Water Data Project for Groots

Development - Support API Requirements

[**Scope**](#_usve484jqe7s) **1**

[**Development Guidelines**](#_7fm3u3n6kq2q) **1**

[Coding standards](#_jpmhdbqrhs3c) 1

[Dotnet Framework](#_9eafuai71t1w) 1

[Database](#_ex4zglxui3g) 2

[ORM](#_124xayjblha8) 2

[Bootstrapping](#_ge4qrvqv4kum) 2

[API Configurations](#_ojfv90uf5box) 2

[JWT as authentication method](#_ogphxbssko9w) 2

[Using Claims](#_umikp0foti78) 2

[About Authorize](#_c00wxvho0v7r) 2

[About CORS](#_h4utly23d7nt) 2

[Dependency Injection](#_7p7t3vw7rf2u) 3

[Testing](#_xc09j8c9tmx3) 3

[**About running tests:**](#_j563a56gpivp) **3**

[**Development tools**](#_4vz836szbi59) **3**

[IDE](#_10qwpxwzeilp) 3

[Virtual Box](#_l1dp6m15rwcj) 3

[Docker](#_cxpx7oiglgrd) 4

[**Repositories**](#_etsnmzinue9d) **4**

[About Support API](#_9blnukymx4fy) 4

[Branching](#_2o847vc1s12x) 4

# Scope

What you need to consider in order to develop Support API product.

**These Guidelines ensure the minimal quality requirements to be delivered.**

# Development Guidelines

## Coding standards

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>

## Dotnet Framework

* **Use:** DotNet Core 3.1
* **API**: ASP.NET Core 3.1

## Database

* **Use:** PostgreSQL 12

Use PGAdmin4 to manage databases.

Alternatively you can use a docker-based postgresql. That’s up to you.

## ORM

* **Framework:** Entity Framework Core 3.1
* **Method**: Code First
* **Upload**: Use *Migrations*
* **Library**: Npgsql.EntityFrameworkCore.PostgreSQL

<https://www.nuget.org/packages/Npgsql.EntityFrameworkCore.PostgreSQL/3.1.4>

### Bootstrapping

Ensure that Migrations run on API startup. That means that we need to ensure that database tables are created and minimum data is seed into the db.

## API Configurations

### JWT as authentication method

We need to use token-based communication with our FrontEnd. This means we need to configure JWT on ASP.NET Core auth pipeline.

Use this tutorial to have an idea of the configuration:

<https://www.c-sharpcorner.com/article/authentication-and-authorization-in-asp-net-core-web-api-with-json-web-tokens/>

#### Using Claims

Use a minimal approach of claims:

* SupportAPIUserName
* KoboUser

### About Authorize

You need to determine which Controllers and/or Actions need to have the [Authorize] attribute.

### About CORS

Use these configs only on development phase.

On method Configure (Startup.cs) use:

app.UseCors(x => x.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());

### Dependency Injection

Use the default ASP.NET Core Dependency Injection configuration.

Every Controller, Service and Unit Test need to use Dependency Injection.

## Testing

Tests to be delivered: UnitTests

Use a Mocking library in order to follow a non-database dependent test.

Suggested Libraries (pick one):

* NUnit
* XUnit

Mocking (pick one):

* NSubstitue
* Moq

Suggested complementary:

* Fluent Assertions
* GenFu

### About running tests:

Once tests are delivered, these will be part of the already existing Dockerfile. If a single test fails, the entire Build process will fail. So please create as many tests as you can and ensure everything runs correctly. This will be highly appreciated and reviewed.

# Development tools

## IDE

You can use any flavour or Visual Studio 2019 (preferred):

* Community Edition (if you don’t have a subscription)
* Professional Edition or higher (if you have a subscription)

You can also use any other tool like:

* Rider (needs subscription)

## Virtual Box

We use virtual box 6.x for performing tests of KoBoToolbox inside a Linux Ubuntu 18/20 LTE version installed.

Ensure you configure it and run the Runbook for development so you can access KoBoToolbox locally.

## Docker

We use Docker inside the Ubuntu Box to start KoBoToolbox services.

If you plan to use Postgresql on your local computer, you can use Dockerized version of this db so your development may speed up a bit.

# Repositories

All repos are located under Nexion Bolivia Organization:

* <https://github.com/NexionBolivia>

Support API should be your main repository:

* <https://github.com/NexionBolivia/support-api>

## About Support API

This is the minimal API created to start our integration work on DockerHub.

You can start using this app in order to have all needed features.

You might want to consider adding the following Projects:

* Services
* Unit Tests

## Branching

We established that [Dev branch](https://github.com/NexionBolivia/support-api/tree/dev) is the main one for development. This branch should not be pushed directly.

In order to maintain an adequate follow-up and opportunity to link Github Actions in the future, we ask you to use Branches with name:

* feature/<your-task>
  + Usually take from branch **dev**

Everytime your branch is ready, create a PR and request for review (if applies) and then perform Merge into **DEV** only.