Previous explanations

The application does not have a developed FrontEnd. My knowledge in FrontEnd is limited and I don't want to give the image of someone who does.

The FrontEnd part has been replaced by Unit Tests that simulate the calls to the BackEnd, which would be the service that manages the Tests.

Projects

Vendon.Core.Application is a class library that makes it easy to instantiate the Tests service in either an MVC application or a Rest application. In this library are defines Services, Mappers, Exceptions and Messages receives an sended to BackEnd.

Vendon.Core.Test.WebApi is an ApiRest WebApp with swagger to interact with backend. To run the application you simply open the solution called and run the application. A swagger screen will be generated in the browser to be able to execute the different endpoints.

EndPoints:

- /Tests/GetAllTestsDefinition
  Do not need parameters. Retrieves all Test names and identifiers from the DB.
- /Tests/GetTest
  It is necessary to provide the identifier of the Test to be retrieved. Is a get method with parameter. /Tests/GetTest?testId=1. The response is a json with Test answer and possible responses. ClientSide do not receive corrrect response.

Response


```
[
  {
    "name": "Test1",
    "testId": 1,
    "answers": [
      {
        "answer": "Answer 1",
        "responses": [
          "Response1",
```

```
            "Response2",
            "Response3"
          ]
        },
        {
          "answer": "Answer 2",
          "responses": [
            "Response1",
            "Response2",
            "Response3"
          ]
        }
      ]
    }
  ]
```

- /Tests/GetTest
  It is a Post method that needs an input json to be able to validate the data by the service in addition to recording the results in the database. En example of json is:

Request

```
{
      "name": "Josele",
      "testId": 1,
      "responses": [
        1,2
      ]
}
```

Response

```
{
  "name": "Test1",
  "numAnswers": 2,
  "numResponsesOk": 2
}
```

Vendon.Core.Repository It is a repository access library, in this case a Cosmos database. The database is built on a test instance of Cosmos on Azure. Contains 2 tables:

- Tests
  Where the Tests defined in the system are stored. PatitionKey it has been defined is /testId. The testId and name fields have been defined as indexed fields since they are the two search criteria that can currently be used.

Indexing

```
{
        "indexingMode": "consistent",
        "automatic": true,
        "includedPaths": [
            {
                "path": "/testId/?"
            },
            {
                "path": "/name/?"
            }
        ],
        "excludedPaths": [
            {
                "path": "/*"
            },
            {
                "path": "/\"_etag\"/?"
            }
        ]
    }
```

- Tests
  Where the results of the tests are saved. PatitionKey it has been defined is /testId, is a good field to group data to scale in physical instances. The testId and name fields have been defined as indexed fields since they are the two search criteria that can currently be used...

Indexing

```
{
        "indexingMode": "consistent",
        "automatic": true,
        "includedPaths": [
            {
                "path": "/testId/?"
            },
            {
                "path": "/name/?"
            }
        ],
        "excludedPaths": [
            {
                "path": "/*"
            },
            {
                "path": "/\"_etag\"/?"
            }
        ]
    }
```

Vendon.Core.ApplicationTest It is a project that contains some unit tests. A Test is configured for real access to DB. The other Test is defined as a BD Mock in case of not having access to real BD.To run the tests you simply have to go in Visual Studio to “Pruebas/Ejecutar todas las pruebas”. El resultado obtenido se ha capturado en el archivo ResultadoEjecucionPruebas.