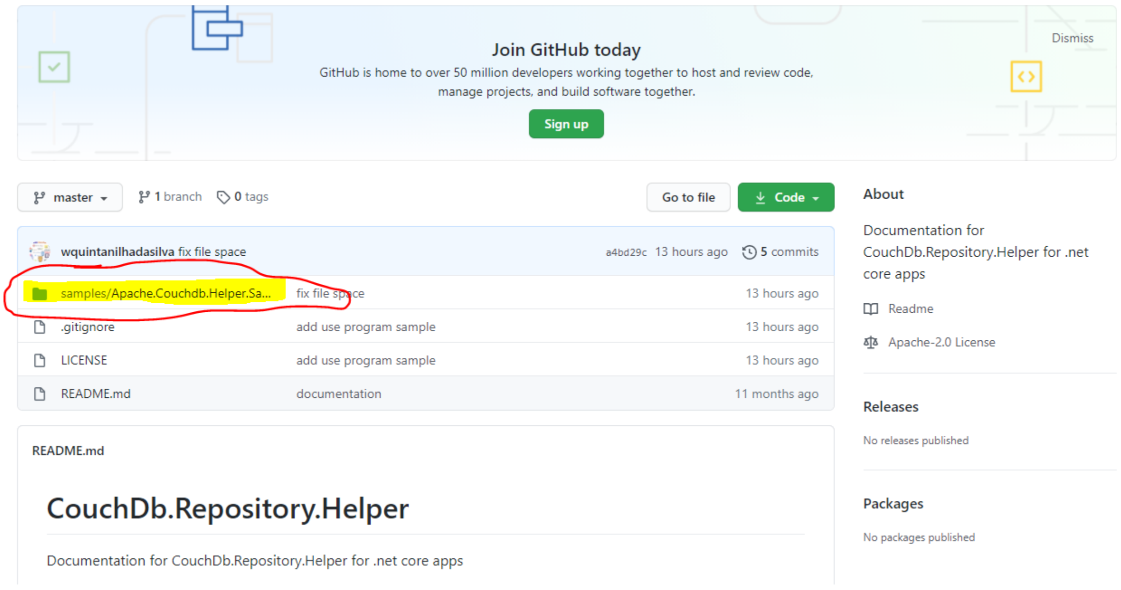
Open the sample solution shown in the image below and follow the explanations below.



Join GitHub today

GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together.

wquintanilhadasilva fix file space		a4bd29c 13 hours ago · 5 commits
samples/Apache.Couchdb.Helper.Sa...	fix file space	13 hours ago
.gitignore	add use program sample	13 hours ago
LICENSE	add use program sample	13 hours ago
README.md	documentation	11 months ago

About

Documentation for CouchDb.Repository.Helper for .net core apps

Readme

Apache-2.0 License

Releases

No releases published

Packages

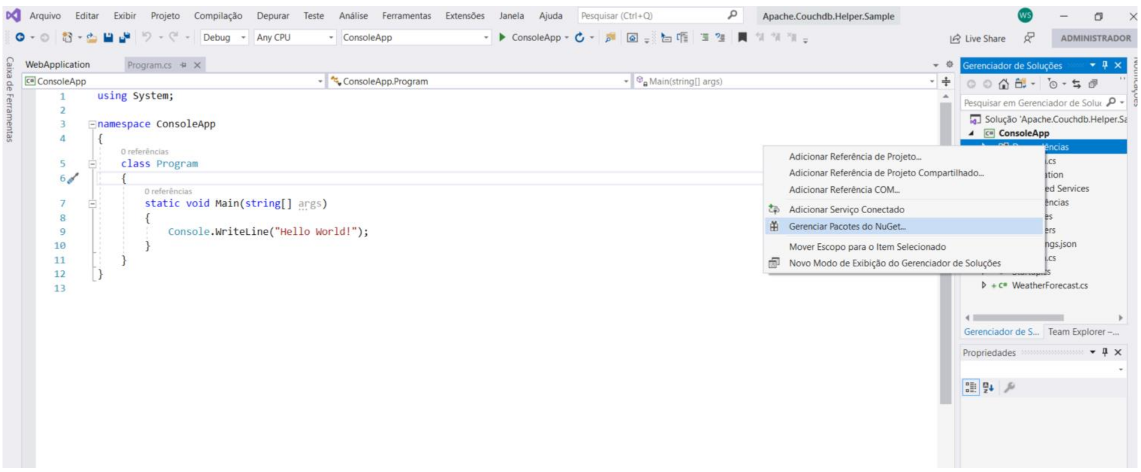
No packages published

README.md

CouchDb.Repository.Helper

Documentation for CouchDb.Repository.Helper for .net core apps

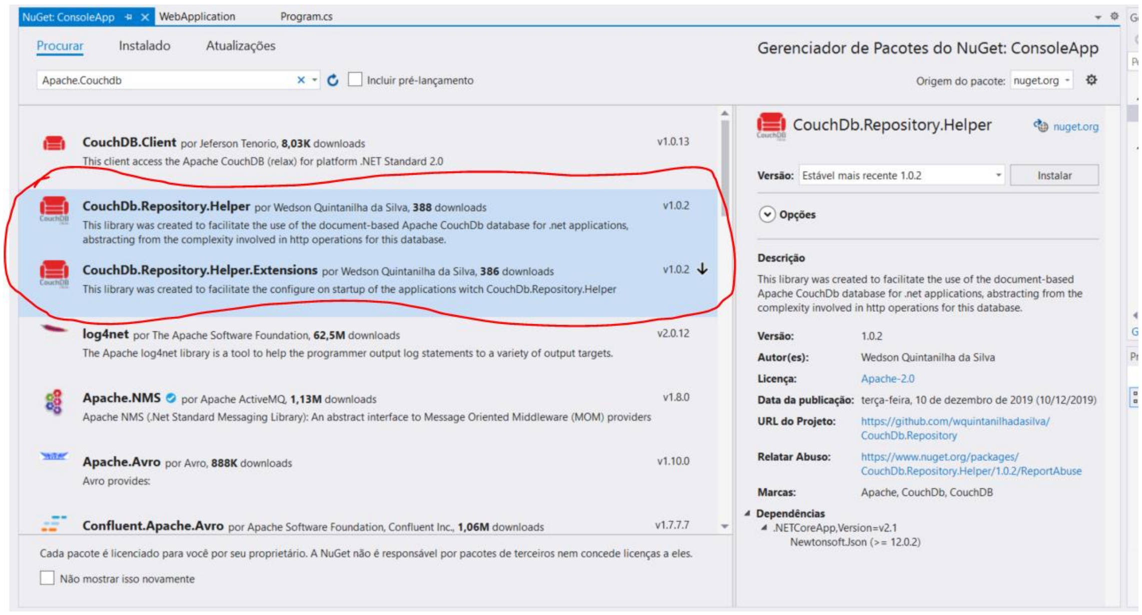
Add nuget packages bellow:



```
using System;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Arquivo
Editar
Exibir
Projeto
Compilação
Depurar
Teste
Análise
Ferramentas
Extensões
Janela
Ajuda
Pesquisar (Ctrl+Q)
Apache.Couchdb.Helper.Sample
Debug
Any CPU
ConsoleApp
Live Share
ADMINISTRADOR
WebApplication
Program.cs
ConsoleApp.Program
Main(string[] args)
Gerenciador de Soluções
Pesquisar em Gerenciador de Solu
Solução 'Apache.Couchdb.Helper.Sa
ConsoleApp
Adicionar Referência de Projeto...
Adicionar Referência de Projeto Compartilhado...
Adicionar Referência COM...
Adicionar Serviço Conectado
Gerenciar Pacotes do NuGet...
Mover Escopo para o Item Selecionado
Novo Modo de Exibição do Gerenciador de Soluções
WeatherForecast.cs
Gerenciador de S...
Team Explorer -...
Propriedades
Caixa de Ferramentas

Install packages:



Gerenciador de Pacotes do NuGet: ConsoleApp

Search: Apache.Couchdb — Incluir pré-lançamento — Origem do pacote: nuget.org

CouchDB.Client por Jeferson Tenorio, **8,03K** downloads — v1.0.13
This client access the Apache CouchDB (relax) for platform .NET Standard 2.0

CouchDb.Repository.Helper por Wedson Quintanilha da Silva, **388** downloads — v1.0.2
This library was created to facilitate the use of the document-based Apache CouchDb database for .net applications, abstracting from the complexity involved in http operations for this database.

CouchDb.Repository.Helper.Extensions por Wedson Quintanilha da Silva, **386** downloads — v1.0.2
This library was created to facilitate the configure on startup of the applications witch CouchDb.Repository.Helper

log4net por The Apache Software Foundation, **62,5M** downloads — v2.0.12
The Apache log4net library is a tool to help the programmer output log statements to a variety of output targets.

Apache.NMS por Apache ActiveMQ, **1,13M** downloads — v1.8.0
Apache NMS (.Net Standard Messaging Library): An abstract interface to Message Oriented Middleware (MOM) providers

Apache.Avro por Avro, **888K** downloads — v1.10.0
Avro provides:

Confluent.Apache.Avro por Apache Software Foundation, Confluent Inc., **1,06M** downloads — v1.7.7.7

Cada pacote é licenciado para você por seu proprietário. A NuGet não é responsável por pacotes de terceiros nem concede licenças a eles.

Não mostrar isso novamente

CouchDb.Repository.Helper — nuget.org

Versão: Estável mais recente 1.0.2 — Instalar

Opções

Descrição

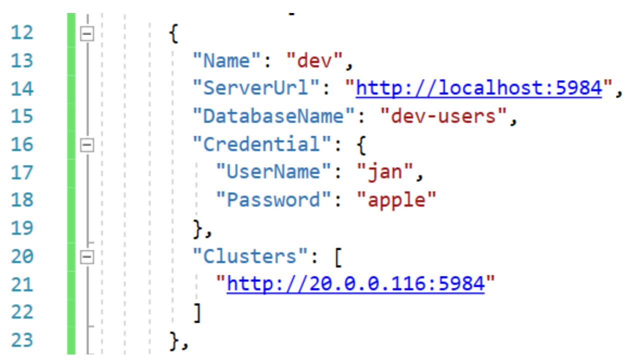
This library was created to facilitate the use of the document-based Apache CouchDb database for .net applications, abstracting from the complexity involved in http operations for this database.

Versão:	1.0.2
Autor(es):	Wedson Quintanilha da Silva
Licença:	Apache-2.0
Data da publicação:	terça-feira, 10 de dezembro de 2019 (10/12/2019)
URL do Projeto:	https://github.com/wquintanilhadasilva/CouchDb.Repository
Relatar Abuso:	https://www.nuget.org/packages/CouchDb.Repository.Helper/1.0.2/ReportAbuse
Marcas:	Apache, CouchDb, CouchDB

Dependências

.NETCoreApp,Version=v2.1
Newtonsoft.Json (>= 12.0.2)

Add config sections on appsettings.json:

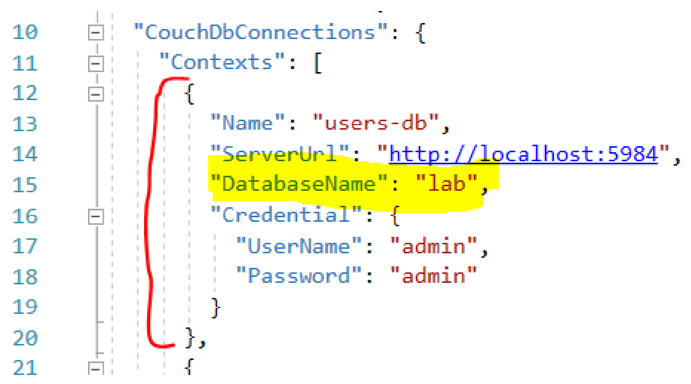


```
    {
      "Name": "dev",
      "ServerUrl": "http://localhost:5984",
      "DatabaseName": "dev-users",
      "Credential": {
        "UserName": "jan",
        "Password": "apple"
      },
      "Clusters": [
        "http://20.0.0.116:5984"
      ]
    },
```

"CouchDbConnections" – Config section name. Will be referenced in startup code. Pode ser qualquer nome.

This section must contain one, only one, element called "Contexts": []. This will be an array of configurations from one or more Apache CouchdDb databases that the application will interact with.

Each context must be defined according to the following structure:

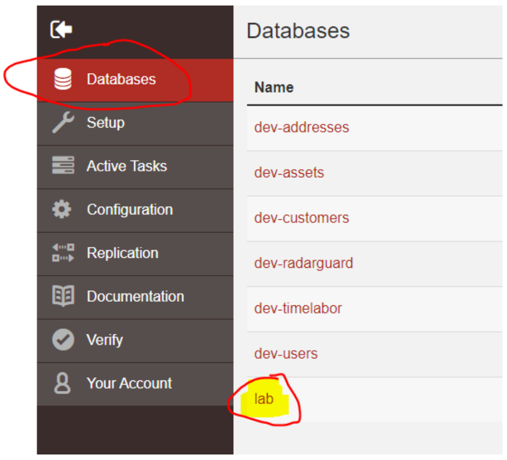


```
  "CouchDbConnections": {
    "Contexts": [
      {
        "Name": "users-db",
        "ServerUrl": "http://localhost:5984",
        "DatabaseName": "lab",
        "Credential": {
          "UserName": "admin",
          "Password": "admin"
        }
      },
      {
```

"Name": Name of the context that will be used by the application to access the database

"ServerUrl": Server address, it can be a ***dns*** or ***ip*** address with its respective port

"DatabaseName": Database name used by the context:



Databases

Name
dev-addresses
dev-assets
dev-customers
dev-radarguard
dev-timelabor
dev-users
lab

"Credential": Opcional. Adicione esta configuração ao arquivo caso o seu banco de dados tenha as configurações de segurança ativadas, ou seja, caso tenha usuário e senha definido para o banco de dados.

Optional. Add this setting to the file if your database has security settings enabled, that is, if you have a username and password set for the database

“Clusters”: Se tiver uma configuração de cluster do seu Apache Couchdb, use esta configuração para informar ao framework que há clusters que podem ser acionados caso o servidor principal deixe de responder. As configurações dos clusters devem respeitar o que está definido na documentação do próprio Apache CouchDb. Informe o ip ou o endereço dns de um ou mais servidores clusters, separados por vírgula. O acesso ao banco de dados será usando as mesmas definições de usuário, senha e nome do banco de dados definido para o cluster principal.

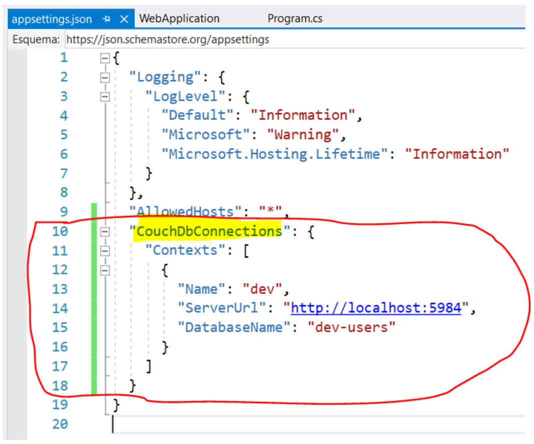
If you have a cluster configuration for your Apache Couchdb, use this configuration to inform the framework that there are clusters that can be triggered if the main server stops responding. The configurations of the clusters must respect what is defined in the Apache CouchDb documentation itself. Enter the ip or dns address of one or more cluster servers, separated by commas. Access to the database will be using the same user name, password and database name defined for the main cluster.

Exemplo (Sample):

```
"Clusters": [
  "http://100.1.1.1:5461",
  "http://localhost:8081"
]
```

Uma configuração simples sem cluster e sem usuário e senha pode ser observada no exemplo abaixo:

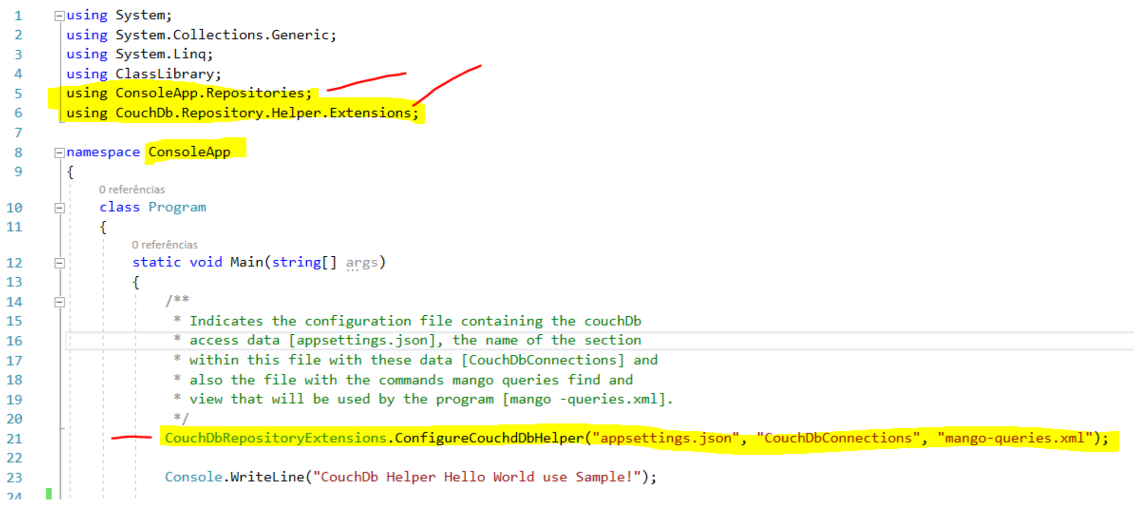
A simple configuration without cluster and without user and password can be seen in the example below:



```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "CouchDbConnections": {
    "Contexts": [
      {
        "Name": "dev",
        "ServerUrl": "http://localhost:5984",
        "DatabaseName": "dev-users"
      }
    ]
  }
}
```

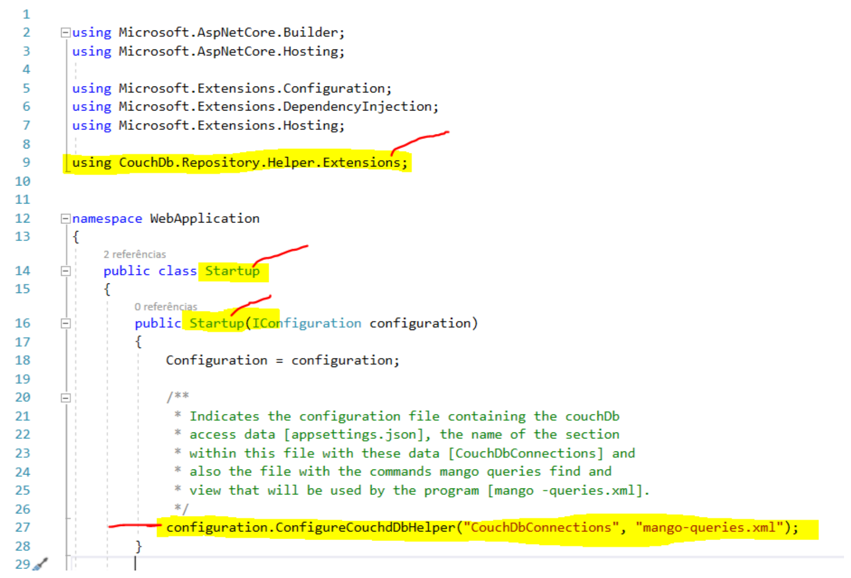
Configure the framework indicating the configuration file, the section within it and the main file that will contain the "find" commands with the syntax mango querie.

ConsoleApplication (startup program – Program.cs class):



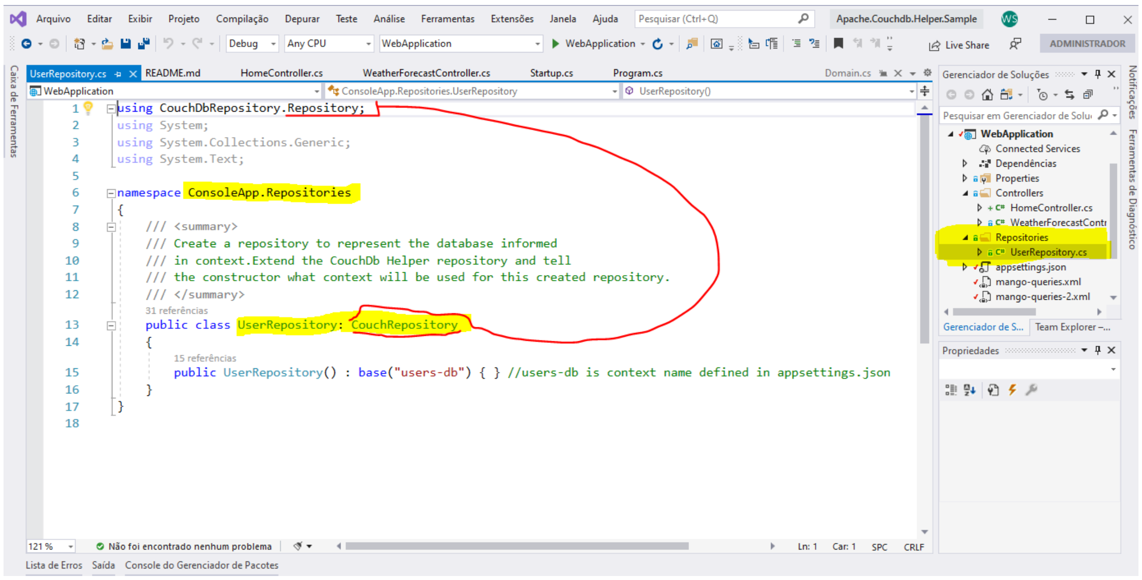
```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using ClassLibrary;
using ConsoleApp.Repositories;
using CouchDb.Repository.Helper.Extensions;

namespace ConsoleApp
{
    0 referências
    class Program
    {
        0 referências
        static void Main(string[] args)
        {
            /**
             * Indicates the configuration file containing the couchDb
             * access data [appsettings.json], the name of the section
             * within this file with these data [CouchDbConnections] and
             * also the file with the commands mango queries find and
             * view that will be used by the program [mango -queries.xml].
             */
            CouchDbRepositoryExtensions.ConfigureCouchDbHelper("appsettings.json", "CouchDbConnections", "mango-queries.xml");

            Console.WriteLine("CouchDb Helper Hello World use Sample!");
```

WebApplication (Startup.cs class):



```csharp

using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

using CouchDb.Repository.Helper.Extensions;


namespace WebApplication
{
    2 referências
    public class Startup
    {
        0 referências
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;

            /**
             * Indicates the configuration file containing the couchDb
             * access data [appsettings.json], the name of the section
             * within this file with these data [CouchDbConnections] and
             * also the file with the commands mango queries find and
             * view that will be used by the program [mango -queries.xml].
             */
            configuration.ConfigureCouchDbHelper("CouchDbConnections", "mango-queries.xml");
        }
```

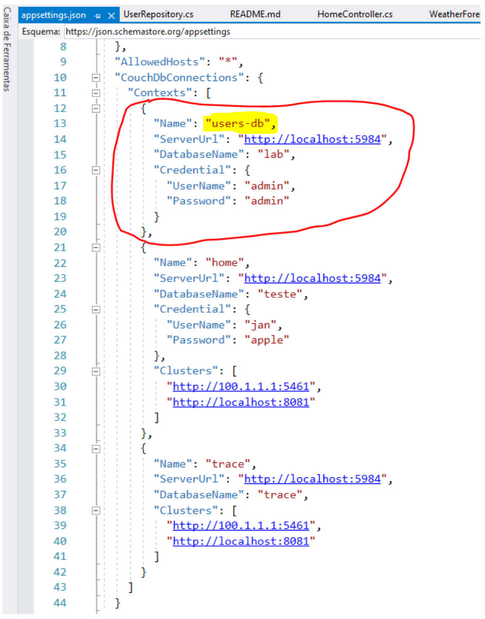
Add a class to represent the database in the application. In this case, I'm calling it a repository, you are free to call the name you want:



Arquivo Editar Exibir Projeto Compilação Depurar Teste Análise Ferramentas Extensões Janela Ajuda
Apache.Couchdb.Helper.Sample
Debug
Any CPU
WebApplication
UserRepository.cs README.md HomeController.cs WeatherForecastController.cs Startup.cs Program.cs Domain.cs
ConsoleApp.Repositories.UserRepository
UserRepository()
Gerenciador de Soluções
WebApplication
Connected Services
Dependências
Properties
Controllers
HomeController.cs
WeatherForecastController.cs
Repositories
UserRepository.cs
appsettings.json
mango-queries.xml
mango-queries-2.xml
Propriedades
Não foi encontrado nenhum problema
Lista de Erros Saída Console do Gerenciador de Pacotes

```csharp
using CouchDbRepository.Repository;
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApp.Repositories
{
    /// <summary>
    /// Create a repository to represent the database informed
    /// in context.Extend the CouchDb Helper repository and tell
    /// the constructor what context will be used for this created repository.
    /// </summary>
    31 referências
    public class UserRepository: CouchRepository
    {
        15 referências
        public UserRepository() : base("users-db") { } //users-db is context name defined in appsettings.json
    }
}
```

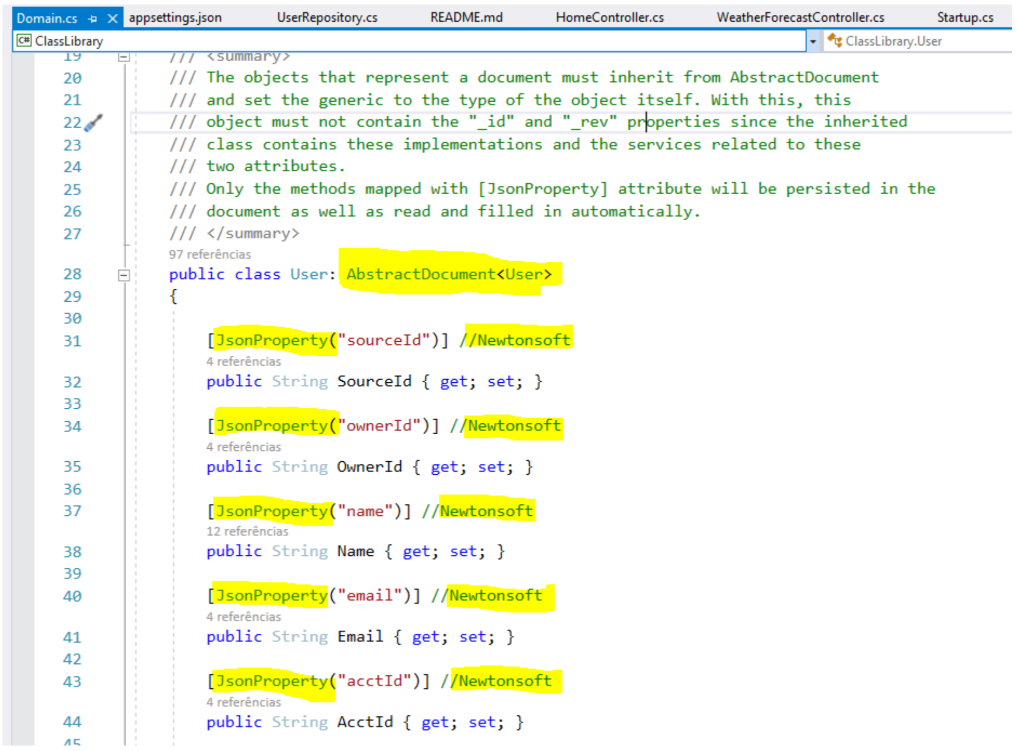
This class needs to extend the CouchDbRepository class. In the constructor, the name of the context where the database will be used must be informed, defined in the file appsettings.json.



```json
  },
  "AllowedHosts": "*",
  "CouchDbConnections": {
    "Contexts": [
      {
        "Name": "users-db",
        "ServerUrl": "http://localhost:5984",
        "DatabaseName": "lab",
        "Credential": {
          "UserName": "admin",
          "Password": "admin"
        }
      },
      {
        "Name": "home",
        "ServerUrl": "http://localhost:5984",
        "DatabaseName": "teste",
        "Credential": {
          "UserName": "jan",
          "Password": "apple"
        },
        "Clusters": [
          "http://100.1.1.1:5461",
          "http://localhost:8081"
        ]
      },
      {
        "Name": "trace",
        "ServerUrl": "http://localhost:5984",
        "DatabaseName": "trace",
        "Clusters": [
          "http://100.1.1.1:5461",
          "http://localhost:8081"
        ]
      }
    ]
  }
```

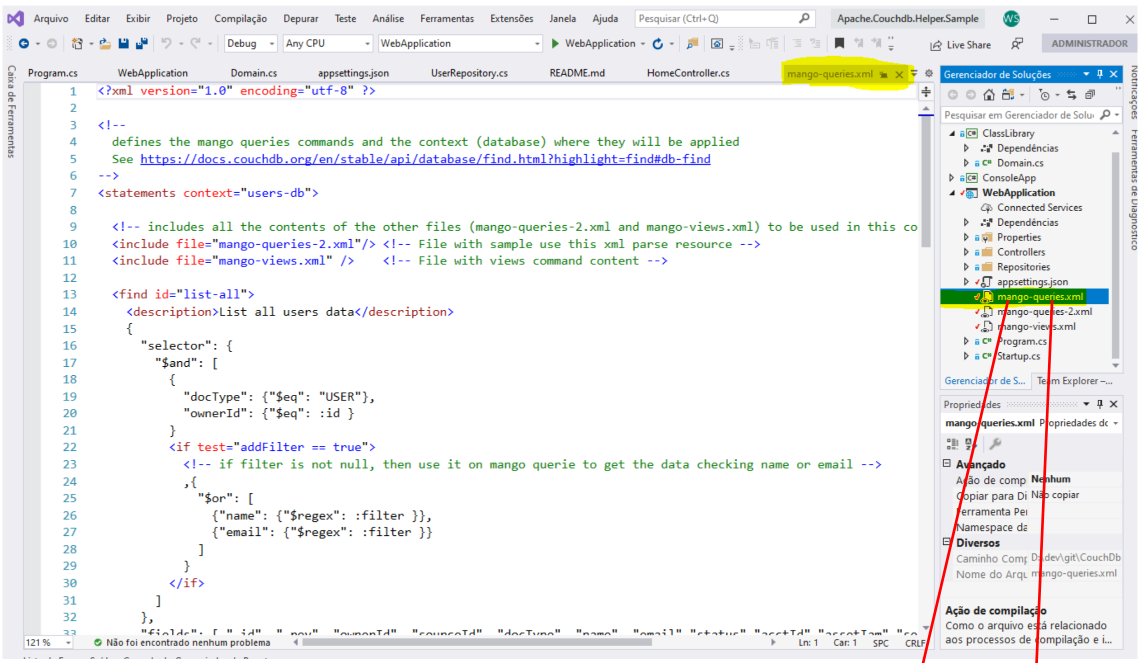
Create domain classes that will represent the documents in the database. This class must extend the "AbstractDocument" class.

The attributes that will be serialized in the database document must be noted with "JsonProperty". FOR THE sake of INTEGRITY AND RISK, I emphasize that ALL ATTRIBUTES IN THE DATABASE MUST HAVE A PROPERTY IN THE OBJECT, otherwise an exception will be thrown.



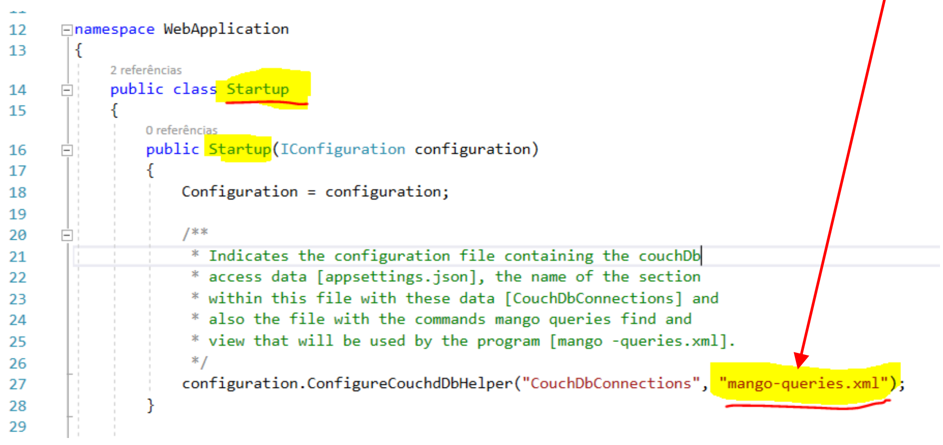
```csharp
    /// <summary>
    /// The objects that represent a document must inherit from AbstractDocument
    /// and set the generic to the type of the object itself. With this, this
    /// object must not contain the "_id" and "_rev" properties since the inherited
    /// class contains these implementations and the services related to these
    /// two attributes.
    /// Only the methods mapped with [JsonProperty] attribute will be persisted in the
    /// document as well as read and filled in automatically.
    /// </summary>
    97 referências
    public class User: AbstractDocument<User>
    {

        [JsonProperty("sourceId")] //Newtonsoft
        4 referências
        public String SourceId { get; set; }

        [JsonProperty("ownerId")] //Newtonsoft
        4 referências
        public String OwnerId { get; set; }

        [JsonProperty("name")] //Newtonsoft
        12 referências
        public String Name { get; set; }

        [JsonProperty("email")] //Newtonsoft
        4 referências
        public String Email { get; set; }

        [JsonProperty("acctId")] //Newtonsoft
        4 referências
        public String AcctId { get; set; }
```

Create the file that will contain the find commands (mango queries) or their template.

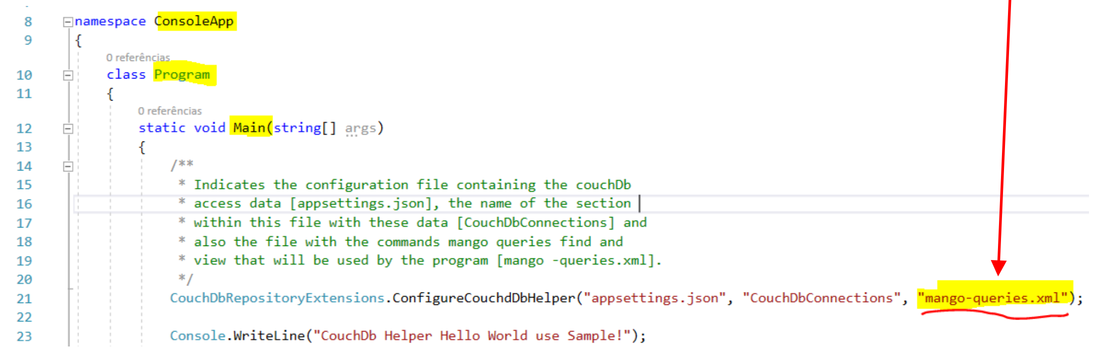


```xml
<?xml version="1.0" encoding="utf-8" ?>

<!--
  defines the mango queries commands and the context (database) where they will be applied
  See https://docs.couchdb.org/en/stable/api/database/find.html?highlight=find#db-find
-->
<statements context="users-db">

  <!-- includes all the contents of the other files (mango-queries-2.xml and mango-views.xml) to be used in this co
  <include file="mango-queries-2.xml"/> <!-- File with sample use this xml parse resource -->
  <include file="mango-views.xml" />    <!-- File with views command content -->

  <find id="list-all">
    <description>List all users data</description>
    {
      "selector": {
        "$and": [
          {
            "docType": {"$eq": "USER"},
            "ownerId": {"$eq": :id }
          }
          <if test="addFilter == true">
            <!-- if filter is not null, then use it on mango querie to get the data checking name or email -->
            ,{
              "$or": [
                {"name": {"$regex": :filter }},
                {"email": {"$regex": :filter }}
              ]
            }
          </if>
        ]
      },
      "fields": [ "_id", "_rev", "ownerId", "sourceId", "docType", "name", "email" "status" "acctId" "assetTam" "co
```

The name of this file must be the same as indicated when initializing the framework configuration there in the initial method. Examples:

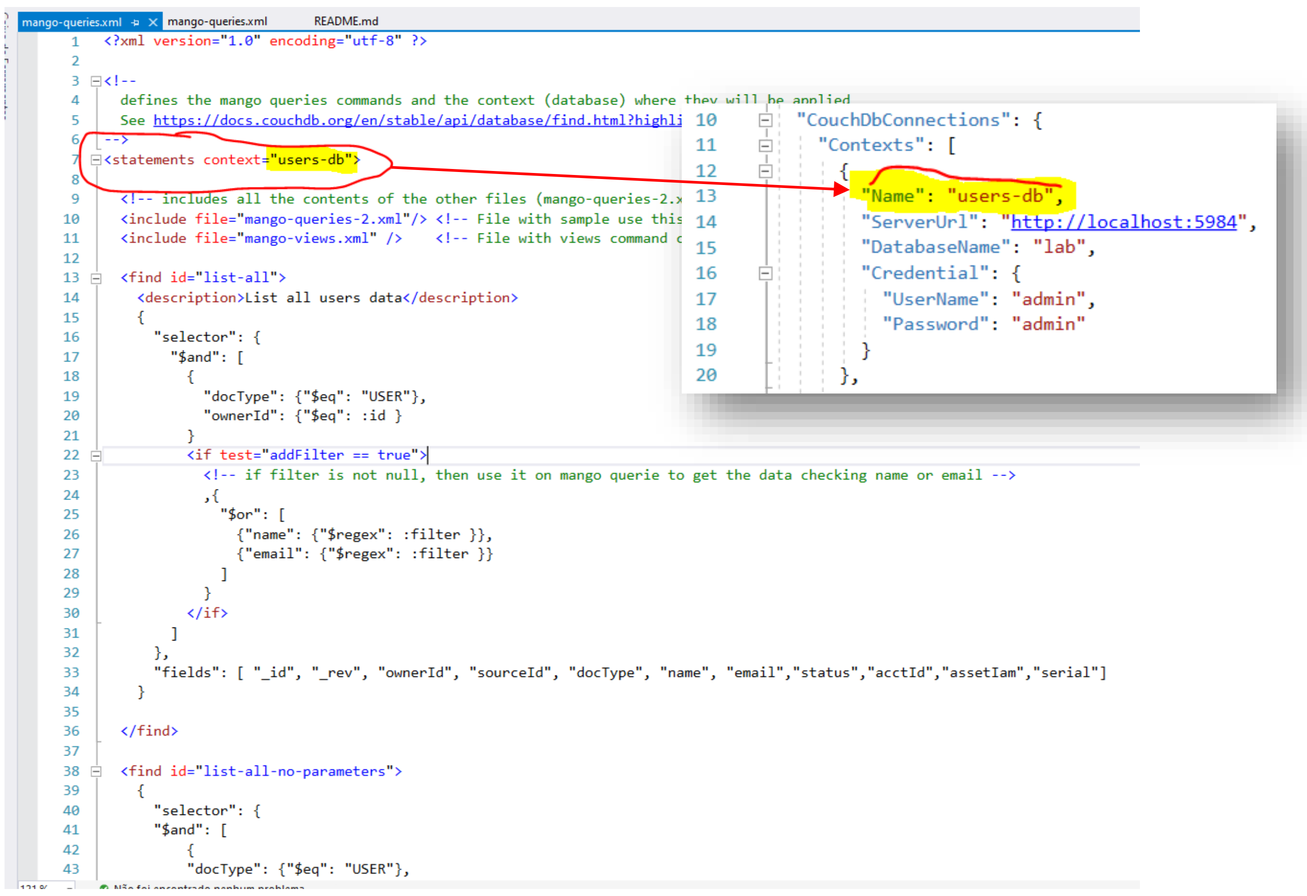


```csharp
namespace WebApplication
{
    2 referências
    public class Startup
    {
        0 referências
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;

            /**
             * Indicates the configuration file containing the couchDb
             * access data [appsettings.json], the name of the section
             * within this file with these data [CouchDbConnections] and
             * also the file with the commands mango queries find and
             * view that will be used by the program [mango -queries.xml].
             */
            configuration.ConfigureCouchdDbHelper("CouchDbConnections", "mango-queries.xml");
        }
```



```csharp
namespace ConsoleApp
{
    0 referências
    class Program
    {
        0 referências
        static void Main(string[] args)
        {
            /**
             * Indicates the configuration file containing the couchDb
             * access data [appsettings.json], the name of the section
             * within this file with these data [CouchDbConnections] and
             * also the file with the commands mango queries find and
             * view that will be used by the program [mango -queries.xml].
             */
            CouchDbRepositoryExtensions.ConfigureCouchdDbHelper("appsettings.json", "CouchDbConnections", "mango-queries.xml");

            Console.WriteLine("CouchDb Helper Hello World use Sample!");
```

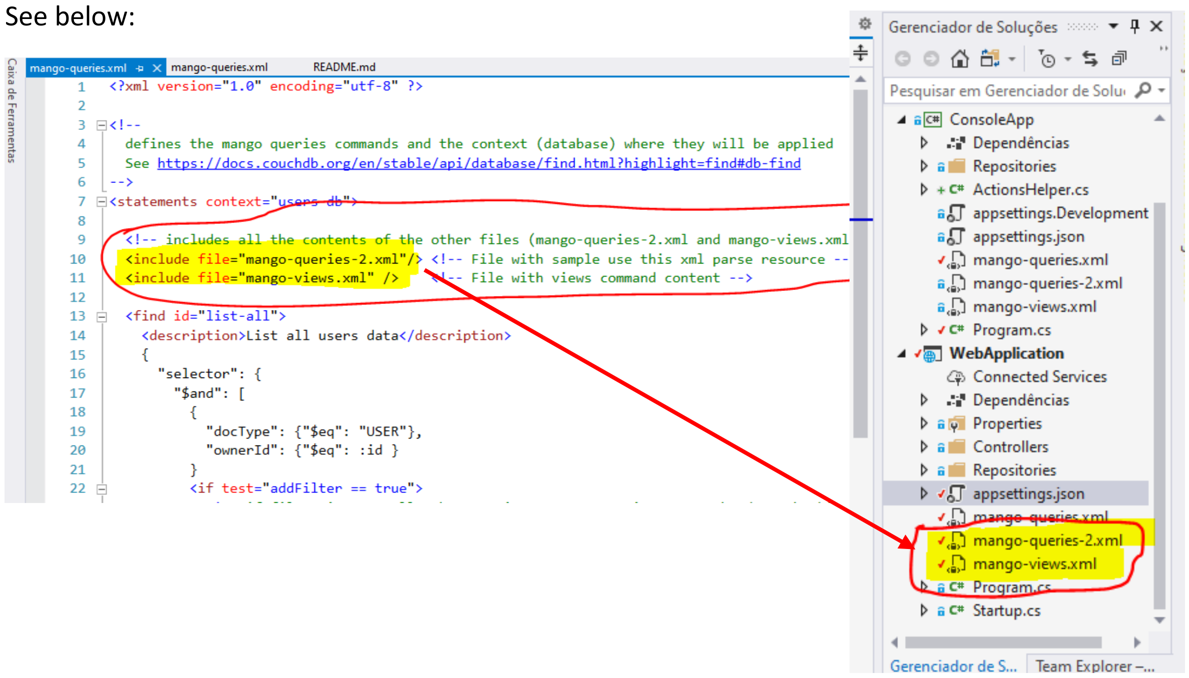
Define the resources available for the application, indicating the context to which it should be applied. Each file only allows a single "*statements*" block. Use any of the contexts mapped in the *appsettings.json* configuration file:



```
<?xml version="1.0" encoding="utf-8" ?>

<!--
  defines the mango queries commands and the context (database) where they will be applied
  See https://docs.couchdb.org/en/stable/api/database/find.html?highli
-->
<statements context="users-db">

  <!-- includes all the contents of the other files (mango-queries-2.x
  <include file="mango-queries-2.xml"/> <!-- File with sample use this
  <include file="mango-views.xml" />    <!-- File with views command c

  <find id="list-all">
    <description>List all users data</description>
    {
      "selector": {
        "$and": [
          {
            "docType": {"$eq": "USER"},
            "ownerId": {"$eq": :id }
          }
          <if test="addFilter == true">
            <!-- if filter is not null, then use it on mango querie to get the data checking name or email -->
            ,{
              "$or": [
                {"name": {"$regex": :filter }},
                {"email": {"$regex": :filter }}
              ]
            }
          </if>
        ]
      },
      "fields": [ "_id", "_rev", "ownerId", "sourceId", "docType", "name", "email","status","acctId","assetIam","serial"]
    }

  </find>

  <find id="list-all-no-parameters">
    {
      "selector": {
      "$and": [
          {
          "docType": {"$eq": "USER"},
```

```
"CouchDbConnections": {
  "Contexts": [
    {
      "Name": "users-db",
      "ServerUrl": "http://localhost:5984",
      "DatabaseName": "lab",
      "Credential": {
        "UserName": "admin",
        "Password": "admin"
      }
    },
```

Organize your find commands into separate files. To do this, you can include these additional files in the main file (main file is the file that was mapped when the application was started). See below:



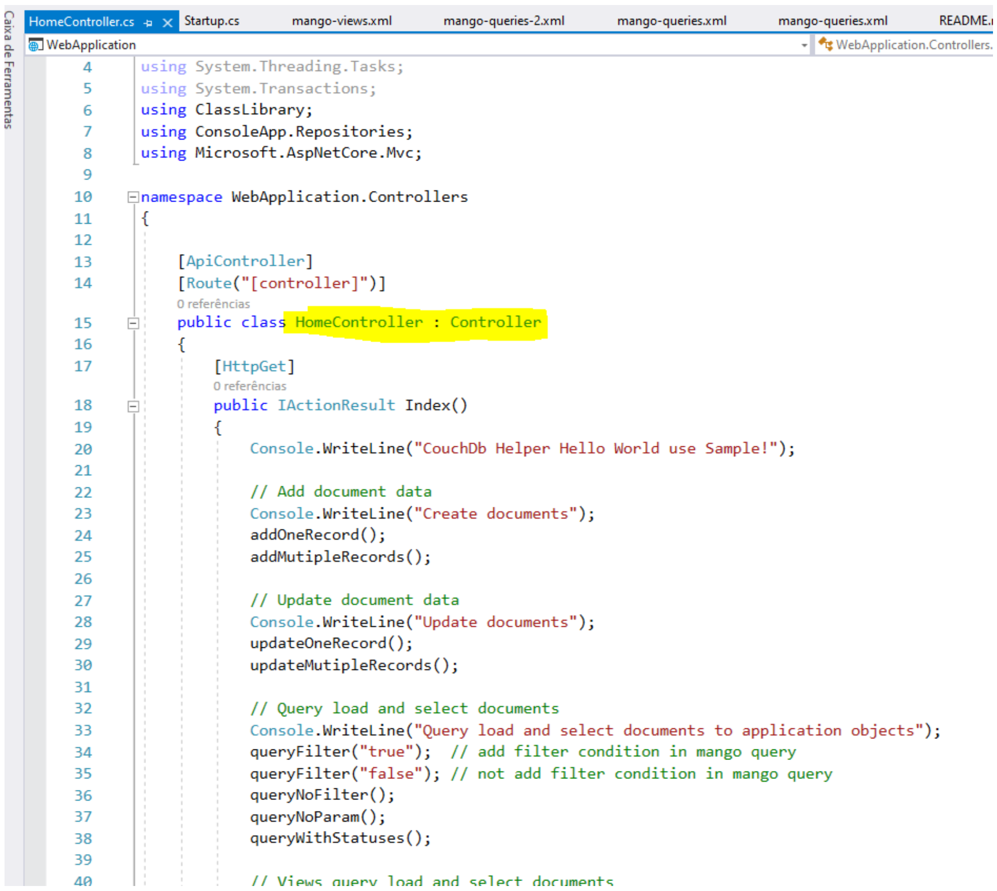
```
<?xml version="1.0" encoding="utf-8" ?>

<!--
  defines the mango queries commands and the context (database) where they will be applied
  See https://docs.couchdb.org/en/stable/api/database/find.html?highlight=find#db-find
-->
<statements context="users-db">

  <!-- includes all the contents of the other files (mango-queries-2.xml and mango-views.xml
  <include file="mango-queries-2.xml"/> <!-- File with sample use this xml parse resource --
  <include file="mango-views.xml" />    <!-- File with views command content -->

  <find id="list-all">
    <description>List all users data</description>
    {
      "selector": {
        "$and": [
          {
            "docType": {"$eq": "USER"},
            "ownerId": {"$eq": :id }
          }
          <if test="addFilter == true">
```

In this file, use the resources available in the couchdb documentation to create your queries: https://docs.couchdb.org/en/stable/api/database/find.html

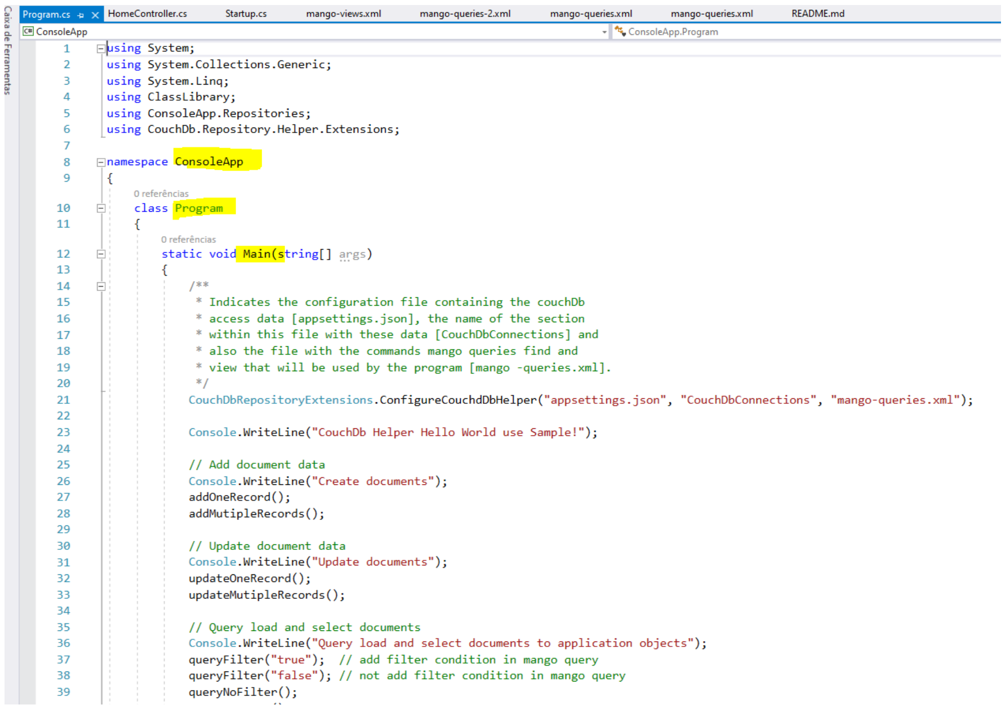
In the sample solution ...

The HomeController class of the WebApplication project contains examples of use for CRUD operations in an API.

The Program class of the ConsoleApplication project contains examples of use for CRUD operations in a console application.



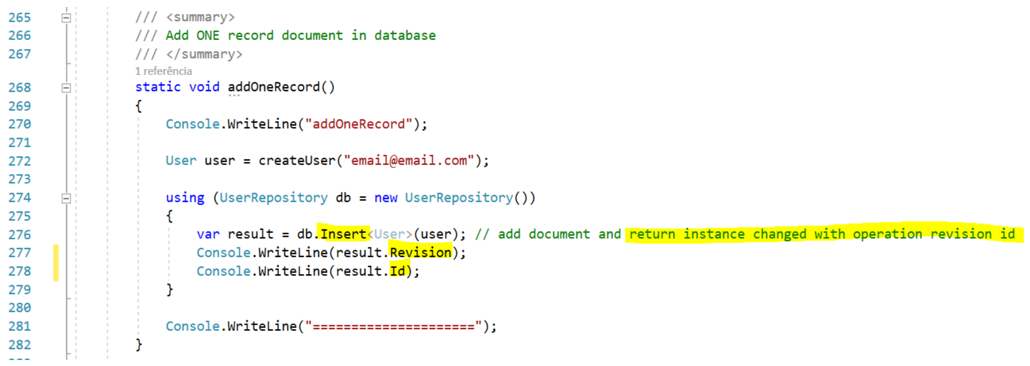
```
using System.Threading.Tasks;
using System.Transactions;
using ClassLibrary;
using ConsoleApp.Repositories;
using Microsoft.AspNetCore.Mvc;

namespace WebApplication.Controllers
{

    [ApiController]
    [Route("[controller]")]
    0 referências
    public class HomeController : Controller
    {
        [HttpGet]
        0 referências
        public IActionResult Index()
        {
            Console.WriteLine("CouchDb Helper Hello World use Sample!");

            // Add document data
            Console.WriteLine("Create documents");
            addOneRecord();
            addMutipleRecords();

            // Update document data
            Console.WriteLine("Update documents");
            updateOneRecord();
            updateMutipleRecords();

            // Query load and select documents
            Console.WriteLine("Query load and select documents to application objects");
            queryFilter("true");  // add filter condition in mango query
            queryFilter("false"); // not add filter condition in mango query
            queryNoFilter();
            queryNoParam();
            queryWithStatuses();

            // Views query load and select documents
```



```
using System;
using System.Collections.Generic;
using System.Linq;
using ClassLibrary;
using ConsoleApp.Repositories;
using CouchDb.Repository.Helper.Extensions;

namespace ConsoleApp
{
    0 referências
    class Program
    {
        0 referências
        static void Main(string[] args)
        {
            /**
             * Indicates the configuration file containing the couchDb
             * access data [appsettings.json], the name of the section
             * within this file with these data [CouchDbConnections] and
             * also the file with the commands mango queries find and
             * view that will be used by the program [mango -queries.xml].
             */
            CouchDbRepositoryExtensions.ConfigureCouchdDbHelper("appsettings.json", "CouchDbConnections", "mango-queries.xml");

            Console.WriteLine("CouchDb Helper Hello World use Sample!");

            // Add document data
            Console.WriteLine("Create documents");
            addOneRecord();
            addMutipleRecords();

            // Update document data
            Console.WriteLine("Update documents");
            updateOneRecord();
            updateMutipleRecords();

            // Query load and select documents
            Console.WriteLine("Query load and select documents to application objects");
            queryFilter("true");  // add filter condition in mango query
            queryFilter("false"); // not add filter condition in mango query
            queryNoFilter();
```

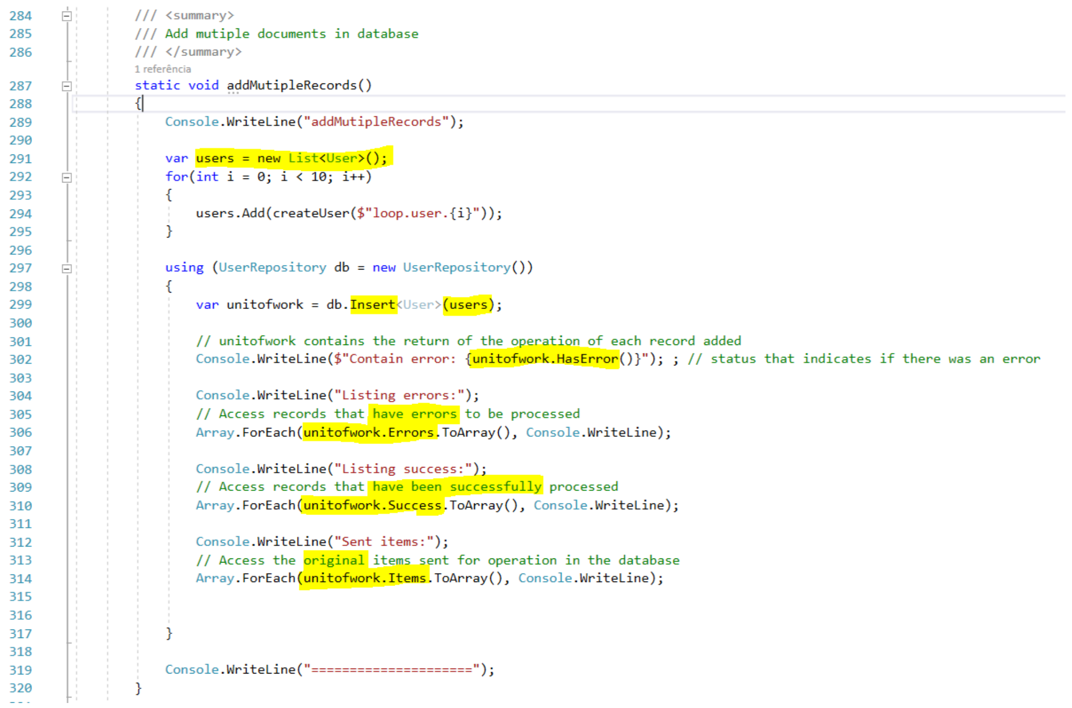
Manager documents:

:: **Insert** ONE document – Return ID and REVISION data



```
/// <summary>
/// Add ONE record document in database
/// </summary>
1 referência
static void addOneRecord()
{
    Console.WriteLine("addOneRecord");

    User user = createUser("email@email.com");

    using (UserRepository db = new UserRepository())
    {
        var result = db.Insert<User>(user); // add document and return instance changed with operation revision id
        Console.WriteLine(result.Revision);
        Console.WriteLine(result.Id);
    }

    Console.WriteLine("====================");
}
```

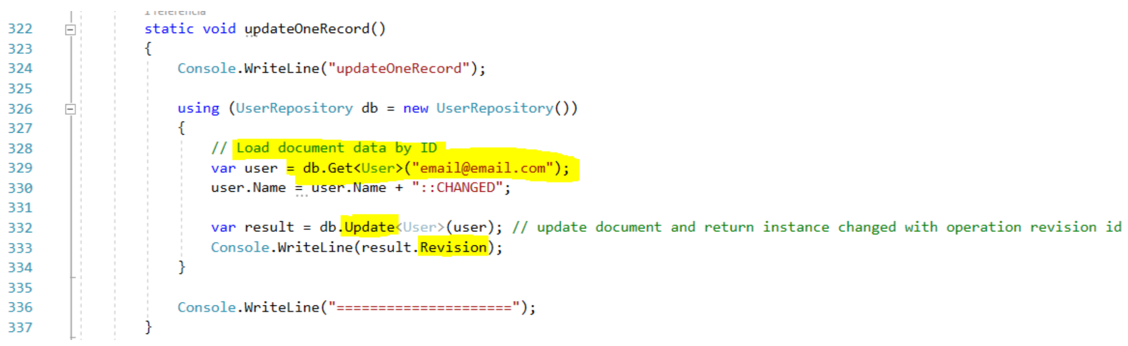
:: **Insert** MUTIPLE documents – Returns the unit of work with the individual results



```
/// <summary>
/// Add mutiple documents in database
/// </summary>
1 referência
static void addMutipleRecords()
{
    Console.WriteLine("addMutipleRecords");

    var users = new List<User>();
    for(int i = 0; i < 10; i++)
    {
        users.Add(createUser($"loop.user.{i}"));
    }

    using (UserRepository db = new UserRepository())
    {
        var unitofwork = db.Insert<User>(users);

        // unitofwork contains the return of the operation of each record added
        Console.WriteLine($"Contain error: {unitofwork.HasError()}"); ; // status that indicates if there was an error

        Console.WriteLine("Listing errors:");
        // Access records that have errors to be processed
        Array.ForEach(unitofwork.Errors.ToArray(), Console.WriteLine);

        Console.WriteLine("Listing success:");
        // Access records that have been successfully processed
        Array.ForEach(unitofwork.Success.ToArray(), Console.WriteLine);

        Console.WriteLine("Sent items:");
        // Access the original items sent for operation in the database
        Array.ForEach(unitofwork.Items.ToArray(), Console.WriteLine);


    }

    Console.WriteLine("====================");
}
```

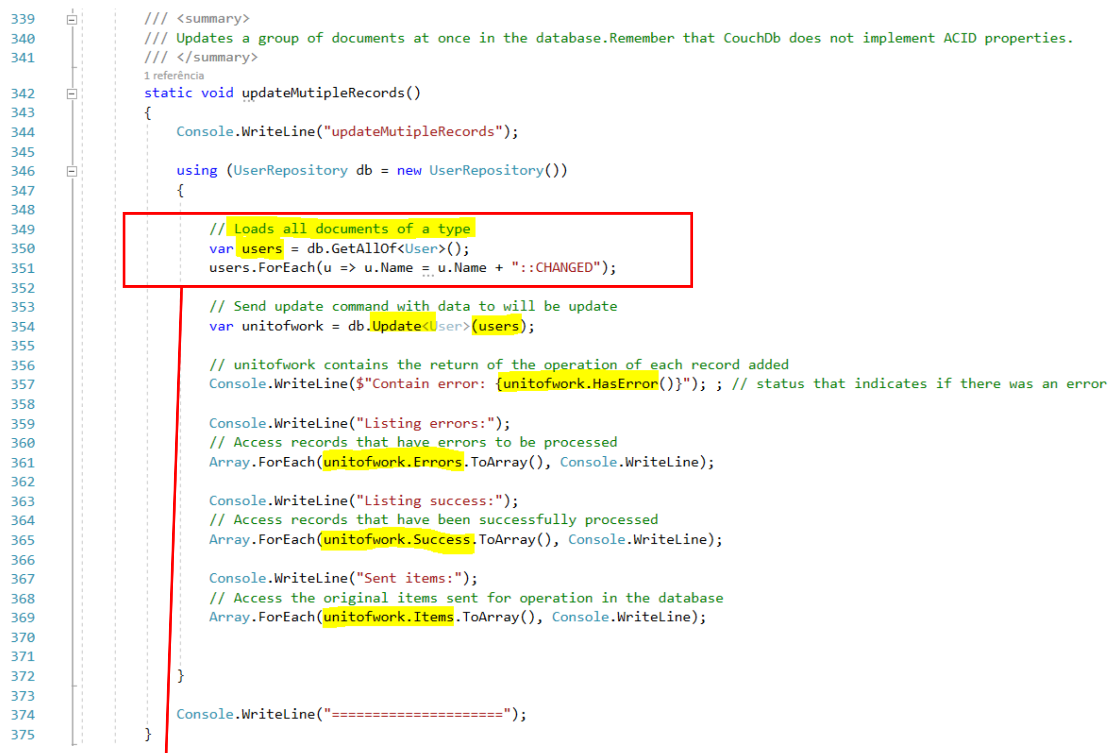
:: <u>**Update**</u> ONE document – Return document with changed REVISION data

Current id and revision is required to make changes to documents. Therefore, first retrieve the document and modify this recovered document.



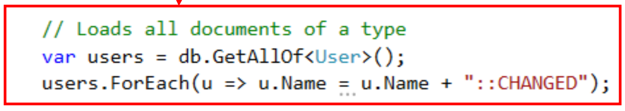
```
static void updateOneRecord()
{
    Console.WriteLine("updateOneRecord");

    using (UserRepository db = new UserRepository())
    {
        // Load document data by ID
        var user = db.Get<User>("email@email.com");
        user.Name = user.Name + "::CHANGED";

        var result = db.Update<User>(user); // update document and return instance changed with operation revision id
        Console.WriteLine(result.Revision);
    }

    Console.WriteLine("====================");
}
```

:: **Update** MUTIPLE documents – Returns the unit of work with the individual results



```csharp
/// <summary>
/// Updates a group of documents at once in the database.Remember that CouchDb does not implement ACID properties.
/// </summary>
1 referência
static void updateMutipleRecords()
{
    Console.WriteLine("updateMutipleRecords");

    using (UserRepository db = new UserRepository())
    {

        // Loads all documents of a type
        var users = db.GetAllOf<User>();
        users.ForEach(u => u.Name = u.Name + "::CHANGED");

        // Send update command with data to will be update
        var unitofwork = db.Update<User>(users);

        // unitofwork contains the return of the operation of each record added
        Console.WriteLine($"Contain error: {unitofwork.HasError()}"); ; // status that indicates if there was an error

        Console.WriteLine("Listing errors:");
        // Access records that have errors to be processed
        Array.ForEach(unitofwork.Errors.ToArray(), Console.WriteLine);

        Console.WriteLine("Listing success:");
        // Access records that have been successfully processed
        Array.ForEach(unitofwork.Success.ToArray(), Console.WriteLine);

        Console.WriteLine("Sent items:");
        // Access the original items sent for operation in the database
        Array.ForEach(unitofwork.Items.ToArray(), Console.WriteLine);


    }

    Console.WriteLine("====================");
}
```

The code below retrieves all documents of a type.

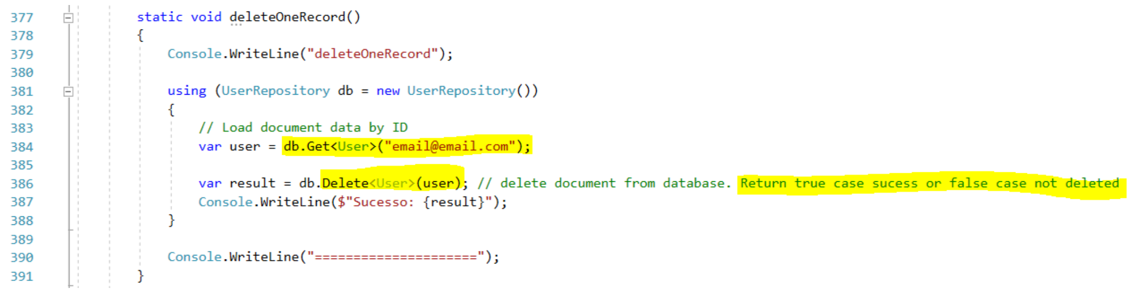


```csharp
// Loads all documents of a type
var users = db.GetAllOf<User>();
users.ForEach(u => u.Name = u.Name + "::CHANGED");
```

This is just an example, in your case, you will recover only the documents you want to delete. If you already have the documents updated, you do not need this step, just submit them, but be careful, you need to ensure that the revision is the most recent!

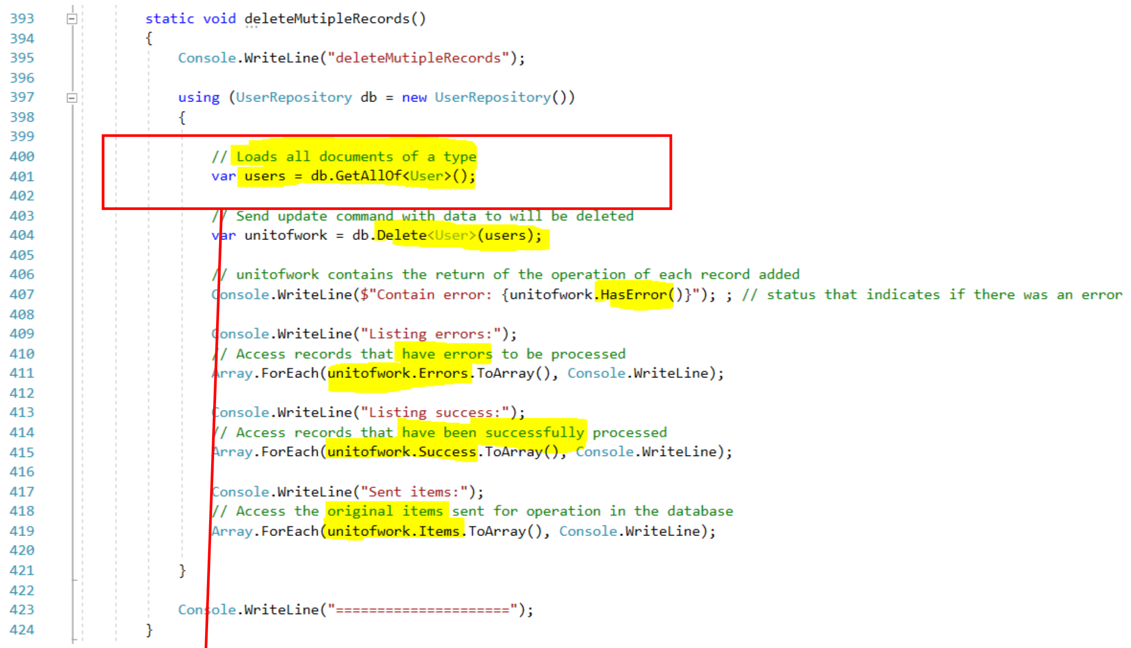
:: **Delete** ONE document – Return document with changed REVISION data

Current id and revision is required to delete documents. Therefore, first retrieve the document by key and modify this recovered document.



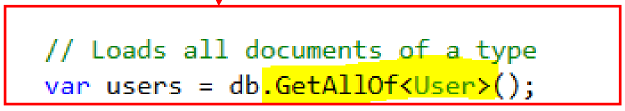
```csharp
static void deleteOneRecord()
{
    Console.WriteLine("deleteOneRecord");

    using (UserRepository db = new UserRepository())
    {
        // Load document data by ID
        var user = db.Get<User>("email@email.com");

        var result = db.Delete<User>(user); // delete document from database. Return true case sucess or false case not deleted
        Console.WriteLine($"Sucesso: {result}");
    }

    Console.WriteLine("====================");
}
```

:: **Delete** MUTIPLE documents – Returns the unit of work with the individual results



```
static void deleteMutipleRecords()
{
    Console.WriteLine("deleteMutipleRecords");

    using (UserRepository db = new UserRepository())
    {

        // Loads all documents of a type
        var users = db.GetAllOf<User>();

        // Send update command with data to will be deleted
        var unitofwork = db.Delete<User>(users);

        // unitofwork contains the return of the operation of each record added
        Console.WriteLine($"Contain error: {unitofwork.HasError()}"); ; // status that indicates if there was an error

        Console.WriteLine("Listing errors:");
        // Access records that have errors to be processed
        Array.ForEach(unitofwork.Errors.ToArray(), Console.WriteLine);

        Console.WriteLine("Listing success:");
        // Access records that have been successfully processed
        Array.ForEach(unitofwork.Success.ToArray(), Console.WriteLine);

        Console.WriteLine("Sent items:");
        // Access the original items sent for operation in the database
        Array.ForEach(unitofwork.Items.ToArray(), Console.WriteLine);

    }

    Console.WriteLine("====================");
}
```

The code below retrieves all documents of a type.

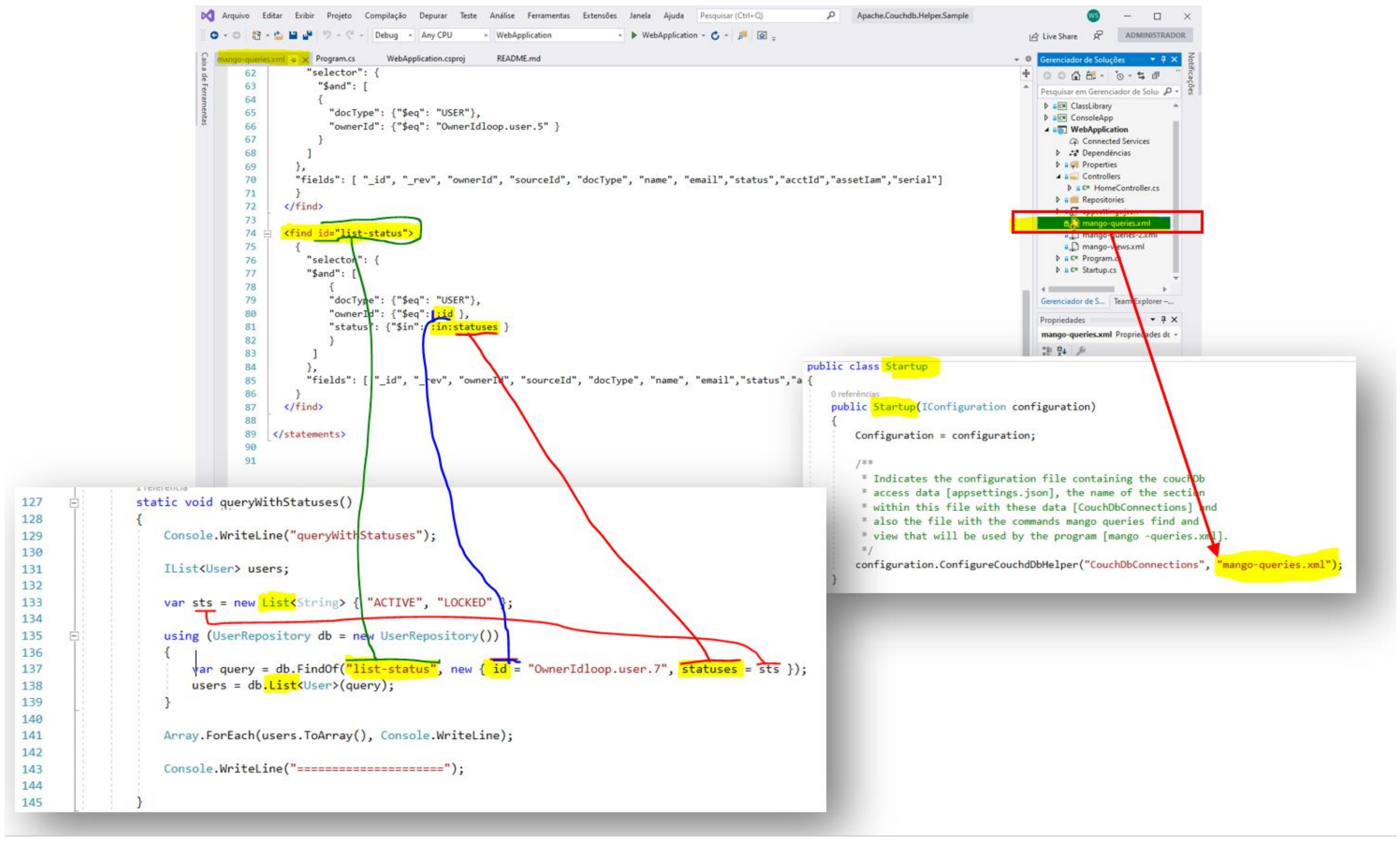


```
// Loads all documents of a type
var users = db.GetAllOf<User>();
```

This is just an example, in your case, you will recover only the documents you want to delete. If you already have the documents updated, you do not need this step, just submit them, but be careful, you need to ensure that the revision is the most recent!

:: **SELECT** documents by querie filter

Define and save your queries in the xml file (you can use one or more of a file according to your project organization) using the "find" commands with Mango-Queries syntax from couchdb itself. The main file must be informed in the initial configuration of the project and it must include the others if they exist.



```
    "selector": {
      "$and": [
      {
        "docType": {"$eq": "USER"},
        "ownerId": {"$eq": "OwnerIdloop.user.5" }
      }
     ]
    },
   "fields": [ "_id", "_rev", "ownerId", "sourceId", "docType", "name", "email","status","acctId","assetIam","serial"]
   }
  </find>

  <find id="list-status">
   {
    "selector": {
    "$and": [
        {
        "docType": {"$eq": "USER"},
        "ownerId": {"$eq": :id },
        "status": {"$in": :in:statuses }
        }
     ]
    },
    "fields": [ "_id", "_rev", "ownerId", "sourceId", "docType", "name", "email","status","a
   }
  </find>

</statements>
```

```
public class Startup
{
    0 referências
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;

        /**
         * Indicates the configuration file containing the couchDb
         * access data [appsettings.json], the name of the section
         * within this file with these data [CouchDbConnections] and
         * also the file with the commands mango queries find and
         * view that will be used by the program [mango -queries.xml].
         */
        configuration.ConfigureCouchdDbHelper("CouchDbConnections", "mango-queries.xml");
    }
```

```
static void queryWithStatuses()
{
    Console.WriteLine("queryWithStatuses");

    IList<User> users;

    var sts = new List<String> { "ACTIVE", "LOCKED" };

    using (UserRepository db = new UserRepository())
    {
        var query = db.FindOf("list-status", new { id = "OwnerIdloop.user.7", statuses = sts });
        users = db.List<User>(query);
    }

    Array.ForEach(users.ToArray(), Console.WriteLine);

    Console.WriteLine("====================");
}
```

The above codes were extracted from the sample projects published in this github. Download the solution and analyze the codes in detail for further understanding.