epsic ÉCOLE PROFESSIONNELLE LAUSANNE

LABO 1

450 - TESTER DES APPLICATIONS

VERONICA GETAZ - EPSIC 2024-2025

MODALITÉS

- Travail en groupe à choix libre : 1p à 2p

- Calendrier :
  - Travail en classe les 6.1.2025 + 13.1.2025 + 20.1.2025 + 3.2.2025 + 10.2.2025
  - Travail à la maison si nécessaire
  - Rendu sur Moodle :
    - Lien repo Github/Gitlab public ou partagé avec *vgepsic*
    - Max l**undi 10 février 2025 à 22h**

OBJECTIFS & NOTATION

	Objectifs	Notation
1	Définir une application simple	20%
2	Établir un Plan de Test	20%
BONUS	Mettre en place un pipeline CI/CD GIT pour vérifier l'exécution des tests	**+20%**
3	Développer du code de qualité en appliquant la méthode TDD	50%
4	Consigner les résultats dans un rapport de test	10%

- Critères de notation :
  - Respect des délais et consignes
  - Contenu pertinent, complet et original

1. DÉFINIR UNE APPLICATION SIMPLE

- Technologie à choix, application simple comme celle traité en cours
- Définir les spécifications produit : user story + critères d'acceptation (minimum 3)
- Format du délivrable : .md en tant que Read-me décrivant aussi :
  - Objectifs de l'app
  - Instructions d'installation & exécution (compatible machine Win11)
  - Architecture générale de l'application: technologies
- Exemples d'app :

Application	Critère d'acceptation	Input	Output
Statistiques de mots dans une phrase	Nombre de mots	"Bonjour à tous!"	3
Validateur de mot de passe	Longueur minimale	"P@ss"	Non valide
Vérification d'adresse email	Domaine valide	"jean@.com"	Invalide ("Domaine manquant")

2. ÉTABLIR UN PLAN DE TEST

- Établir un plan de test pour votre application au complet, avec :
  - Objectifs : Eléments testés (in scope) + Eléments ignorés (hors scope)
  - Environnement de test
  - Cas de test (min 3 cas de test pour chaque critère d'acceptation)

- Format : pdf @ racine du projet

BONUS : CRÉATION DE LA PIPELINE CI/CD

- Créer un workflow git qui exécutera vos tests à chaque push vers le repository
  - Voir une documentation des workflows gitHub + Python ici : https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
- Ajouter un badge à votre repository git, qui permet de voir le status de votre workflow de tests que vous venez de créer
  - Voir une documentation des badges ici : https://docs.github.com/en/actions/monitoring-and-troubleshooting-workflows/adding-a-workflow-status-badge

- **Workflow à créer avant d'itérer en TDD** (pour vérification automatique des tests)

3. DÉVELOPPER DU CODE DE QUALITÉ EN APPLIQUANT LA MÉTHODE TDD

- Utiliser le processus itératif TDD pour développer votre application

- Rendu TDD sur Git :

- 1 itération TDD par critère d'acceptation. Minimum 3 itérations
- 3 phases RGB par itération *:* 1 commit par phase. **Commentaire du commit claire avec numéro d'itération +**
  - R: titre ou IDs des cas de test
  - G: description brève (1 phrase) du code développé
  - B: quel(s) critère(s) des principes du Clean Code ont motivé à faire ce refactoring

4. CONSIGNER LES RÉSULTATS DANS UN RAPPORT TEST

- Si BONUS bien réalisé avant itérations TDD 🫶 :
  - RIEN (puisque la pipeline consigne déjà les résultats 🥰)

- Si pas de BONUS réalisé 💪 :
  - Générez un rapport de test depuis le plan de test, en le complétant avec :
    - Résultat obtenu
    - Analyse

PLANNING SUGGÉRÉ

Date	Activité
6 janvier	Choix app : techno + spécifications produit
13 janvier	Plan de Test
20 janvier	pipeline CI/CD
3 février	Itérations TDD
10 février	Itérations TDD + Rapport de test

- **Si Retard : 1 point de pénalité par tranche de 48h entamée**

Rendu max	Max (/6)
10-02-2025 @ 22h	6
12-02-2025 @ 22h	5
14-02-2025 @ 22h	4
16-02-2025 @ 22h	3
18-02-2025 @ 22h	2
APRES	1