
TEST GENERATOR
PRODUCT DESIGN SPECIFICATION

Version *1.0*
27/09/2018

VERSION HISTORY

[Use the table below to provide the version number, the author implementing the version, the date of the version, the name of the person approving the version, the date that particular version was approved, and a brief description of the reason for creating the revised version.]

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	<i>Samra Salihic</i>	<i>28/09/2018</i>	<i>Vedad Pepa</i>		/

TABLE OF CONTENTS

Contents

1. INTRODUCTION.....	4
1.1. Purpose of The Product Design Specification Document	4
1.2. ABOUT THE APPLICATION	4
2. GENERAL OVERVIEW AND DESIGN GUIDELINES/APPROACH	5
2.1. Assumptions / Constraints / Standards	5
3. ARCHITECTURE DESIGN.....	6
3.1. BACKEND ARCHITECTURE DESIGN	6
3.2. FRONTEND ARCHITECTURE DESIGN.....	7
4. USE-CASE.....	9

1. INTRODUCTION

1.1. PURPOSE OF THE PRODUCT DESIGN SPECIFICATION DOCUMENT

The Product Design Specification document documents and tracks the necessary information required to effectively define architecture and system design in order to give the development team guidance on architecture of the system to be developed. Some portions of this document such as the user interface (UI) may on occasion be shared with the client/user, and other stakeholder whose input/approval into the UI is needed.

1.2. ABOUT THE APPLICATION

This application “Test Generator” serves for automated test generation. The basic goal of the application is to give the user opportunity to store questions. Every question can be described with a question type, difficulty level, number of points, etc. When a user wants to make a test, the application gives him the opportunity to define the test parameters like number of questions, question types, etc. Obviously, it makes it easier to make a specific test.

2. GENERAL OVERVIEW AND DESIGN GUIDELINES/APPROACH

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

2.1. ASSUMPTIONS / CONSTRAINTS / STANDARDS

The application consists of client-side (frontend) and server-side (backend).

For realization of the client-side is used Angular, version 6.0.6, with the following packages and their versions (2.1. Picture).

Package	Version
@angular-devkit/architect	0.6.8
@angular-devkit/build-angular	0.6.8
@angular-devkit/build-optimizer	0.6.8
@angular-devkit/core	0.6.8
@angular-devkit/schematics	0.6.8
@angular/cli	6.0.8
@angular/tsc-wrapped	0.4.2
@ngtools/webpack	6.0.8
@schematics/angular	0.6.8
@schematics/update	0.6.8
rxjs	6.2.1
typescript	2.7.2
webpack	4.8.3

2.1. Picture

For the realization of the server-side is used ASP.NET Core 2.0 including Entity Framework. The package references and their versions are shown in the next picture (2.2. Picture).

```
<PackageReference Include="EntityFramework" Version="6.2.0" />
<PackageReference Include="Microsoft.AspNetCore.All" Version="2.0.8" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="2.1.0" />
<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="2.1.0" />
<PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="2.0.3" />
<PackageReference Include="Swashbuckle.AspNetCore" Version="3.0.0" />
```

2.2 Picture

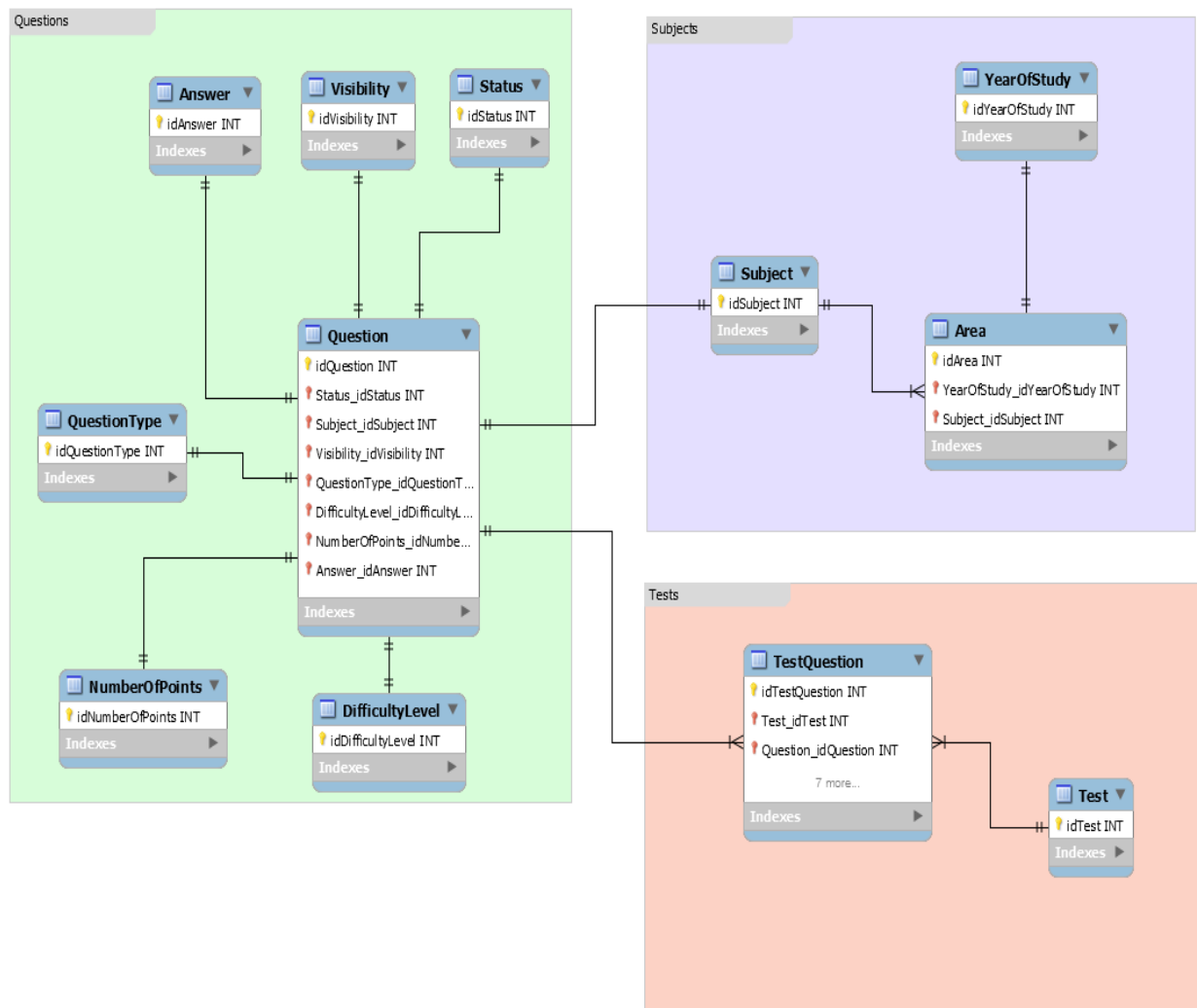
3. ARCHITECTURE DESIGN

This section outlines the system and hardware architecture design of the system that is being built.

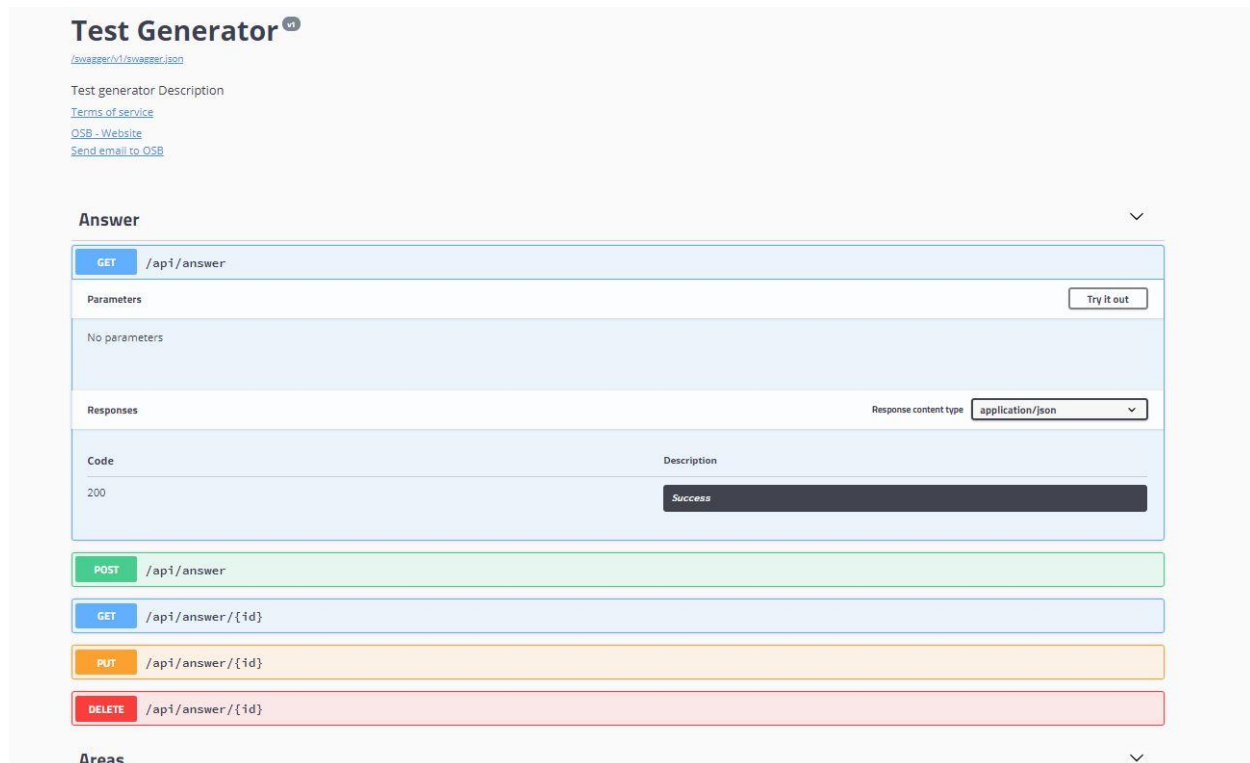
At the previous capture we saw that the application contains of two separate applications – backend and frontend.

3.1. BACKEND ARHITECTURE DESIGN

The backend architecture design is based on MVC pattern. All about the backend structure, methods and models we can see when we run the application and open “./swagger/index.html”. The swagger also shows the models implemented in the application, but for a better resonation we can see the Entity Relationship Diagram on the next picture.



The swagger documentation example is shown in the following picture.



3.2. FRONTEND ARCHITECTURE DESIGN

For the frontend documentation we used Compodoc. To see the documentation you have to run “compodoc src -s” command in the application terminal.


The Compodoc documentation shows us Dependencies, Modules, Classes, Interceptors, Routes, etc. Now we can see the whole architecture design.

What needs to be emphasized is that for authentication JWT (Json Web Token) is used. The example picture of Compodoc documentation is shown in the next picture.

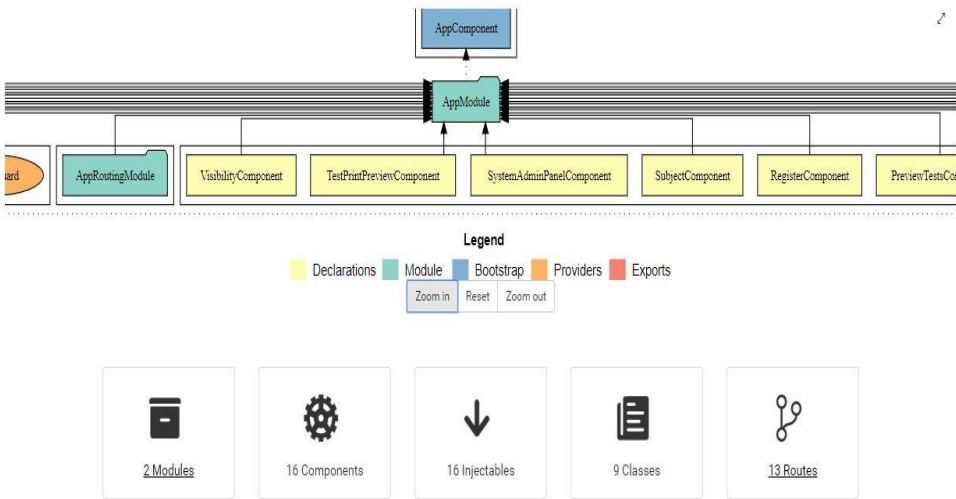
test-generator documentation

Type to search

Getting started
Overview
README
Dependencies
Modules
Classes
Injectables
Interceptors
Miscellaneous
Routes
Documentation coverage

Documentation generated using


Overview



Legend

- Declarations
- Module
- Bootstrap
- Providers
- Exports

Zoom in Reset Zoom out

2 Modules

16 Components

16 Injectables

9 Classes

13 Routes

4. USE-CASE

