Rapport de test

Résultat du test et leur interprétation

```
library/tests/test_library.py::BookIntegrationTest::test_create_book PASSED [ 10%]
library/tests/test_models.py::test_author_label PASSED                   [ 20%]
library/tests/test_models.py::test_title_label PASSED                    [ 30%]
library/tests/test_models.py::test_author_max_length PASSED              [ 40%]
library/tests/test_models.py::test_title_max_length PASSED               [ 50%]
library/tests/test_models.py::test_object_name_is_author_title PASSED    [ 60%]
library/tests/test_search.py::SearchViewTest::test_search_view PASSED    [ 70%]
library/tests/test_search.py::SearchViewTest::test_search_view_no_results PASSED [ 80%]
library/tests/test_urls.py::test_book_info_url PASSED                    [ 90%]
library/tests/test_views.py::IndexViewTest::test_index_view PASSED       [100%]
```

Tests unitaires :

test_models.py « Passed »

L'interprétation de ce résultat montre qu'une série de tests unitaires est effectuée sur le modèle Book de l'application Django. Le test utilise la classe TestCase de Django pour vérifier divers aspects du modèle Book. La méthode setUpTestData() initialise une instance de Book avec un auteur et un titre spécifiques. Les tests vérifient ensuite que les étiquettes des champs 'author' et 'title' sont correctes en comparant leurs noms verbeux (verbose_name) aux valeurs attendues. De plus, les tests s'assurent que la longueur maximale des champs 'author' et 'title' est de 50 caractères. Enfin, un test vérifie que la méthode str de l'objet Book renvoie correctement une chaîne de caractères formatée combinant l'auteur et le titre du livre. Ces tests garantissent que le modèle Book est correctement configuré avec les champs et les comportements attendus.

test_urls.py « Passed »

L'interprétation de ce résultat montre qu'un test unitaire est effectué pour vérifier la configuration correcte de l'URL et de la vue associée dans une application Django. Le test utilise pytest avec le marqueur django_db pour accéder à la base de données. Une instance de Book est créée et enregistrée dans la base de données avec un auteur et un titre spécifiques. Ensuite, la fonction reverse génère l'URL en utilisant le nom de la vue "info" et l'ID du livre. Le test vérifie que l'URL générée correspond au chemin attendu "/{book.id}". Enfin, le test utilise la fonction resolve pour s'assurer que l'URL se résout bien vers la fonction de vue book_info. Cela garantit que la configuration des URL et des vues dans l'application est correcte.

Test d'intégration:

test_search.py « Passed »

L'interprétation de ce résultat montre qu'un ensemble de tests d'intégration est effectué pour vérifier le bon fonctionnement de la vue de recherche dans une application Django. La classe SearchViewTest utilise TestCase pour initialiser un client de test et créer un objet Book avec un titre et un auteur spécifiques. Le test test_search_view envoie une requête GET à la vue de recherche avec un paramètre de requête correspondant au titre du livre existant. Le test vérifie que la réponse a un statut HTTP 200 et qu'elle contient le titre et l'auteur du livre. Le test test_search_view_no_results envoie une requête GET avec un paramètre de requête ne correspondant à aucun livre dans la base de données et vérifie que la réponse a un statut HTTP 200 et contient le message "Aucun livre trouvé." Ces tests d'intégration assurent que la vue de recherche renvoie les résultats corrects et gère les cas où aucune correspondance n'est trouvée.

Tests avec fixitures:

test_models.py « Passed »

L'interprétation de ce résultat montre qu'un ensemble de tests unitaires, utilisant pytest et une fixture pour le modèle Book, est effectué pour vérifier divers aspects du modèle Book dans une application Django. La fixture book crée un objet Book avec un auteur et un titre spécifiques. Les tests vérifient ensuite que les étiquettes des champs 'author' et 'title' sont correctes en comparant leurs noms verbeux (verbose_name) aux valeurs attendues. De plus, les tests s'assurent que la longueur maximale des champs 'author' et 'title' est de 50 caractères. Enfin, un test vérifie que la méthode str de l'objet Book renvoie correctement une chaîne de caractères formatée combinant l'auteur et le titre du livre. Ces tests unitaires garantissent que le modèle Book est correctement configuré avec les champs et les comportements attendus.

test_urls.py « Passed »

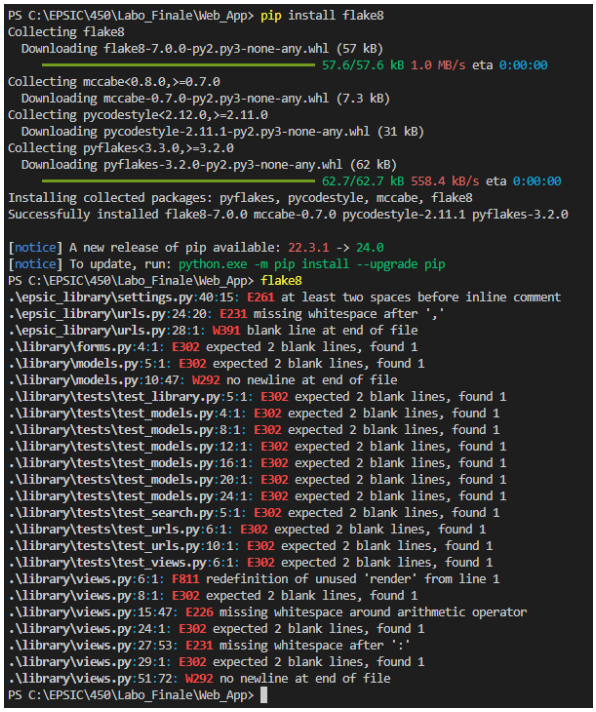
L'interprétation de ce résultat montre qu'un test d'intégration utilisant une fixture est effectué pour vérifier la configuration correcte de l'URL et de la vue associée dans une application Django. La fixture book crée un objet Book avec un auteur et un titre spécifiques. Le test test_book_info_url utilise la fonction reverse pour générer l'URL de la vue "info" en utilisant l'ID du livre, puis vérifie que l'URL générée correspond au chemin attendu "/{book.id}". Ensuite, le test utilise la fonction resolve pour s'assurer que l'URL se résout correctement vers la fonction de vue book_info. Le passage de ce test confirme que le routage et la résolution des vues dans l'application sont configurés correctement pour la vue "info" du modèle Book.

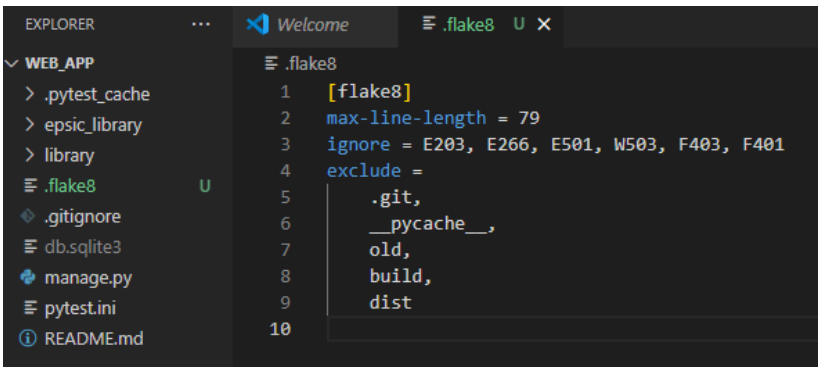
Test avec Mocks:

test_views.py « Passed »

L'interprétation de ce résultat montre qu'un test d'intégration est effectué pour vérifier la vue index d'une application Django en utilisant un mock pour le modèle Book. Le test utilise TestCase et Client de Django pour simuler une requête à la vue index. La méthode Book.objects.all est mockée pour renvoyer un objet dont la méthode count retourne 10. Le test envoie une requête GET à la vue index et vérifie que le contexte de la réponse contient bien le nombre de livres attendu, 10. Le passage de ce test confirme que la vue index traite correctement le nombre de livres retournés par le modèle Book et l'inclut correctement dans le contexte de la réponse, en s'assurant que le mock fonctionne comme prévu.

Contrôle statique de code:





Le résultat du contrôle statique de code avec Flake8 indique plusieurs violations des règles de style PEP8 dans le projet. Ces violations incluent : des espaces manquants avant les commentaires en ligne (E261), des espaces manquants après des virgules (E231), des lignes vides en trop à la fin des fichiers (W391), un nombre insuffisant de lignes vides entre les définitions de fonctions ou de classes (E302), l'absence de nouvelle ligne à la fin des fichiers (W292), la redéfinition non utilisée de la fonction 'render' (F811), et des espaces manquants autour des opérateurs arithmétiques (E226).