KARSILAYAN Esin & TIMERA Abdourahmane

Conception Agile de Projets Informatiques

Planning Pocker

Dans le cadre de notre cours de conception agile de projets informatiques, nous devions réaliser un jeu de planning pocker. En informatique, le planning pocker est un jeu dont le but est de produire des estimations sur l'effort de développement des fonctionnalités. Il permet à tous les membres d'une équipe de s'exprimer en utilisant un système de vote.

Les choix techniques

Pour ce projet, nous avons décidé de travailler avec le framework Django (Python), pour mieux nous organiser et écrire notre code de manière plus efficace.

Pour le code, nous avons également utilisé du HTML et du CSS pour le rendu de la page web et le design de l'interface.

Nos principales fonctions sont :

- **index(request)** qui va permettre aux joueurs de charger la page d'accueil pour qu'ils puissent jouer
- **start_game(request)** qui démarre une nouvelle partie en prenant soin de nettoyer les données précédentes pour éviter les duplications, il va également permettre la création de nouveaux joueurs et va ensuite initialiser la session.
- **voter(request)** qui va gérer le tout le processus de vote et ce pour toutes les fonctionnalités, va vérifier l'unanimité et va les enregistrer. C'est également ici que les calculs vont être fait notamment pour les autres modes de jeu.
- **passer_a_la_suivante(request)** qui va se charger tout simplement de passer à la requête suivante lorsque tous les joueurs auront voté.
- **telecharger_donne(request)** qui va permettre de télécharger les donnés au format JSON

Mise en place de l'intégration continue

Pour ce qui est des tests, nous en avons fait pour les différentes fonctionnalités de notre jeu.

Nous avons tout d'abord créé notre fichier tests.py et nous avons importé les principales fonctionnalités de notre programme. Ainsi nous pouvons tester la création des joueurs, la fonctionnalité qui permet de voter et celle qui permet de passer à la fonctionnalité suivante. Pour chaque fonction qui existe déjà et qui a été importée, nous faisons des tests avec plusieurs valeurs.

```python
def joueur(self):
    data = {
        'mode': 'unanime',
        'player_names': 'Alice, Bob, Charlie',
    }
    response = self.client.post(self.url, data=data)
    self.assertEqual(response.status_code, 200)
    self.assertEqual(joueur.objects.count(), 3)
```

Image 1.Test pour la saisie des joueurs et du mode de jeu

```python
def backblog(self):
    backlog_data = json.dumps([
        {"id": 1, "title": "Fonct1", "description": "Description1"},
        {"id": 2, "title": "Fonct2", "description": "Description2"},
    ])
    backlog_file = SimpleUploadedFile("backlog.json", backlog_data.encode('utf-8'), content_type="application/json")
    data = {
        'mode': 'unanime',
        'player_names': 'Alice, Bob',
        'backlog': backlog_file,
    }
    response = self.client.get(reverse('backblog'))
    self.assertEqual(response.status_code, 200)
    self.assertTemplateUsed(response, 'pokergame/backblog.html')
```

Image 2. Test pour le chargement du backlog

```python
class voter :
    def debut(self):
        self.joueur1 = joueur.objects.create(nom='Alice')
        self.joueur2 = joueur.objects.create(nom='Bob')
        self.fonctionnalite1 = fonctionnalite.objects.create(id=1, titre='Fonction 1', description='Desc 1')
        self.url = reverse('voter')
        self.client.session['joueurs'] = [self.joueur1.id, self.joueur2.id]
        self.client.session['current_joueur_index'] = 0
        self.client.session['current_fonctionnalite_index'] = 0
        self.client.session['mode'] = 'unanime'
        self.client.session.save()

    def creer_vote(self) :
        data = {'button_value': '5'}
        response = self.client.post(self.url, data=data)
        self.assertEqual(response.status_code, 200)
        self.assertEqual(vote.objects.count(), 1)
```

Image 3. Test pour le vote des joueurs

```python
class passer_fonctionnalite_suivante :
    def setUp(self):
        self.fonctionnalite1 = fonctionnalite.objects.create(id=1, titre='Fonct1', description='Description1')
        self.fonctionnalite2 = fonctionnalite.objects.create(id=2, titre='Fonction2', description='Description2')
        self.url = reverse('passer_a_la_suivante')
        self.client.session['current_fonctionnalite_index'] = 0
        self.client.session.save()

    def test_pass_to_next_feature(self):
        response = self.client.post(self.url)
        self.assertEqual(response.status_code, 200)
        self.assertEqual(self.client.session['current_fonctionnalite_index'], 1)
```

Image 4. Test pour passer à la fonctionnalité suivante