# TECHNICAL RADIATION

# Technical Radiation

In this assignment your job is to create a RESTful API using .NET Core for a new emerging company called **Technical Radiation**. The company specializes in providing users with news on technical matters, e.g. new gadgets, new technology, etc… The manager is in love with the REST approach, so don't disappoint him!

## Assignment description

Here below are the functionality that should be provided in this API. The routes are categorized in unauthorized and authorized routes, depending on whether you need authentication or not to access the routes.

### Rules

This applies to all endpoints enlisted below:

- *If the item is not found with the given filter or restriction, the api should return a suitable HTTP status code for when a resource is not found.*
- *If a model is not correctly structured when either using POST or PUT, the api should return a suitable HTTP status code to indicate that there was an error made by the client.*
- *If an error occurs on the server, the client should be notified with a suitable HTTP status code.*
- *When using POST or PUT, the data sent to the server should be on JSON format within the request body.*
- *When a resource is created, HATEOAS should be honored for better navigation.*
- *Responses that make use of HATEOAS should include a property called **links** which should include both **rel** and **href** which describes the relationship and the anchor itself.*

## Models

The **ModifiedBy** property on all models should just be some hard-coded value, e.g. **TechnicalRadiationAdmin**

- **NewsItemDto**
  - **Id, Title, ImgSource, ShortDescription**
- **NewsItemDetailDto**
  - **Id, Title, ImgSource, ShortDescription, LongDescription, PublishDate**
- **NewsItemInputModel**
  - **Title** (required)**, ImgSource** *(required and must be a valid URL)***, ShortDescription** *(required and max length 50)***, LongDescription** *(min length 50, max length 255)***, PublishDate** *(required)*
- **NewsItem**
  - **Id, Title, ImgSource, ShortDescription, LongDescription, PublishDate, ModifiedBy** *(code-generated)***, CreatedDate** *(code-generated)***, ModifiedDate** *(code-generated)*
- **AuthorDto**
  - **Id, Name**
- **AuthorDetailDto**
  - **Id, Name, ProfileImgSource, Bio**
- **AuthorInputModel**
  - **Name** *(required)***, ProfileImgSource** *(required and must be a valid URL)***, Bio** *(max length 255)*
- **Author**
  - **Id, Name, ProfileImgSource, Bio, ModifiedBy** *(code-generated)***, CreatedDate** *(code-generated)***, ModifiedDate** *(code-generated)*
- **NewsItemAuthors**
  - **AuthorId**
  - **NewsItemId**
- **CategoryDto**
  - **Id, Name, Slug**
- **CategoryDetailDto**
  - **Id, Name, Slug, NumberOfNewsItems**
- **CategoryInputModel**
  - **Name** *(required and max length 60)*
- **Category**
  - **Id, Name, Slug, ModifiedBy** *(code-generated)***, CreatedDate** *(code-generated)***, ModifiedDate** *(code-generated)*
- **NewsItemCategories**
  - **CategoryId**
  - **NewsItemId**

## (40%) Unauthorized routes

**NEWS ITEMS (15%)**

1. **(10%)** Create a base endpoint (**/api**) which fetches all news. By default there should be a limit of 25 news items. The limit can be changed via a query parameter called **pageSize**. The pages can be indexed by a query parameter called **pageNumber**. The news items should be ordered by a descending publish date. This endpoint should use an **envelope** to encapsulate the paging, the data seen below are only describing the items within the envelope.

```
/* NewsItem */
[{
    "id": 1,
    "title": "Title of the news article",
    "imgSource": "https://example.com/news-item/1.jpg",
    "shortDescription": "A short description of the news article",
    "_links": {
        "self": { "href": "api/1" },
        "edit": { "href": "api/1" },
        "delete": { "href": "api/1" },
        "authors": [ { "href": "api/authors/1" } ],
        "categories": [
            { "href": "api/categories/1" },
            { "href": "api/categories/2" }
        ]
    }
}]
```

2. **(5%)** Create an endpoint (**/api/{newsItemId}**) which accepts an URL parameter for the news item id. This endpoint should retrieve a news item by id.

```
/* NewsItemDetail */
{
    "id": 1,
    "title": "Title of the news article",
    "imgSource": "https://example.com/news-item/1.jpg",
    "shortDescription": "A short description of the news article",
    "longDescription": "A full description of the news article",
    "publishDate": "24/07/2018, 13:11:56",
    "_links": {
        "self": { "href": "api/1" },
        "edit": { "href": "api/1" },
        "delete": { "href": "api/1" },
        "authors": [ { "href": "api/authors/1" } ],
        "categories": [
            { "href": "api/categories/1" },
            { "href": "api/categories/2" }
        ]
    }
}
```

**CATEGORIES (10%)**

3. **(5%)** Create an endpoint (**/api/categories**) which fetches all categories.

```
/* Category */
[{
    "id": 1,
    "name": "Gadgets",
    "slug": "gadgets",
    "_links": {
        "self": { "href": "api/categories/1" },
        "edit": { "href": "api/categories/1" },
        "delete": { "href": "api/categories/1" }
    }
}]
```

4. **(5%)** Create an endpoint (**/api/categories/{categoryId}**) which accepts an URL parameter for the category id. This endpoint should retrieve a category by id.

```
/* CategoryDetail */
{
    "id": 1,
    "name": "Gadgets",
    "slug": "gadgets",
    "numberOfNewsItems": 12,
    "parentCategoryId": 0,
    "_links": {
        "self": { "href": "api/categories/1" },
        "edit": { "href": "api/categories/1" },
        "delete": { "href": "api/categories/1" }
    }
}
```

**AUTHORS (15%)**

5. **(5%)** Create an endpoint (**/api/authors**) which fetches all authors.

```
/* Author */
[{
    "id": 1,
    "name": "John Doe",
    "_links": {
        "self": { "href": "api/authors/1" },
        "edit": { "href": "api/authors/1" },
        "delete": { "href": "api/authors/1" },
        "newsItems": { "href": "api/authors/1/newsItems" },
        "newsItemsDetailed": [
            { "href": "api/1" },
            { "href": "api/2" }
        ]
    }
}]
```

6. **(5%)** Create an endpoint (**/api/authors/{authorId}**) which accepts an URL parameter for the author id. This endpoint should retrieve an author by id.

```
/* AuthorDetail */
{
    "id": 1,
    "name": "John Doe",
    "profileImgSource": "https://example.com/authors/1.jpg",
    "bio": "My name is John Doe and I love writing books!",
    "_links": {
        "self": { "href": "api/authors/1" },
        "edit": { "href": "api/authors/1" },
        "delete": { "href": "api/authors/1" },
        "newsItems": { "href": "api/authors/1/newsItems" },
        "newsItemsDetailed": [
            { "href": "api/1" },
            { "href": "api/2" }
        ]
    }
}
```

7. **(5%)** Create an endpoint (**/api/authors/{authorId}/newsItems**) which accepts an URL parameter for the author id. This endpoint should retrieve all news items written by the author with the provided id.

```
/* NewsItem */
[{
    "id": 1,
    "title": "Title of the news article",
    "imgSource": "https://example.com/news-item/1.jpg",
    "shortDescription": "A short description of the news article",
    "_links": {
        "self": { "href": "api/1" },
        "edit": { "href": "api/1" },
        "delete": { "href": "api/1" },
        "authors": [ { "href": "api/authors/1" } ],
        "categories": [
            { "href": "api/categories/1" },
            { "href": "api/categories/2" }
        ]
    }
}]
```

## (50%) Authorized routes

The authorized routes check for a secret key which is stored within the Authorization HTTP request header. This secret key can be hard-coded on the server side. The authorized routes should be decorated with a custom attribute which checks whether the incoming Authorization request header is valid.

**NEWS ITEMS (12.5%)**

1. **(5%)** Create an endpoint (**/api**) which creates a news item.

```
/* NewsItemDetail */
{
    "id": 1,
    "title": "Title of the news article",
    "imgSource": "https://example.com/news-item/1.jpg",
    "shortDescription": "A short description of the news article",
    "longDescription": "A full description of the news article",
    "publishDate": "24/07/2018, 13:11:56",
    "_links": {
        "self": { "href": "api/1" },
        "edit": { "href": "api/1" },
        "delete": { "href": "api/1" },
        "authors": [ { "href": "api/authors/1" } ],
        "categories": [
            { "href": "api/categories/1" },
            { "href": "api/categories/2" }
        ]
    }
}
```

2. **(5%)** Create an endpoint (**/api/{newsItemId}**) which updates an existing news item.

3. **(2.5%)** Create an endpoint (**/api/{newsItemId}**) which deletes an existing news item.

**CATEGORIES (20%)**

4. **(5%)** Create an endpoint (**/api/categories**) which creates a category. **Slug** should be generated by making the name lowercase and join the words together with a hyphen (-), e.g. Science Fiction will be science-fiction

```
/* CategoryDetail */
{
    "id": 1,
    "name": "Gadgets",
    "slug": "gadgets",
    "numberOfNewsItems": 12,

    "_links": {
        "self": { "href": "api/categories/1" },
        "edit": { "href": "api/categories/1" },
        "delete": { "href": "api/categories/1" }
    }
}
```

5. **(7.5%)** Create an endpoint (**/api/categories/{categoryId}**) which updates an existing category.

6. **(2.5%)** Create an endpoint (**/api/categories/{categoryId}**) which deletes an existing category.

7. **(5%)** Create an endpoint (**/api/categories/{categoryId}/newsItems/{newsItemId}**) which links a news item to a specific category. The id for the category and news item are provided as URL parameters.

**AUTHORS (17.5%)**

8. **(5%)** Create an endpoint (**/api/authors**) which creates an author.

```
/* AuthorDetail */
{
    "id": 1,
    "name": "John Doe",
    "profileImgSource": "https://example.com/authors/1.jpg",
    "bio": "My name is John Doe and I love writing books!",
    "_links": {
        "self": { "href": "api/authors/1" },
        "edit": { "href": "api/authors/1" },
        "delete": { "href": "api/authors/1" },
        "newsItems": { "href": "api/authors/1/newsItems" },
        "newsItemsDetailed": [
            { "href": "api/1" },
            { "href": "api/2" }
        ]
    }
}
```

9. **(5%)** Create an endpoint (**/api/authors/{authorId}**) which updates an existing author.

10. **(2.5%)** Create an endpoint (**/api/authors/{authorId}**) which deletes an existing author.

11. **(5%)** Create an endpoint (**/api/authors/{authorId}/newsItems/{newsItemId}**) which links an author to a news item. The id for the author and news item are provided as URL parameters.

## (10%) Structure

1. **(5%)** The assignment should be setup using a three-layer structure where you make use of a presentation layer, service layer, repository layer and a project which keeps all models
2. **(2.5%)** All models should reside in the Models project (**DTO, InputModels** and **Entities**)
3. **(2.5%)** The data should be in static lists of news items, authors and categories which reside in the repository layer (*which you need to populate yourself*)

# Submission

Submit a single compressed file (**.zip, *.rar*) in Canvas. If you are working in groups, please remember to comment the names of the group members in the comment section in Canvas.

# Other

In this assignment it is required to use .NET Core. Any editor will suffice, although Visual Studio Code is a good option.