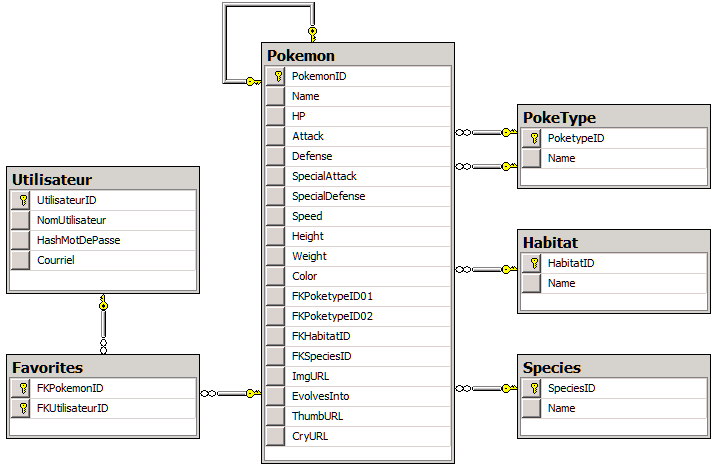
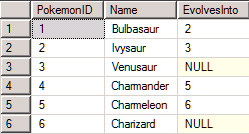
# Présentation

* Modèle de données



* Un pokémon appartient à une espèce, un habitat et un ou deux types (FKPokeType02 est donc possiblement null).
* La relation récursive Pokémon 🡸🡺 Pokémon permet d'établir une relation de l'évolution du pokémon. La clé migrée est EvolvesInto (plutôt que l'habituel FKPokemonID).
* Par exemple, Bulbasaur évolue en Ivysaur, qui lui-même évolue en Venusaur qui est son évolution finale.



# Accès aux données et gestion d'erreur

* Les entités Pokemon, PokeType, Species, Habitat, Favorites et Utilisateur vous sont fournis de même que les dépôts correspondants.
* Comme l'accès aux données est centralisé par une entité Pokedex, qui permet de manipuler toutes les entités, il ne devrait pas être nécessaire d'instancier directement des dépôts spécifiques.
* Par exemple, pour créer un utilisateur, plutôt que d'instancier UtilisateurRepository, il est possible d'écrire…

Utilisateur u = new Utilisateur();

u.NomUtilisateur = "AshKetchum";

//[...]

Pokedex pokedex = new Pokedex();

pokedex.Utilisateurs.Add(u);

* Les entités sont aussi semi-intelligentes et ont une certaine compréhension de leurs relations avec les entités voisines. Par exemple, pour connaître le nom de l'espèce d'un pokémon (qui est dans Species.Name), il est possible d'écrire…

Pokedex pokedex = new Pokedex();

Pokemon Bulbasaur = pokedex.Pokemons.Find(1);

Debug.WriteLine(Bulbasaur.Species.Name);

* Ou encore, pour obtenir les pokémons favoris d'un utilisateur…

Pokedex pokedex = new Pokedex();

Utilisateur u = pokedex.Utilisateurs.Find("AshKetchum");

List<Pokemons> = u.Favorites.List();

## Erreurs se produisant au niveau de la base de données.

* La base de données a été conçue pour éviter d'avoir des données incohérentes et il peut se produire des situations où l'ajout, la modification ou la suppression d'un enregistrement sera refusé.
* Par exemple, si un utilisateur modifie un pokémon et lui donne un nom identique à un pokémon existant, l'opération UPDATE échoue puisqu'un index unique a été créé sur le champ Name de la table Pokémon.
* De la même façon, si utilisateur essaie de supprimer un Habitat mais que cet habitat est relié à un pokémon, l'opération n'est pas permise.
* Plutôt qu'écrire le code nécessaire pour prévenir toutes ces erreurs avant qu'elles ne se produisent, il sera plus facile de tenter de réaliser l'opération dans un try/catch et d'afficher le message résultant à l'utilisateur.
* La couche d'accès a été écrite de façon à ce que les erreurs se produisant au niveau de la base de données génère des exceptions de type PokedexFailedOperationException.
* Pour gérer ces erreurs, vous devez donc, à chaque appel à la couche de données

try {

pokedex.*Entite.OperationTouchantLaBd()*

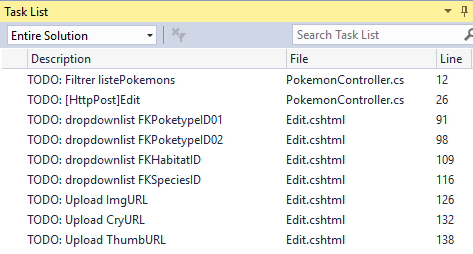
} catch (PokedexFailedOperationException e) {

ModelState.AddError(e.Message);

}

# Tâches à réaliser.

* Les tâches à réaliser débutent à 1.1.2 (gestion utilisateur)
* Dans Visual Studio, ouvrez la liste de tâche de façon à pouvoir parcourir les repères (bookmarks) sur les différentes sections de code à réaliser.



# 

# Gestion Utilisateur

## Gestion de l'accès

* Marquez le contrôleur UtilisateurController de l'attribut [Authorize]
* Marquez les méthodes appropriées de l'attribut [AllowAnnonymous]

## Menu Login (MenuLogin.cshtml)

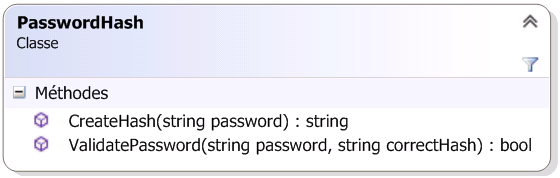
* Modifiez MenuLogin.cshtml (dossier Views\Shared) de façon à ce que seuls les choix Login et Inscription soient visibles à un utilisateur qui n'est pas authentifié sur le site.
* Modifiez MenuLogin.cshtml de façon à ce que seuls les choix Compte, Logout et Favoris soient visibles à un utilisateur qui est authentifié sur le site.

## Inscription

* Modifiez le code de la méthode [HttpPost] Inscription du contrôleur UtilisateurController de façon à ce qu'un utilisateur soit automatiquement connecté après une inscription réussie. Comme l'utilisateur n'a pas pu choisir s'il désire rester connecté ou non sur l'écran d'inscription, il est préférable que le cookie soit détruit à la fin de la session (ou au moment du logout).

## Stockage du mot de passe

* Le projet tel que distribué stocke les mots de passes des utilisateurs directement dans la base de données. Il est préférable de stocker le résultat d'une fonction de Hashing plutôt que le mot de passe directement.
* Vous devez utiliser la classe PasswordHash pour stocker le hash plutôt que le mot de passe des utilisateurs dans la base de données.
* La classe PasswordHash vous offre deux méthodes pour arriver à ce résultat, CreateHash et ValidatePassword.



* Modifiez les méthodes Add et Update de UtilisateurRepository de façon à transformer le mot passe donné par l'utilisateur en hash à l'aide de la fonction CreateHash.
* Utilisez ValidatePassword pour valider un mot de passe entré à la fenêtre de login contre un hash obtenu de la bd. Le mot de passe est validé dans la fonction ValidateurLoginView de la classe LoginView.
* Notez que la vue Edit ne sera plus fonctionnelle après ce changement.
* Après avoir fait ce changement, vous allez devoir créer un nouvel utilisateur pour pouvoir vous authentifier sur le site (à moins que le cookie localhost subsiste).

# Modifier un Pokémon

* Vous devez vous-même créer la méthode [HttpPost] Edit du contrôleur Pokemon pour réaliser les fonctionnalités décrites dans la section 1.1.6.

## Habitat, Species et 2x PokeType[[1]](#footnote-1)

* Vous devez intégrer dans la vue des dropdownlist pour permettre à l'utilisateur de modifier l'espèce (species), l'habitat et le (ou les) types.
* L'emplacement où vous devez insérer le dropdownlist est indiqué dans les commentaires.
* Notez que la deuxième relation à l'entité PokeType est optionnelle, et par conséquent FKPokeTypeID02 est nullable. Assurez-vous que le dropDownList permet à l'utilisateur de laisser le 2e PokeType vide.

## EvolvesInto

* Vous devez intégrer dans la vue un dropdownlist pour permettre à l'utilisateur de sélectionner le pokémon vers lequel il peut évoluer.
* L'emplacement où vous devez insérer le dropdownlist est indiqué dans les commentaires.
* Prévoyez que certains pokémons n'évoluent pas (permettre choix vide dans le dropdownlist).

## Image et Son

* Chaque pokémon a trois fichiers qui lui est rattaché. La première est une image grand format (ci-après nommée ImgUrl), la deuxième est une image de petite taille (ci-après nommée ThumbURL) et la dernière est un fichier de son de format ogg (ci-apès nommé CryURL).
* Vous devez ajouter le code nécessaire au traitement des fichiers ImgURL, ThumbURL et CryURL dans la vue Edit du contrôleur Pokémon.
* Les fichiers ne sont pas sauvegardés à l'intérieur de la base de données, on conserve plutôt un lien vers chacun des fichiers.
* Sauvegarder les fichiers dans les dossiers prévus à cette fin (cries, img, sprite).
* Lors de la modification, l'utilisateur a le choix de spécifier ou non un fichier pour ImgURL, ThumbURL et/ou CryURL.
* Si l'utilisateur envoi un fichier, alors on veut sauvegarder ce nouveau fichier dans le dossier. Le fichier reçoit le nom correspondant au PokemonID du pokémon sauvegardé (le fichier /Content/img/85.png appartient au pokémon dont le PokemonID est 85).
* Si l'utilisateur n'envoie pas de fichier pour une propriété donnée, on laisse l'ancien fichier intact.
* Finalement, si la mise-à-jour n'a pas générée une erreur, retourner à la liste de pokémons.

# Liste Pokémon (/Pokemon/Index)

* Affichez l'image (thumbnail correctement)
* Ajoutez le code nécessaire pour rendre le filtre de la liste de pokémons fonctionnel
* Vous n'avez pas à écrire de nouvelles fonctions dans les repository, filtrez plutôt la liste dans le contrôleur avec la méthode where.
* [Enumerable.Where](https://docs.microsoft.com/en-us/dotnet/api/system.linq.queryable.where)

# Liste Pokemon par Habitat

* Cette vue ne sera pas intégrée au reste du programme. (Cette question permet de réviser les vues partielles)
* L'objectif est d'afficher une liste d'habitat et des pokémons qui sont reliés à cette habitat.
* Dans le contrôleur Pokémon, écrire la méthode [HttpGet] PokemonPourHabitat.
* Dans la vue Index de Habitat, intégrer la vue partielle retournée par PokemonPourHabitat.

1. Prenez exemple de la vue Index du contrôleur Pokemon pour la creation des dropdownlist. [↑](#footnote-ref-1)