Relatório de projeto

# Autor

|  |  |  |  |
| --- | --- | --- | --- |
| **Email** | **Nome** | **Versão** | **Data** |
| Dvalentim.ribeiro@gmail.com | Daniel Ribeiro | 1 | 2018-02-12 |
|  |  |  |  |

# Sumário

O presente documento descreve as principais decisões na elaboração da Web Api. Foram criados três controladores que correspondem às três funcionalidades solicitadas:

* Verificação do estado do servidor. **HealthController**
* Obtenção e alteração de questões. **QuestionsController**
* Partilha de ecrã entre utilizadores. **ShareController**

No total foram utilizadas 4 horas de esforço para a concretização da solução, em que uma hora foi destinada a análise e desenho, as duas horas seguintes foram utilizadas na implementação e a última hora foi usada na elaboração do relatório final e *deploy* no **GitHub.** Todo o projeto foi desenvolvido no *Visual Studio 2015 Community Edition.*

# Modelo

As duas classes da figura 1 (Question e Choice) representam a relação entre questões e respostas utilizadas pelo controlador **QuestionsController**. Não foram criadas classes para representar o estado dos restantes controladores, mas poderá ser equacionado em futuras evoluções.



Figura 1 - Relação entre questões e opções de escolha. Notação UML

# Implementação

A concretização dos controladores foi iniciada com recurso à funcionalidade de *scaffolding* disponível no Visual Studio. As respostas dos controladores **HealthController** e **ShareController**  são simuladas.

O treço de código seguinte ilustra funcionalidade de pesquisa de questões. Tal como descrito no requisito funcional da API, considerou-se obrigatório o uso dos atributos *limit* e *offset* e marcou-se como opcional o atributo *filter.*

Retorna um conjunto de questões desde que o nome (question) contenha o valor do filtro ou qualquer uma das escolhas (choice) contenha nos seus atributos (choice e votes) o valor do filtro.

1. IQueryable  <  Question  >  questions  =  db.Questions;
2. var  hasFilter  =  dto.filter  !=  null;
3. if  (hasFilter)  {
4. var  filter  =  dto.filter.Trim().ToLower();
5. questions  =  db.Questions.Where(q  =>  q.question.Contains(filter)  ||  q.Choices.Any(c  =>  c.choice.Contains(filter)  ||  c.votes.ToString().Contains(filter)));
6. }  //Pagining
7. questions  =  questions.OrderBy(q  =>  q.id).Skip(dto.offset).Take(dto.limit);
8. return  Ok(questions);