#### QuestionPropositon

##### Table

|  |  |  |
| --- | --- | --- |
| QuestionProposition | | |
| Primary Key | Id | int, not null |
| Foreign Key | QuestionId | int, not null |
|  | NameEnglish | nvvarchar(max), null |
|  | NameFrench | nvarchar(max), null |
|  | NameDutch | nvarchar(mac), null |

##### QuestionPropositionTO

namespace OnlineServices.Common.EvaluationServices.TransfertObjects

{

public class QuestionPropositionTO : IEntity<int>

{

public int Id {get; set;}

public int QuestionId {get; set;}

public int Position {get; set;}

public MultiLanguageString Libelle {get; set;}

}

}

##### QuestionPropositionEF

namespace EvaluationServices.Datalayer.Entities

{

public class QuestionPropositionEF : IEntity<int>

{

[Key]

public int Id

[ForeignKey(“Questions”)]

public int QuestionId {get; set;}

public QuestionEF Question {get; set;}

public int Position {get; set;}

public string NameEnglish {get; set;}

public string NameFrench {get; set;}

public string NameDutch {get; set;}

}

}

##### QuestionPropositionExtensions

namespace EvaluationServices.DataLayer.Extensions

{

public static QuestionPropositionExtentions

{

public static QuestionPropositionTO ToTransfertObject(this QuestionPropositionEF proposition)

{

if (proposition is null) throw new ArgumentNullException(nameof(proposition));

return new QuestionPropositionTO

{

Id = proposition.Id

QuestionId = proposition.QuestionId

Position = proposition.Position

Libelle = new MultiLanguageString(

proposition.NameEnglish,

proposition.NameFrench,

proposition.NameDutch);

};

}

public static QuestionPropositionEF ToEF(this QuestionPropositionTO proposition)

{

if (proposition is null) throw new ArgumentNullException(nameof(proposition));

return new QuestionPropositionEF

{

Id = proposition.Id,

QuestionId = proposition.QuestionId,

Position = proposition.Position,

NameEnglish = proposition.Libelle.English,

NameFrench = proposition.Libelle.French,

NameDutch = proposition.Libelle.Dutch

}

}

}

}

###### IQuestionPropositionRepository

namespace OnlineServices.Common.EvaluationServices.Interfaces.Repositories

{

public interface IQuestionPropositionRepository : IRepository<QuestionPropositionTO, int>

{

// méthodes spécifiques au QuestionPropositionRepository à rajouter ?

}

}

###### ESUnitOfWork

namespace EvaluationServices.DataLayer

{

public class ESUnitOfWork : IESUnitOfWork, IDisposable

{

...

// ajout QuestionPropositionRepository

private IQuestionPropositionRepository questionRepository;

public IQuestionPropositionRepository QuestionPropositionRepository

=> new QuestionPropositionRepository(evaluationContext);

...

}

}

###### QuestionPropositionRepository

namespace EvaluationServices.DataLayer.Repositories

{

public class QuestionPropositionRepository : IQuestionPropositionRepository

{

// context

private readonly EvaluationContext evaluationContext;

// constructeur

public QuestionPropositionRepository(Evaluation context)

{

evaluationContext =

context ?? throw new ArgumentNullException($”{nameof(context)} in EvaluationRepository”);

}

// Crud

public QuestionPropositionTO Add(QuestionProposition Entity)

=> evaluationContext.QuestionProposition.Add(Entity.ToEF()).Entity.ToTransfertObject();

public IEnumerable<QuestionPropositionTO> GetAll()

=> evaluationContext.QuestionProposition.Select(q => q.ToTransfertObject()).ToList()

public QuestionPropositionTO GetByID (int id)

=> evaluationContext.QuestionProposition.FirstOrDefault(q => q.Id == Id).ToTransfertObject();

public bool Remove(QuestionPropositionTO entity)

{

var toRemove = evaluationContext.QuestionProposition.FirstOrDefault(q => q.Id == entity.Id);

var removed = evaluationContext.QuestionProposition.Remove(toRemove);

return (removed.State == EntityState.Deleted);

}

public bool Remove(int Id)

{

var toRemove = evaluationContext.QuestionProposition.FirstOrDefault(q => q.Id == Id);

var removed = evaluationContext.QuestionProposition.Remove(toRemove);

return (removed.State == EntityState.Deleted);

}

}

}