**Laboratoire #13**

Tests

 

**Résumé de la tâche ⛳🏌️‍♀️🏌️‍♂️**

* Il y a des **golfeurs** et des **scores par trou**. À chaque fois qu’un golfeur complète un trou, on lui ajoute une rangée dans la table ScoreTrou. Si vous ne savez pas comment fonctionnent les scores au golf :
  + **Score** : Pour chaque trou, le but est de réussir en un certain nombre de coups. (Ex : 3 coups) Si un joueur réussit en 2 coups, son score sera **-1**. Si un joueur réussit en 5 coups, son score sera **+2**. (Donc les scores négatifs sont bons et les scores positifs sont mauvais !)
  + **Terme** : Les scores sont associés à des noms au golf : 0 = Par, -1 = Birdie, -2 = Eagle, +1 = Bogey, +2 = Double bogey, etc.
* Le projet est déjà « complet », on va surtout créer quelques tests (5) pour se familariser avec **xUnit Test**.

**Étape 0 - Mystery Box SQL 📦💾**

* Ouvrez la solution dans Visual Studio.
* Créez la base de données du labo avec la commande : **CREATE DATABASE Labo13**
* Exécutez les migrations avec Evolve : (La solution n’est pas dans le même dossier que le projet, donc commencez par faire **cd .\Labo13\**)

**evolve migrate sqlserver -c "Data Source=.\SQLEXPRESS;Initial Catalog=Labo13;Integrated Security=True;Persist Security Info=False;Pooling=False;MultipleActiveResultSets=False;Encrypt=False;TrustServerCertificate=False" -s Golf --target-version 1.1**

* **5-6**  Une image contenant texte

  Description générée automatiquementDans la même solution, créez un nouveau projet **xUnit Test**.

**Étape 1 - Tests unitaires 1️⃣**🤏

* Créez une classe **GolfController\_TestsUnitaires** dans le projet xUnit. Il n’y a rien à injecter dans cette classe et pas besoin de constructeur.
* **7-25**  Test #1 ⚙ : Testez l’action **ScoresTrou**. On veut s’assurer que si jamais la vue est modifiée, l’action continuera de fonctionner.
  + Une vue est-elle bel et bien retournée ?
  + Le Model envoyé à la vue Razor est-il bel et bien une liste de ScoreTrou ?
* **7-25**  Test #2 ⚙ : Testez l’action **DeleteScoreTrou**. Le but est de tester le scénario où on lui envoie un id invalide. (Ex : 500pu)
  + Est-ce qu’il y a bel et bien une redirection vers l’action ScoresTrou ?

**Étape 2 - Tests d’intégration** 💿🤝

* **27-28**  Commencez par préparer une classe qui permettra de rendre le **DbContext** accessible aux classes de test.
  + Pour ce labo, vous pouvez utiliser la BD Labo13, pas besoin de faire une BD de test.
* **30**  Créez une classe **GolfController\_TestsIntegration** dans le projet xUnit. Il faudra injecter la classe que vous venez de créer pour avoir accès au **DbContext**.
* **31-32**  Test #1 ⚙ : Testez l’action **Index**. On veut être sûr que la vue SQL qu’elle utilise reste toujours fonctionnelle au fil de l’évolution de la BD.
  + Le Model envoyé à la vue est-il bel et bien une List<VwDetailsScoreGolfeur> ?
* **33-34**  Test #2 ⚙ : Testez l’action **DeleteScoreTrou**. Si jamais on fait des changements sur la table ScoreTrou dans le futur, on veut s’assurer que la suppression fonctionne encore.
  + La quantité de ScoreTrou dans la BD a-t-elle bel et bien diminué de 1 ?
* **33-34**  Test #3 ⚙ : Testez l’action **CreateScoreTrou**. On veut être sûr que la procédure stockée et le déclencheur impliqués sont toujours bel et bien fonctionnels lors de cette transaction.
  + Le **CreerScoreTrouVM** à fournir à l’action est pour **Tiger Woods**, avec le score **-2**.
  + La quantité de ScoreTrou dans la BD a-t-elle bel et bien augmenté de 1 ?
  + Le nouveau score total de Tiger Woods est-il bel et bien -2 ?
  + Le nombre de trous de Tiger Woods est-il bel et bien 6 ?

Vous avez complété le dernier labo de la session !

Il ne vous reste plus que le **TP2 Partie 2** et c’est terminé 👋🙏