

Les 12 Pratiques XP

Table des matières

Les 12 Pratiques XP.....	1
1– Planning Game.....	1
11- Phase d'exploration.....	1
12- Phase d'engagement (Sprint Planning).....	2
2– Petites Releases.....	3
3– Utilisation de métaphores.....	3
4– Conception simple.....	3
5– Tests unitaires et tests unitaires (fonctionnels) / TDD (Test Driven Development).....	4
6– Refactoring.....	4
7– Programmation en binôme.....	4
8– Appropriation collective du code.....	5
9– Intégration continue.....	5
10– Rythme soutenable.....	6
11– Client sur site.....	6
12– Standard de code (compréhension partagée).....	6

1– Planning Game

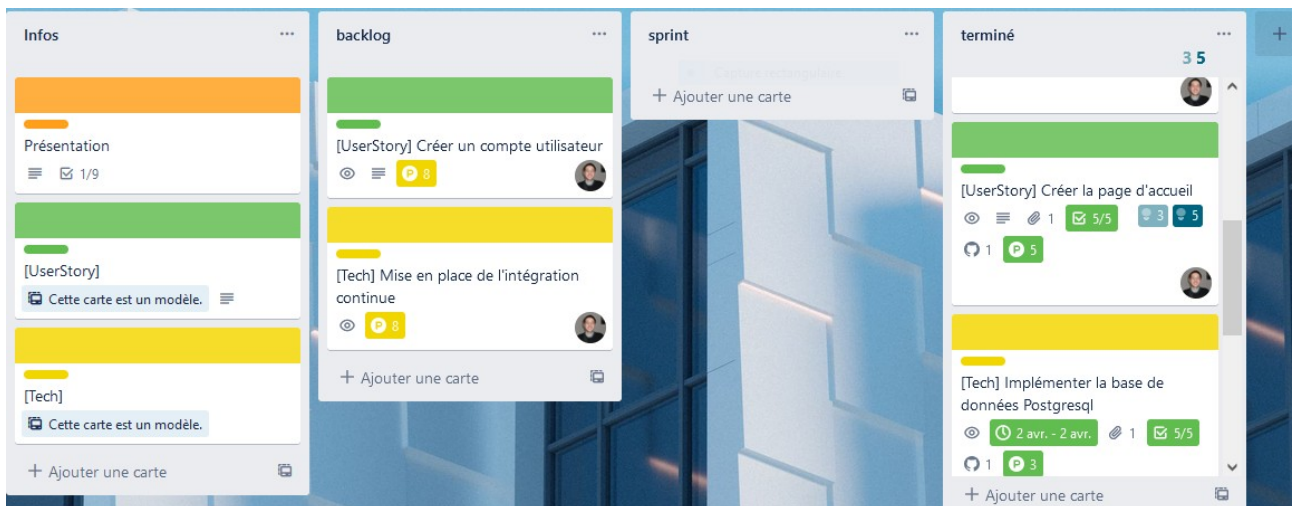
Cette pratique a pour but de planifier les **releases** (livraisons) pendant le déroulé d'un **sprint** (cycle de développement)

livrables et implémentations:

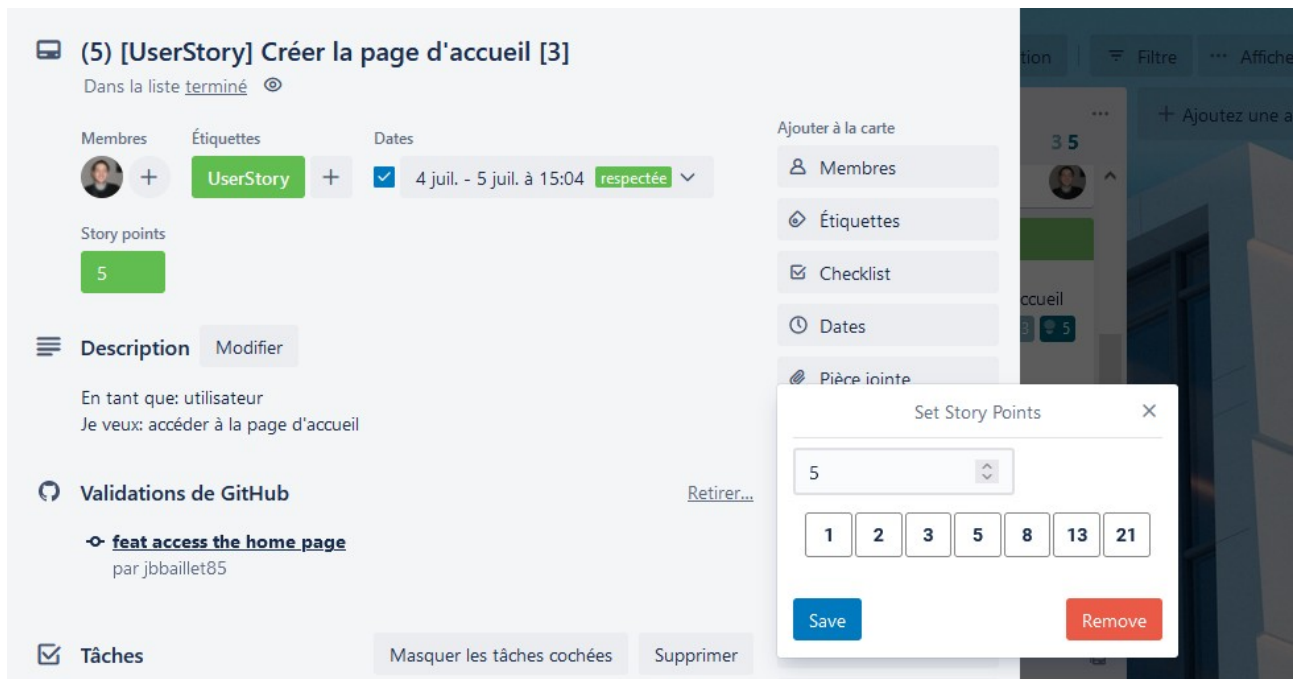
J'ai mis en place le planning game grâce à un tableau trello
(<https://trello.com/b/kXOWpHUQ/veilletraking>).

11- Phase d'exploration

Pendant la phase d'exploration, le client exprime ses besoins en termes de fonctionnalités. Pour cela, il les écrit sous forme de **user stories**, c'est à dire sous forme de courtes descriptions du comportement qu'il attend du système, exprimées avec un point de vue utilisateur. Ces "user stories" sont placées dans le **backlog**.



Au cours de cette même phase, le développeur attribut à chaque user story un **nombre de points de complexité**, fonction du temps qu'ils estiment nécessaire au développement des fonctionnalités contenues dans chaque user story. Cet étape est aussi appelé poker planning.



12- Phase d'engagement (*Sprint Planning*)

Dans la phase d'engagement, les user stories sont triées en fonction de la valeur qu'elles apportent au client et des risques encourus lors de leur développement. Ceci permet d'aboutir à un classement des user stories par ordre de priorité.

livrables et implémentations:

- User stories dans le tableau Kamban de Trello

2- Petites Releases (livraisons)

Pour une bonne gestion des risques, la sortie des releases (livraisons de fonctionnalités) doit intervenir le plus souvent possible. En conséquence, d'une version à l'autre, l'évolution doit être la plus petite possible, tout en s'efforçant d'apporter le plus de valeur ajoutée et des fonctionnalités dans leur intégralité avec une définition du fini. Intégrer les nouvelles fonctionnalités de façon continue et obtenir un retour rapide sur le travail effectué permet de corriger le tir très rapidement si quelque chose ne va pas.

livrables et implémentations:

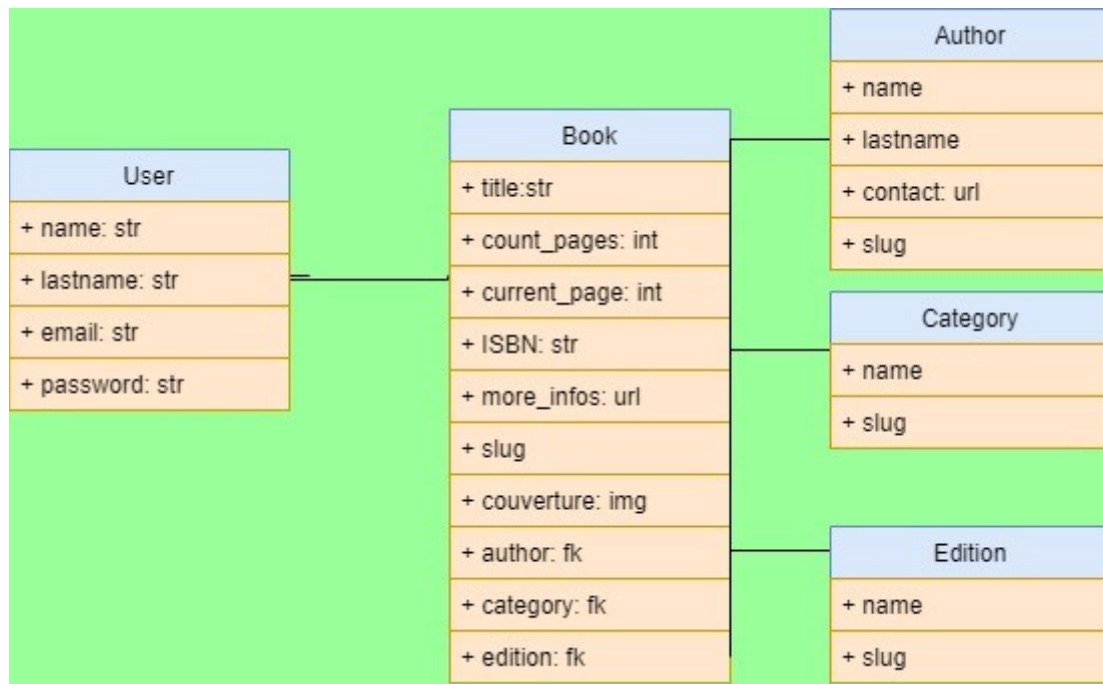
- commit feat <nom de la nouvelle fonctionnalité>
- Mise en place d'une définition of done" (ou définition du fini). <https://trello.com/c/P5lr7uSt/21-definition-of-done-d%C3%A9finition-du-fini>

3– Utilisation de métaphores

XP recommande d'utiliser des métaphores pour décrire l'architecture du système. De telles images permettent à tout le monde d'avoir une vision globale du système et d'en comprendre les éléments principaux ainsi que leurs interactions.

livrables et implémentations:

- diagramme UML entité association

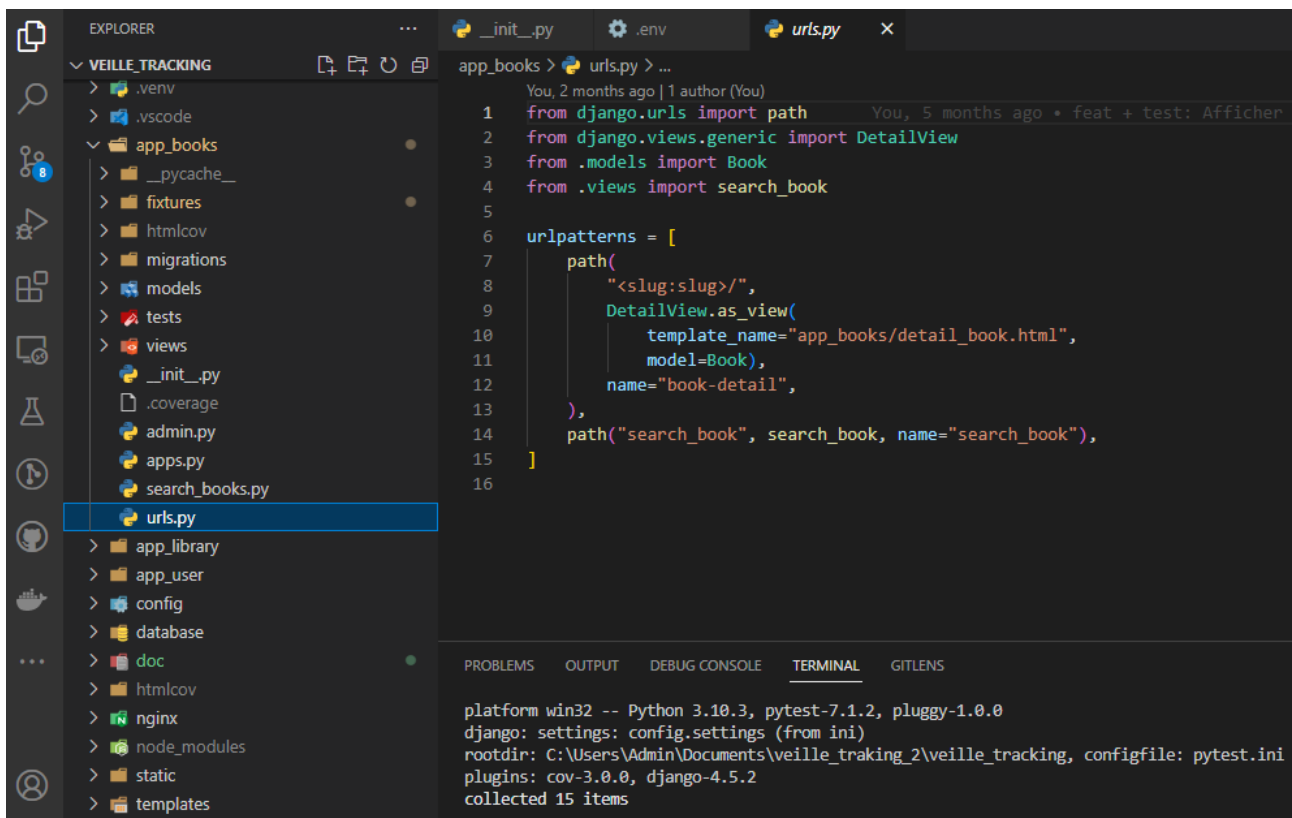


4– Conception simple

La simplicité est une des valeurs fondamentales d'XP. Il faut toujours développer la solution la plus simple possible et éviter de développer plus que ce dont on a besoin. Les seules exigences sont de satisfaire tous les tests, de ne jamais dupliquer une logique et d'utiliser le moins possible de classes et de méthodes.

livrables et implémentations:

- Création d'application django par responsabilité (app_user, app_books, app_library)
- Mise en place du patron de conception MVT (Models, Views, Templates)



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure for 'VEILLE_TRACKING'. It includes folders like .venv, .vscode, app_books, app_library, app_user, config, database, doc, htmlcov, nginx, node_modules, static, and templates. The 'app_books' folder is expanded, showing files like __pycache__, fixtures, htmlcov, migrations, models, tests, views, __init__.py, .coverage, admin.py, apps.py, and search_books.py. The 'urls.py' file is selected and open in the editor. The code in 'urls.py' defines URL patterns for a book detail view and a search view. The terminal at the bottom shows the output of a Django command, indicating the settings are configured and 15 items were collected.

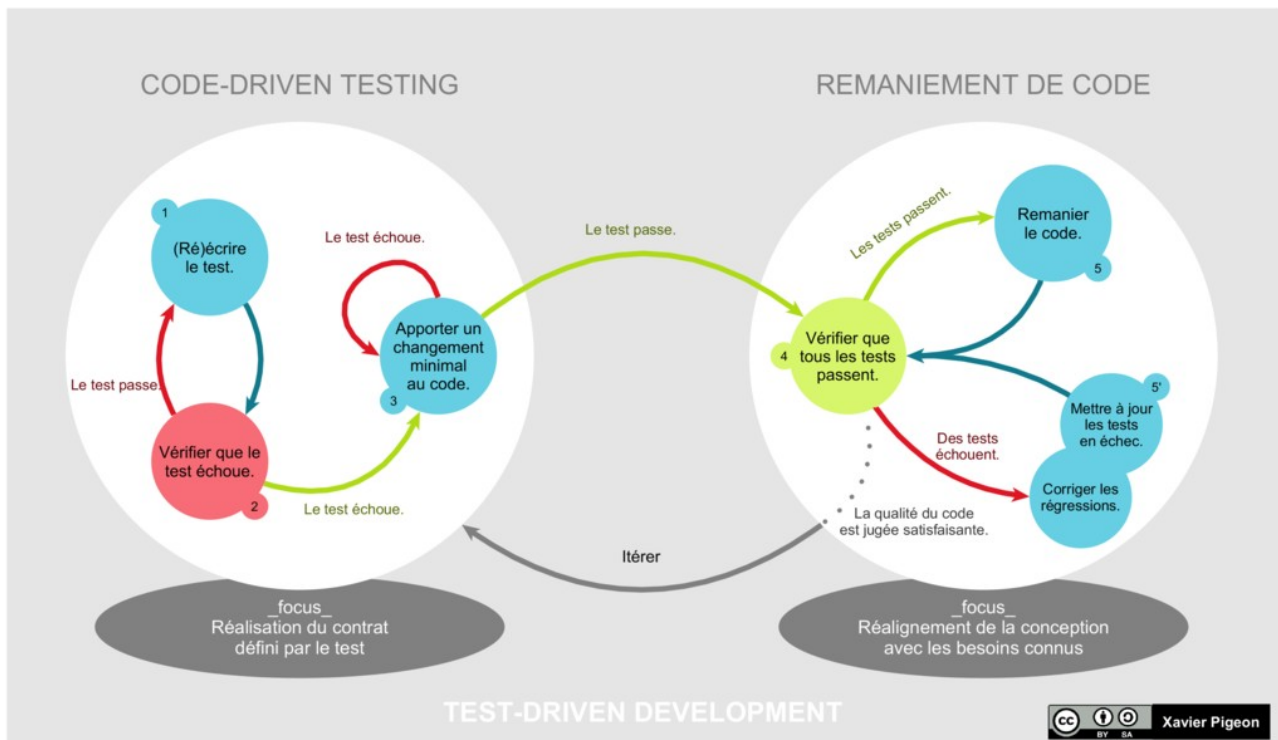
```
1 from django.urls import path
2 from django.views.generic import DetailView
3 from .models import Book
4 from .views import search_book
5
6 urlpatterns = [
7     path(
8         "<slug:slug>/",
9         DetailView.as_view(
10             template_name="app_books/detail_book.html",
11             model=Book),
12         name="book-detail",
13     ),
14     path("search_book", search_book, name="search_book"),
15 ]
```

platform win32 -- Python 3.10.3, pytest-7.1.2, pluggy-1.0.0
django: settings: config.settings (from ini)
rootdir: C:\Users\Admin\Documents\veille_traking_2\veille_tracking, configfile: pytest.ini
plugins: cov-3.0.0, django-4.5.2
collected 15 items

5– Tests unitaires et tests unitaires (fonctionnels) / TDD (Test Driven Development)

Les tests unitaires sont écrits et effectués par les développeurs pour vérifier le bon fonctionnement et la non régression des méthodes ou des classes.

Les tests fonctionnels sont conçus par le client et lui permettent d'une part de vérifier le fonctionnement global du système, de contrôler l'évolution du projet, et d'affiner l'expression de ses besoins.



livrables et implémentations:

- Mise en place de tests avec pytest-django

```
(veille_tracking) PS C:\Users\Admin\Documents\veille_tracking_2\veille_tracking> pipenv run pytest --cov=. --cov-report html
Courtesy Notice: Pipenv found itself running within a virtual environment, so it will automatically use that environment, instead of creating
its own for any project. You can set PIPENV_IGNORE_VIRTUALENVS=1 to force pipenv to ignore that environment and create its own instead. You
can set PIPENV_VERBOSITY=-1 to suppress this warning.
Loading .env environment variables...
===== test session starts =====
platform win32 -- Python 3.10.3, pytest-7.1.2, pluggy-1.0.0
django: settings: config.settings (from ini)
rootdir: C:\Users\Admin\Documents\veille_tracking_2\veille_tracking, configfile: pytest.ini
plugins: cov-3.0.0, django-4.5.2
collected 15 items

app_books\tests\test_SearchBooks.py . [ 6%]
app_books\tests\test_author_model.py . [ 13%]
app_books\tests\test_book_model.py . [ 20%]
app_books\tests\test_category_model.py . [ 26%]
app_books\tests\test_detail_book.py . [ 33%]
app_books\tests\test_edition_model.py . [ 40%]
app_books\tests\test_search_book_views.py . [ 46%]
app_library\tests\test_library.py ... [ 66%]
app_user\tests\tests_models_user.py ... [ 86%]
app_user\tests\tests_views_user.py . [ 93%]
tests\test_homepage.py . [100%]

----- coverage: platform win32, python 3.10.3-final-0 -----
Coverage HTML written to dir htmlcov

===== 15 passed in 5.56s =====
(veille_tracking) PS C:\Users\Admin\Documents\veille_tracking_2\veille_tracking>
```

- Mise en place d'une couverture de test avec pytest-cov

Coverage report: 95%

coverage.py v6.4.2, created at 2022-12-29 14:15 +0100

Module	statements	missing	excluded	coverage
app_books__init__.py	0	0	0	100%
app_books\admin.py	18	0	0	100%
app_books\apps.py	4	0	0	100%
app_books\migrations\0001_initial.py	6	0	0	100%
app_books\migrations__init__.py	0	0	0	100%
app_books\models__init__.py	56	0	0	100%
app_books\search_books.py	12	0	0	100%
app_books\tests__init__.py	0	0	0	100%
app_books\tests\test_SearchBooks.py	11	0	0	100%
app_books\tests\test_author_model.py	16	0	0	100%
app_books\tests\test_book_model.py	24	0	0	100%
app_books\tests\test_category_model.py	14	0	0	100%
app_books\tests\test_detail_book.py	31	0	0	100%
app_books\tests\test_edition_model.py	14	0	0	100%

6– Refactoring

Le but de cette pratique est de simplifier le code, tout en faisant en sorte que tous les tests soient satisfaits. Le refactoring tend à produire un code mieux pensé, plus modulaire, sans duplications de code et donc plus facile à maintenir. Le fait de livrer régulièrement permet de toujours disposer d'une base « saine » de code au cas où des changements seraient à inclure.

livrables et implémentations:

- Remplacer les dépendances dépréciées (exemple: `os.path` par `Path`) dans le code source, afin d'avoir une base de code toujours à jour, et migrer plus facilement.

- `commit - m "refactor: 'message'"`

7– Programmation en binôme

Toute l'écriture du code se fait à deux personnes sur une même machine, avec une seule souris et un seul clavier. On distingue deux rôles : le pilote (« driver »), celui qui a le clavier, cherche la meilleure approche sur une portion de code bien précise tandis que l'autre développeur, le « partner » peut observer avec beaucoup plus de recul et ainsi suggérer d'autres solutions.

livrables et implémentations:

Je n'ai pas pu travailler en binôme sur ce projet. À défaut de programmation en binôme, je partageais souvent avec mon frère sur le code produit. Ceci était simplifier grâce à docker afin d'avoir le même environnement.

8– Appropriation collective du code

Toute l'équipe est sensée connaître la totalité du code. Cela implique que tout le monde peut intervenir pour faire des ajouts ou des modifications sur une portion de code qu'il n'a pas écrit lui-même si cela s'avère nécessaire.

livrables et implémentations:

Cette pratique n'a pas pu être mise en place étant donné que je travaillais seul sur le projet.

9– Intégration continue

Après chaque fin de tâche, c'est à dire plusieurs fois par jour, le code nouvellement écrit doit être intégré à l'existant de manière à avoir à tout moment un existant fonctionnel qui passe avec succès tous les tests. Ainsi, quand une tâche est démarrée, elle peut se fonder sur la version la plus à jour de ce qui a déjà été fait. Il existe plusieurs logiciels d'intégration continue, tel que:

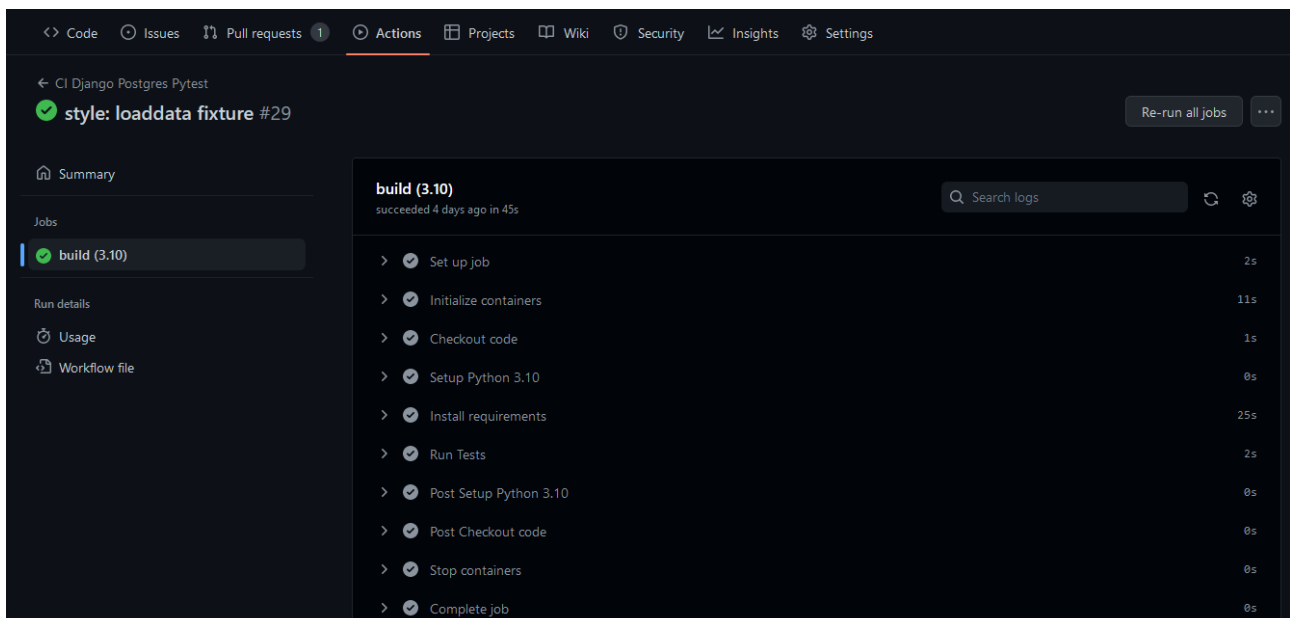
- Travis
- Gitlab-CI
- Circle-CI
- Github Actions

Étant donné que le code source est stocké sur Github, j'ai choisi d'utiliser Github actions pour la mise en place de l'intégration continue du projet. Je me suis aidé des sources suivantes:

- <https://www.codingforentrepreneurs.com/blog/django-github-actions/>
- https://docs.microsoft.com/fr-fr/learn/modules/github-actions-ci/?wt.mc_id=startup-21081-chmaneu

livrables et implémentations:

- Mise en place de la CI Github-Actions



10– Rythme soutenable

Il ne s'agit pas de chercher à travailler peu, mais les spécialistes d'XP ont pris conscience du fait que la surcharge de travail, en plus d'être désagréable, est néfaste. En effet, le code devient moins bien pensé, le refactoring est laissé de côté, ce qui conduit à une baisse de la qualité et à une recrudescence des bugs. L'équipe se doit de réagir en redéfinissant la quantité de user stories à implémenter au cours de l'itération.

livrables et implémentations:

J'ai du adapter le rythme de travail en fonction de mon planning professionnel, les activités de mes 3 enfants, entretenir ma vie conjugal et familial, et mes engagements associatifs. Le projet a avancé lentement, mais sûrement.

11– Client sur site

L'implication forte du client passe par la présence sur site d'une personne minimum à temps plein pendant toute la durée du projet. Cette personne doit avoir à la fois le profil type de l'utilisateur final et une vision plus globale du contexte pour pouvoir préciser les besoins, leur donner une ordre de priorité, les transcrire sous forme de user stories, et établir les tests fonctionnels.

livrables et implémentations:

Étant donné que j'étais à la fois client et développeur, j'ai endossé le rôle de product owner (chef de produit). Je testais régulièrement l'interface sur le navigateur, et je demandais l'avis de mon frère sur l'expérience utilisateur.

12– Standard de code (compréhension partagée)

Il est nécessaire de disposer de normes de nommage et de programmation pour que chacun puisse lire et comprendre facilement le code produit par les autres.

livrables et implémentations:

Dans le cadre d'un projet python il existe des normes de nommages explicités dans la PEP8. Grâce au module python flake8, j'ai pu vérifier que la base de code respecte bien la PEP8.