiller, consigner et documenter ne sont pas des tâches secondaires, mais des pratiques essentielles pour garantir la stabilité et la confiance des utilisateurs.

Dans une perspective future, il serait intéressant d'automatiser certaines tâches de supervision (alertes en temps réel, intégration continue, déploiement automatique) afin de rapprocher le projet des standards professionnels.

En résumé, ce Bloc m'a appris à aller au-delà du développement pur pour envisager le logiciel comme un service vivant, qui doit rester fiable et évolutif dans le temps.

