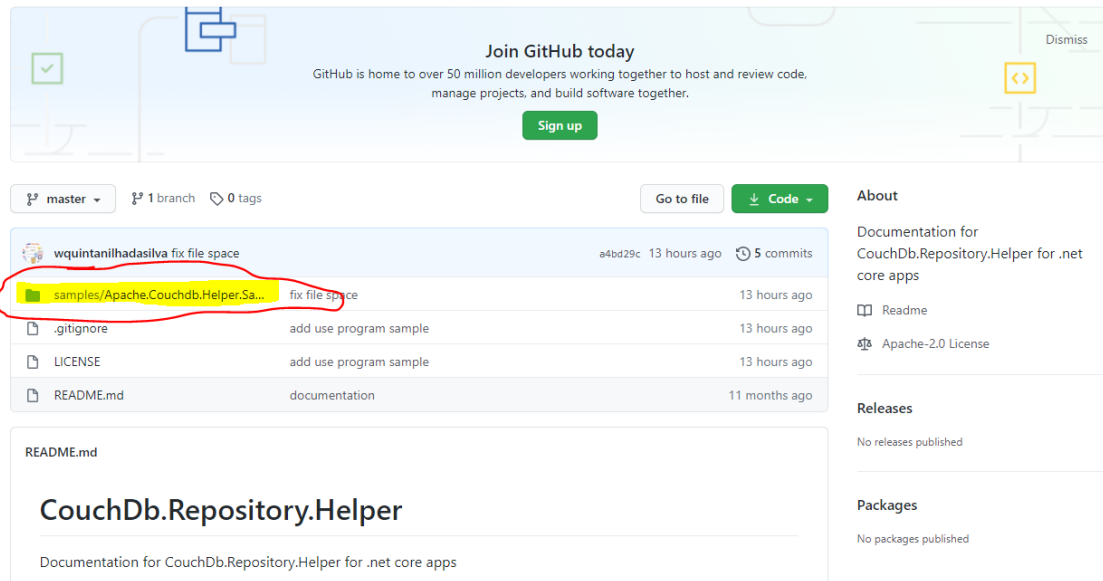
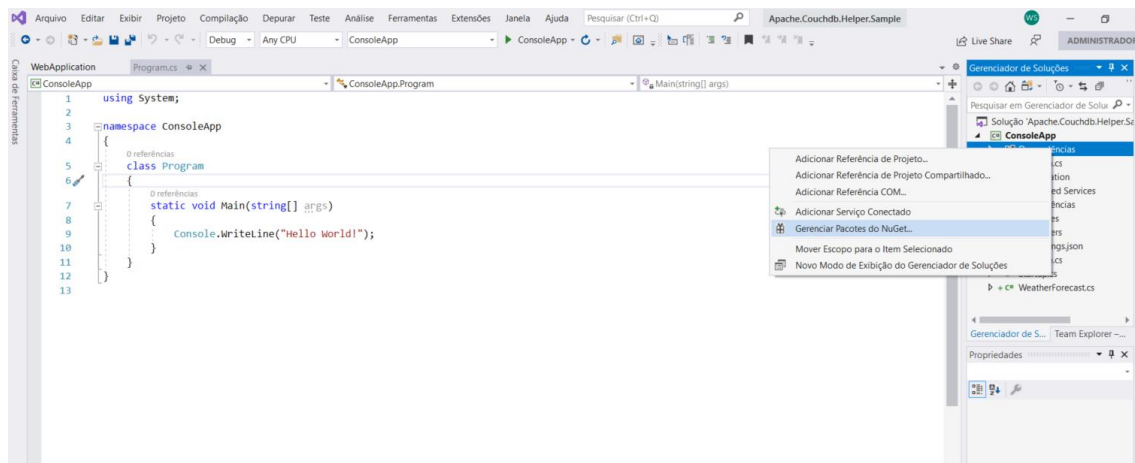


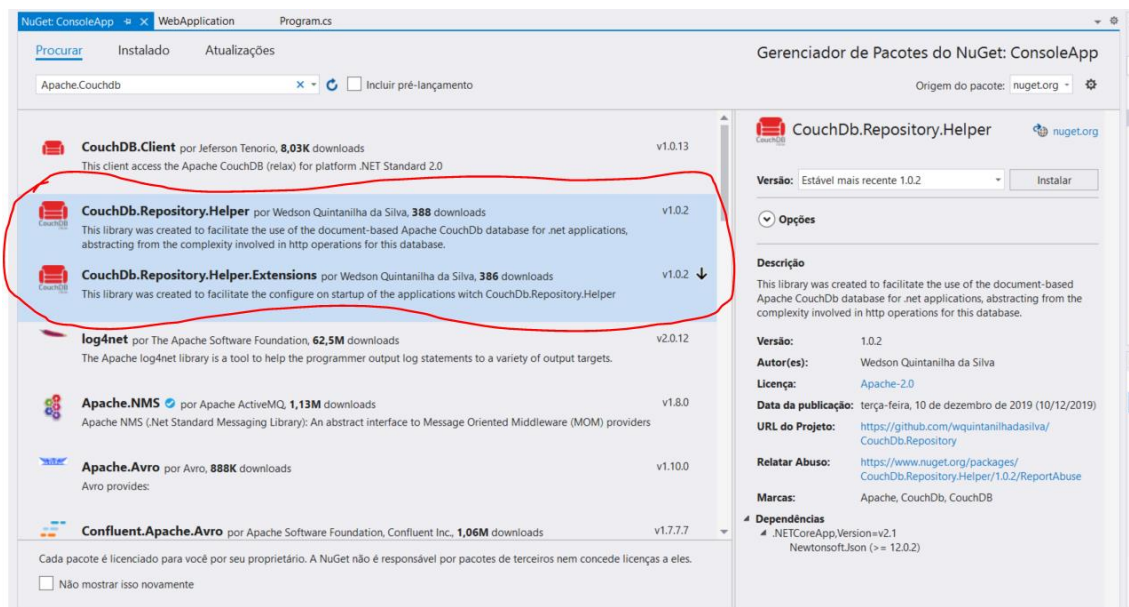
Open the sample solution shown in the image below and follow the explanations below.



Add nuget packages below:



Install packages:



Add config sections on appsettings.json:

```
12 {
13   "Name": "dev",
14   "ServerUrl": "http://localhost:5984",
15   "DatabaseName": "dev-users",
16   "Credential": {
17     "UserName": "jan",
18     "Password": "apple"
19   },
20   "Clusters": [
21     "http://20.0.0.116:5984"
22   ]
23 },
```

“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.

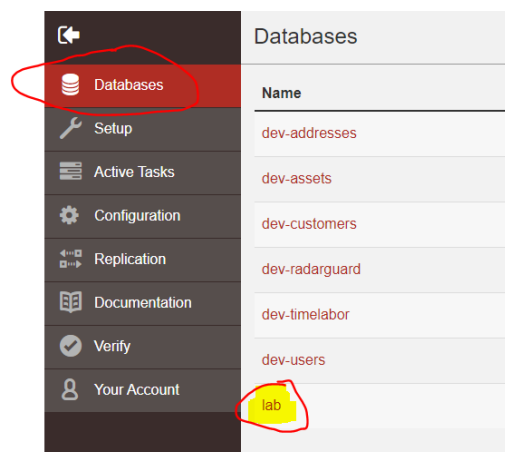
Each context must be defined according to the following structure:

```
10 "CouchDbConnections": {
11   "Contexts": [
12     {
13       "Name": "users-db",
14       "ServerUrl": "http://localhost:5984",
15       "DatabaseName": "lab",
16       "Credential": {
17         "UserName": "admin",
18         "Password": "admin"
19       }
20     },
21   ]
22 }
```

“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:



“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.

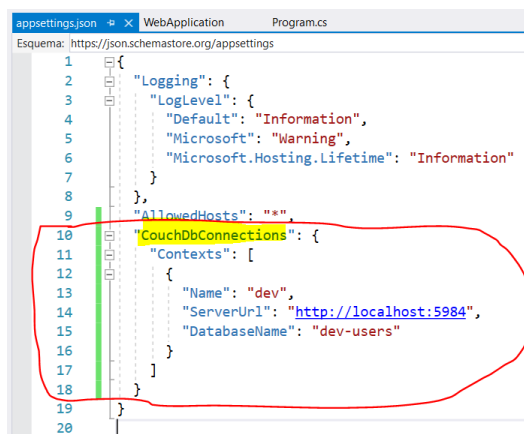
If you have a cluster configuration for your Apache Couchdb, use this configuration to inform the framework that there are clusters that can be triggered if the main server stops responding. The configurations of the clusters must respect what is defined in the Apache CouchDb documentation itself. Enter 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:



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 querye.

ConsoleApplication (startup program – Program.cs class):

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using ClassLibrary;
5 using ConsoleApp.Repositories;
6 using CouchDb.Repository.Helper.Extensions;
7
8 namespace ConsoleApp
9 {
10     0 referências
11     class Program
12     {
13         0 referências
14         static void Main(string[] args)
15         {
16             /**
17              * Indicates the configuration file containing the couchDb
18              * access data [appsettings.json], the name of the section
19              * within this file with these data [CouchDbConnections] and
20              * also the file with the commands mango queries find and
21              * view that will be used by the program [mango-queries.xml].
22              */
23             CouchDbRepositoryExtensions.ConfigureCouchDbHelper("appsettings.json", "CouchDbConnections", "mango-queries.xml");
24
25             Console.WriteLine("CouchDb Helper Hello World use Sample!");
26         }
27     }
28 }

```

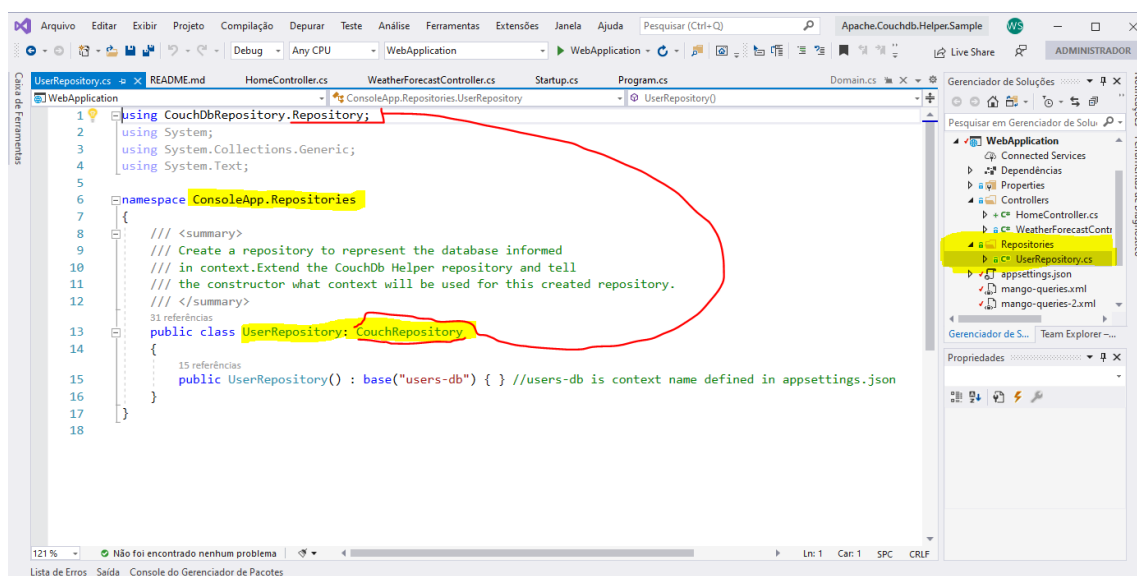
WebApplication (Startup.cs class):

```

1 using Microsoft.AspNetCore.Builder;
2 using Microsoft.AspNetCore.Hosting;
3
4 using Microsoft.Extensions.Configuration;
5 using Microsoft.Extensions.DependencyInjection;
6 using Microsoft.Extensions.Hosting;
7
8 using CouchDb.Repository.Helper.Extensions;
9
10 namespace WebApplication
11 {
12     2 referências
13     public class Startup
14     {
15         0 referências
16         public Startup(IConfiguration configuration)
17         {
18             Configuration = configuration;
19
20             /**
21              * Indicates the configuration file containing the couchDb
22              * access data [appsettings.json], the name of the section
23              * within this file with these data [CouchDbConnections] and
24              * also the file with the commands mango queries find and
25              * view that will be used by the program [mango-queries.xml].
26              */
27             configuration.ConfigureCouchDbHelper("CouchDbConnections", "mango-queries.xml");
28         }
29     }
30 }

```

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:



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.

```

8      },
9      "AllowedHosts": "*",
10     "CouchDbConnections": {
11       "Contexts": [
12         {
13           "Name": "users-db",
14           "ServerUrl": "http://localhost:5984",
15           "DatabaseName": "lab",
16           "Credential": {
17             "Username": "admin",
18             "Password": "admin"
19           }
20         },
21         {
22           "Name": "home",
23           "ServerUrl": "http://localhost:5984",
24           "DatabaseName": "teste",
25           "Credential": {
26             "Username": "jan",
27             "Password": "apple"
28           }
29         },
30         {
31           "Name": "trace",
32           "ServerUrl": "http://localhost:5984",
33           "DatabaseName": "trace",
34           "Credential": {
35             "Username": "jan",
36             "Password": "apple"
37           }
38         }
39       ],
40       "Clusters": [
41         "http://100.1.1.1:5461",
42         "http://localhost:8081"
43       ]
44     }
45   }
46 }

```

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

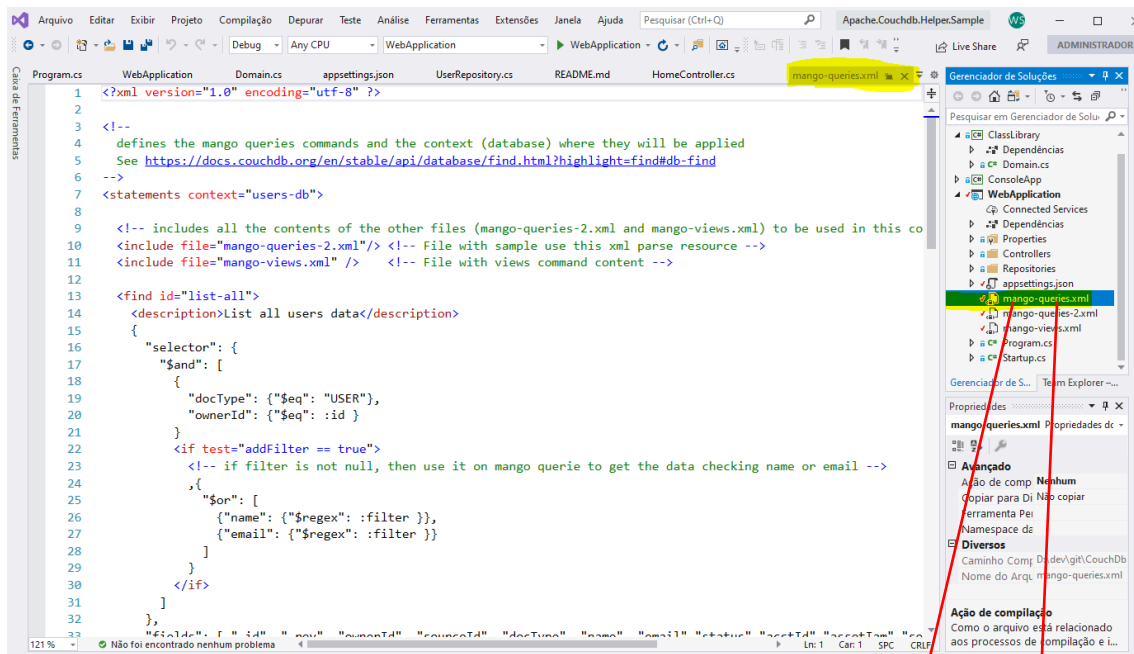
The attributes that will be serialized in the database document must be noted with "JsonProperty". FOR THE sake of INTEGRITY AND RISK, I emphasize that ALL ATTRIBUTES IN THE DATABASE MUST HAVE A PROPERTY IN THE OBJECT, otherwise an exception will be thrown.

```

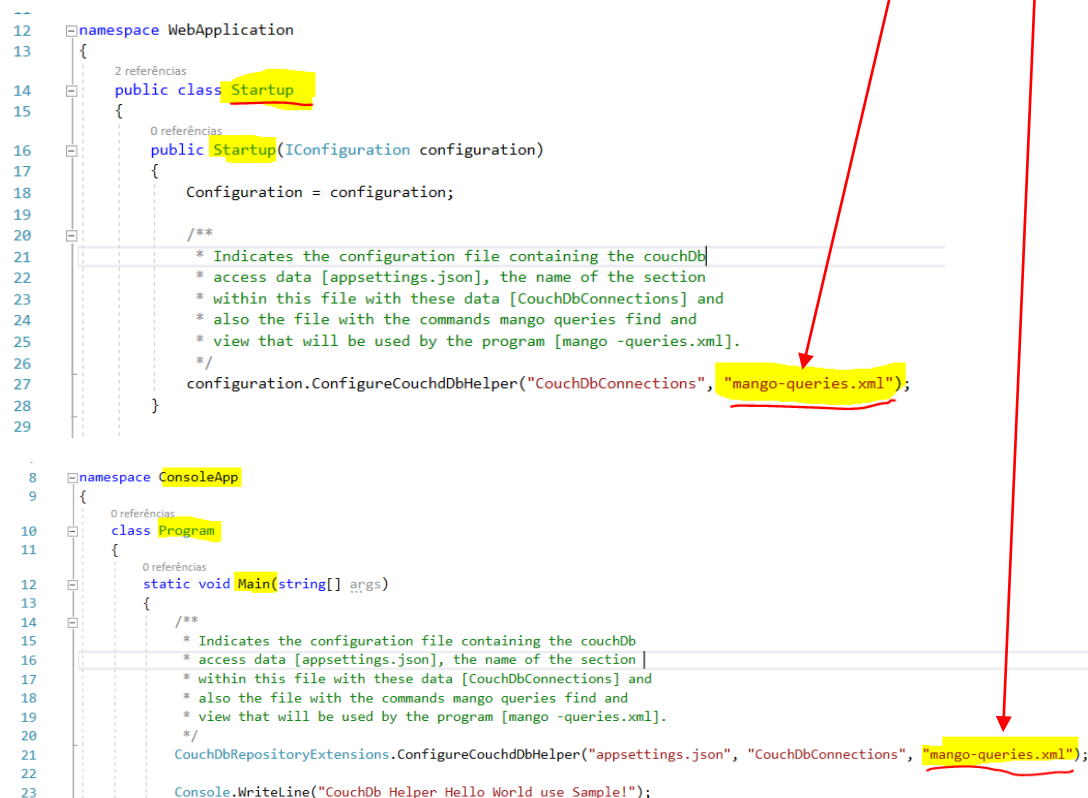
19  /// <summary>
20  /// The objects that represent a document must inherit from AbstractDocument
21  /// and set the generic to the type of the object itself. With this, this
22  /// object must not contain the "_id" and "_rev" properties since the inherited
23  /// class contains these implementations and the services related to these
24  /// two attributes.
25  /// Only the methods mapped with [JsonProperty] attribute will be persisted in the
26  /// document as well as read and filled in automatically.
27  /// </summary>
28  public class User: AbstractDocument<User>
29  {
30
31      [JsonProperty("sourceId")] //Newtonsoft
32      public String SourceId { get; set; }
33
34      [JsonProperty("ownerId")] //Newtonsoft
35      public String OwnerId { get; set; }
36
37      [JsonProperty("name")] //Newtonsoft
38      public String Name { get; set; }
39
40      [JsonProperty("email")] //Newtonsoft
41      public String Email { get; set; }
42
43      [JsonProperty("acctId")] //Newtonsoft
44      public String AcctId { get; set; }
45  }

```

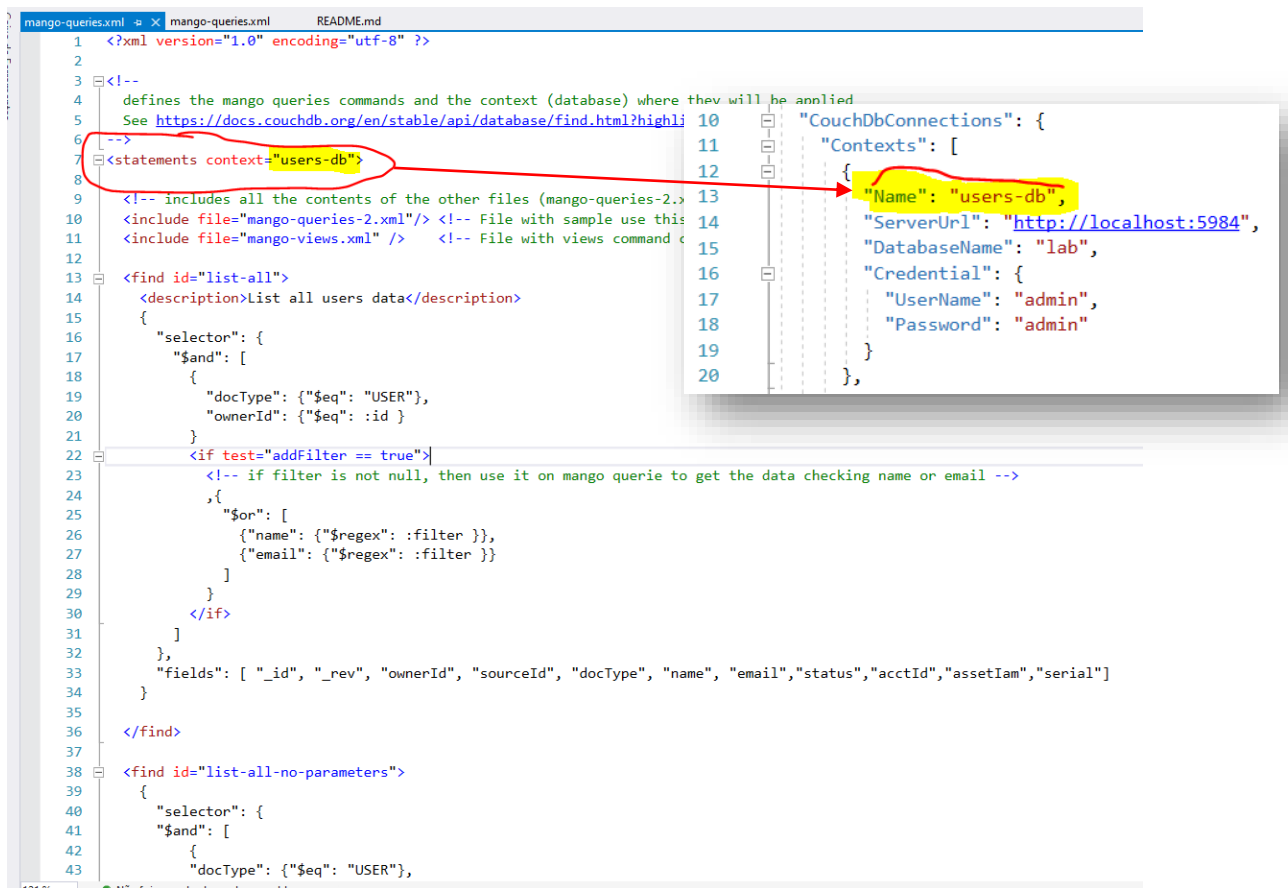
Create the file that will contain the find commands (mango queries) or their template.



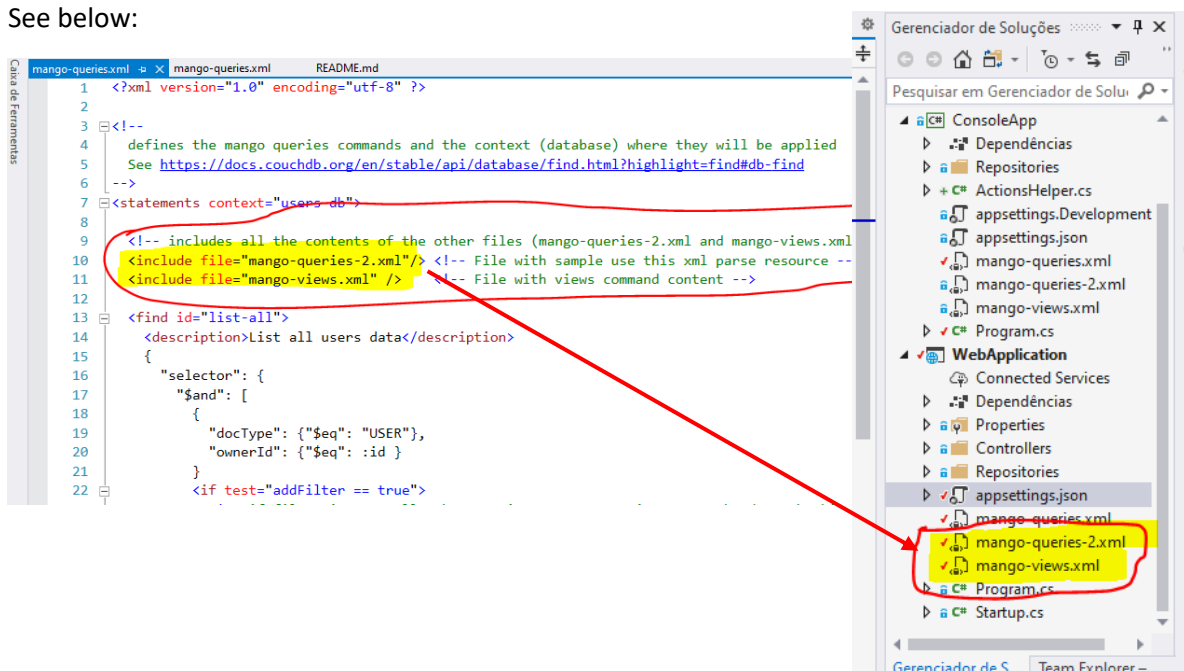
The name of this file must be the same as indicated when initializing the framework configuration there in the initial method. Examples:



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:



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:

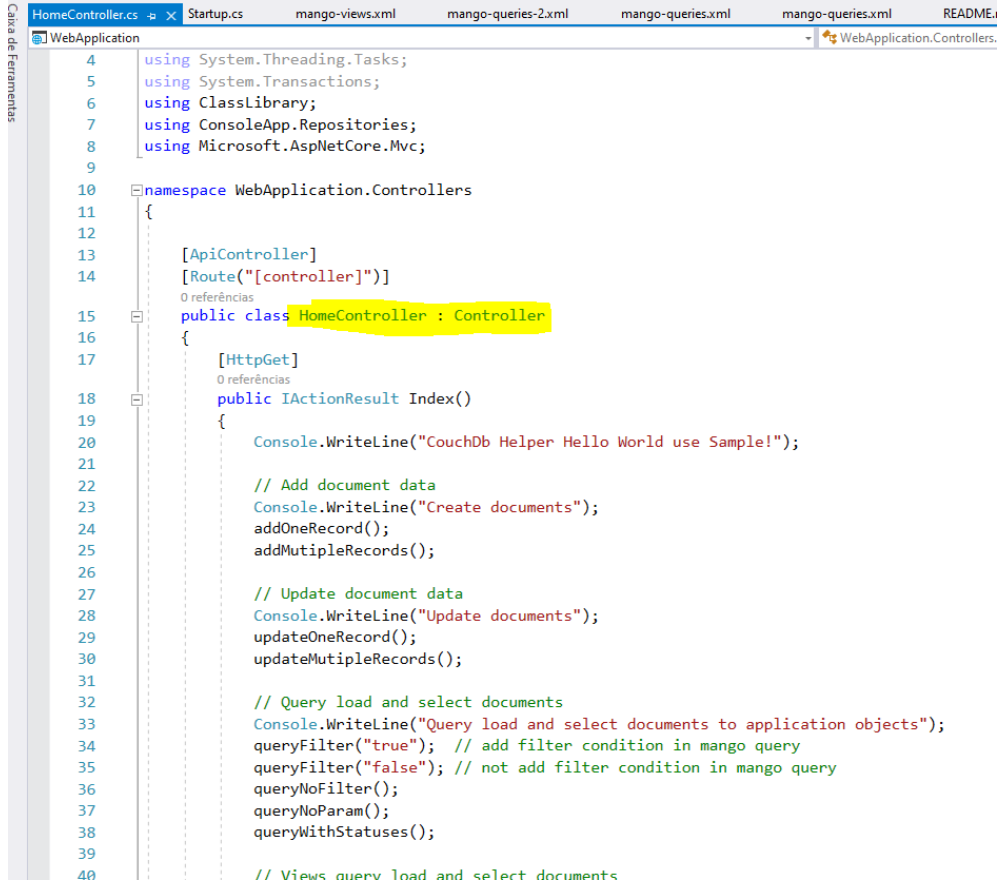


In this file, use the resources available in the couchdb documentation to create your queries: <https://docs.couchdb.org/en/stable/api/database/find.html>

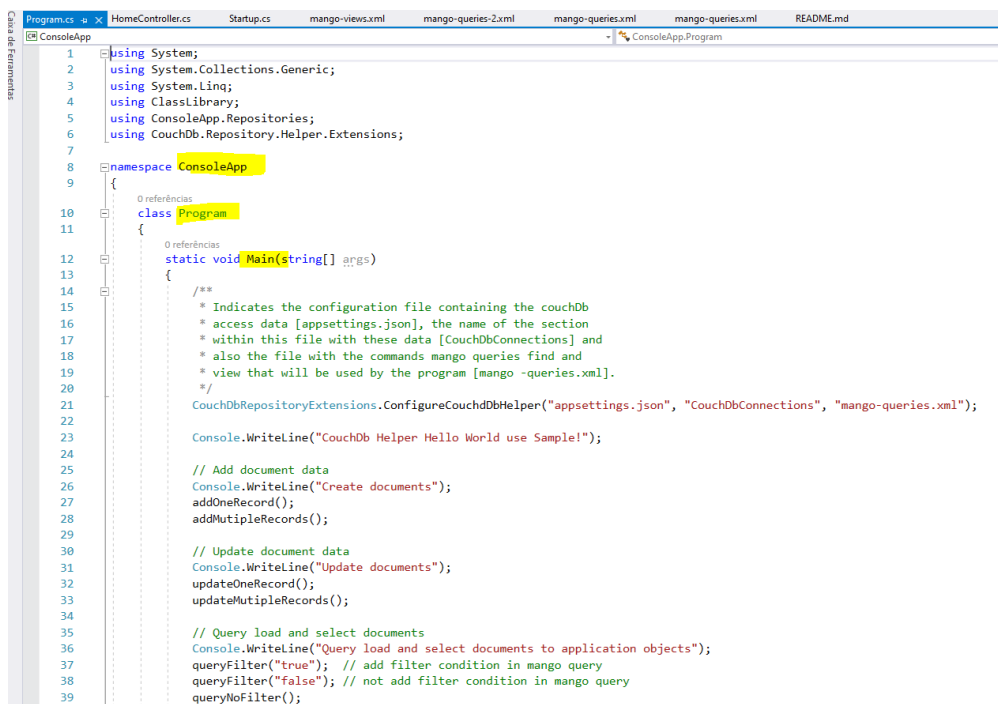
In the sample solution ...

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.



```
4 using System.Threading.Tasks;
5 using System.Transactions;
6 using ClassLibrary;
7 using ConsoleApp.Repositories;
8 using Microsoft.AspNetCore.Mvc;
9
10 namespace WebApplication.Controllers
11 {
12
13     [ApiController]
14     [Route("[controller]")]
15     public class HomeController : Controller
16     {
17         [HttpGet]
18         public IActionResult Index()
19         {
20             Console.WriteLine("CouchDb Helper Hello World use Sample!");
21
22             // Add document data
23             Console.WriteLine("Create documents");
24             addOneRecord();
25             addMutipleRecords();
26
27             // Update document data
28             Console.WriteLine("Update documents");
29             updateOneRecord();
30             updateMutipleRecords();
31
32             // Query load and select documents
33             Console.WriteLine("Query load and select documents to application objects");
34             queryFilter("true"); // add filter condition in mango query
35             queryFilter("false"); // not add filter condition in mango query
36             queryNoFilter();
37             queryNoParam();
38             queryWithStatuses();
39
40             // Views query load and select documents
```



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using ClassLibrary;
5 using ConsoleApp.Repositories;
6 using CouchDb.Repository.Helper.Extensions;
7
8 namespace ConsoleApp
9 {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             /**
15              * Indicates the configuration file containing the couchDb
16              * access data [appsettings.json], the name of the section
17              * within this file with these data [CouchDbConnections] and
18              * also the file with the commands mango queries find and
19              * view that will be used by the program [mango -queries.xml].
20              */
21             CouchDbRepositoryExtensions.ConfigureCouchDbHelper("appsettings.json", "CouchDbConnections", "mango-queries.xml");
22
23             Console.WriteLine("CouchDb Helper Hello World use Sample!");
24
25             // Add document data
26             Console.WriteLine("Create documents");
27             addOneRecord();
28             addMutipleRecords();
29
30             // Update document data
31             Console.WriteLine("Update documents");
32             updateOneRecord();
33             updateMutipleRecords();
34
35             // Query load and select documents
36             Console.WriteLine("Query load and select documents to application objects");
37             queryFilter("true"); // add filter condition in mango query
38             queryFilter("false"); // not add filter condition in mango query
39             queryNoFilter();
```