***// this code is nearly identical to all codes in Repositories******, up to the getters***

***// can the codes be collapsed?***

***//***

***ICensusDivisionRepository.cs***

***// each of these import declarations should be commented –***

***// - what are we importing from each module?***

***// what do each of the imported functions do, in one phrase?***

***//***

using CPH\_IVT.Models;

using System.Collections.Generic;

using System.Threading.Tasks;

***// in what follows, do we need soft deletions – i.e., marking of fields as inaccessible – as well as hard deletions?***

***// what impact might deleting a division have on the database?***

***//***

namespace CPH\_IVT.Services.MongoDB.Repository

{

/// <summary>

/// Provides a mechanism for NoSQL CRUD operations on <see cref="CensusDivision"/> documents.

/// </summary>

public interface ICensusDivisionRepository

{

/// <summary>

/// Asynchronous creation of a <see cref="CensusDivision"/> object.

/// </summary>

/// <param name="censusDivision"><see cref="CensusDivision"/></param>

Task CreateAsync(CensusDivision censusDivision);

Task CreateBulkAsync(ICollection<CensusDivision> censusDivisions);

/// <summary>

/// Asynchronous retrieval of all <see cref="CensusDivision"/> objects from a MongoDB database.

/// </summary>

/// <returns>A collection of <see cref="CensusDivision"/> objects</returns>

Task<ICollection<CensusDivision>> GetAllAsync();

/// <summary>

/// Asynchronous retrieval of a <see cref="CensusDivision"/> object with <see cref="CensusDivision.Number"/> matching <paramref name="divisionNumber"/>.

/// </summary>

/// <param name="divisionNumber"><see cref="CensusDivision.Number"/></param>

/// <returns>A <see cref="CensusDivision"/> object</returns>

Task<CensusDivision> GetByDivisionNumberAsync(string divisionNumber);

Task<ICollection<State>> GetAllStatesByDivsionNumberAsync(string divisionNumber);

Task<ICollection<County>> GetAllCountiesByDivisionNumberAsync(string divisionNumber);

Task<ICollection<HealthIndicator>> GetAllHealthIndicatorsByDivisionNumberAsync(string divisionNumber);

/// <summary>

/// Asynchronous update of a <see cref="CensusDivision"/> object.

/// </summary>

/// <param name="censusDivision"><see cref="CensusDivision"/></param>

/// <returns><see langword="true"/> if successful; <see langword="false"/> otherwise</returns>

Task<bool> UpdateAsync(CensusDivision censusDivision);

/// <summary>

/// Asynchronous hard deletion of a <see cref="CensusDivision"/> object with <see cref="CensusDivision.Number"/> matching <paramref name="divisionNumber"/>.

/// </summary>

/// <param name="divisionNumber"><see cref="CensusDivision.Number"/></param>

/// <returns><see langword="true"/> if successful; <see langword="false"/> otherwise</returns>

Task<bool> DeleteAsync(string divisionNumber);

Task<ICollection<CensusDivision>> GetAllDivisionsByRegionNumberAsync(string regionNumber);

}

***}***

***CensusDivisionRepository.cs***

***// each of these import declarations should be commented –***

***// - what are we importing from each module?***

***// what do each of the imported functions do, in one phrase?***

***//***

using CPH\_IVT.Models;

using CPH\_IVT.Services.MongoDB.Context;

using MongoDB.Driver;

using System.Collections.Generic;

using System.Threading.Tasks;

***// the use of await below concerns me. Is this an indefinite wait? Does the wait ever time out?***

***//***

***// where is CensusDivisionContext.CensusDivisions.InsertOneAsync() defined?***

***// where is CensusDivisionContext.CensusDivisions.InsertManyAsync() defined?***

***// where is CensusDivisionContext.CensusDivisions.Find().ToListAsync() defined?***

***// where is CensusDivisionContext.CensusDivisions.Find().FirstOrDefaultAsync() defined?***

***//***

namespace CPH\_IVT.Services.MongoDB.Repository

{

/// <summary>

/// Represents a realization of <see cref="ICensusDivisionRepository"/>.

/// </summary>

public sealed class CensusDivisionRepository : ICensusDivisionRepository

{

/// <summary>

/// <see cref="ICensusDivisionContext"/>

/// </summary>

private readonly ICensusDivisionContext \_context;

/// <summary>

/// Parameterized constructor.

/// </summary>

/// <param name="context"><see cref="ICensusDivisionContext"/></param>

public CensusDivisionRepository(ICensusDivisionContext context) { \_context = context; }

/// <summary>

/// Asynchronous creation of a <see cref="CensusDivision"/> object.

/// </summary>

/// <param name="censusDivision"><see cref="CensusDivision"/></param>

public async Task CreateAsync(CensusDivision censusDivision) { await \_context.CensusDivisions.InsertOneAsync(censusDivision); }

public async Task CreateBulkAsync(ICollection<CensusDivision> censusDivisions)

{ await \_context.CensusDivisions.InsertManyAsync(censusDivisions); }

/// <summary>

/// Asynchronous retrieval of all <see cref="CensusDivision"/> objects from a MongoDB database.

/// </summary>

/// <returns>A collection of <see cref="CensusDivision"/> objects</returns>

public async Task<ICollection<CensusDivision>> GetAllAsync() { return await \_context.CensusDivisions.Find(\_ => true).ToListAsync(); }

/// <summary>

/// Asynchronous retrieval of a <see cref="CensusDivision"/> object with <see cref="CensusDivision.Number"/> matching <paramref name="censusDivisionNumber"/>.

/// </summary>

/// <param name="censusDivisionNumber"><see cref="CensusDivision.Number"/></param>

/// <returns>A <see cref="CensusDivision"/> object</returns>

public async Task<CensusDivision> GetByDivisionNumberAsync(string divisionNumber)

{

var filter = Builders<CensusDivision>.Filter.Eq(division => division.Number, divisionNumber);

return await \_context.CensusDivisions.Find(filter).FirstOrDefaultAsync();

}

public async Task<ICollection<State>> GetAllStatesByDivsionNumberAsync(string divisionNumber)

{

var division = await GetByDivisionNumberAsync(divisionNumber);

return division.States;

}

public async Task<ICollection<County>> GetAllCountiesByDivisionNumberAsync(string divisionNumber)

{

var states = await GetAllStatesByDivsionNumberAsync(divisionNumber);

var counties = new List<County>();

foreach (var state in states) counties.AddRange(state.Counties);

return counties;

}

public async Task<ICollection<HealthIndicator>> GetAllHealthIndicatorsByDivisionNumberAsync(string divisionNumber)

{

var counties = await GetAllCountiesByDivisionNumberAsync(divisionNumber);

var healthIndicators = new List<HealthIndicator>();

foreach (var county in counties) healthIndicators.AddRange(county.Indicators);

return healthIndicators;

}

public async Task<ICollection<CensusDivision>> GetAllDivisionsByRegionNumberAsync(string regionNumber)

{

var filter = Builders<CensusDivision>.Filter.Eq(division => division.CensusRegionNumber, regionNumber);

return await \_context.CensusDivisions.Find(filter).ToListAsync();

}

/// <summary>

/// Asynchronous update of a <see cref="CensusDivision"/> object.

/// </summary>

/// <param name="censusDivision"><see cref="CensusDivision"/></param>

/// <returns><see langword="true"/> if successful; <see langword="false"/> otherwise</returns>

public async Task<bool> UpdateAsync(CensusDivision censusDivision)

{

var updateResult = await \_context.CensusDivisions.ReplaceOneAsync(

filter: division => division.Number == censusDivision.Number,

replacement: censusDivision);

return updateResult.IsAcknowledged && updateResult.ModifiedCount > 0;

}

/// <summary>

/// Asynchronous hard deletion of a <see cref="CensusDivision"/> object with <see cref="CensusDivision.Number"/> matching <paramref name="divisionNumber"/>.

/// </summary>

/// <param name="divisionNumber"><see cref="CensusDivision.Number"/></param>

/// <returns><see langword="true"/> if successful; <see langword="false"/> otherwise</returns>

public async Task<bool> DeleteAsync(string divisionNumber)

{

var filter = Builders<CensusDivision>.Filter.Eq(censusDivision => censusDivision.Number, divisionNumber);

var deleteResult = await \_context.CensusDivisions.DeleteOneAsync(filter);

return deleteResult.IsAcknowledged && deleteResult.DeletedCount > 0;

}

}

}