

Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное учреждение высшего образования  
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ  
И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании

ОТЧЁТ

Лабораторная работа №2

«Разработка бизнес-логики приложения»

Проверил

Доцент каф. КСУП

\_\_\_\_\_ Горяинов А. Е.

«13» октября 2020 г.

Выполнил

Студент гр. 587-1

\_\_\_\_\_ Пономаренко Н. С.

«13» октября 2020 г.

Томск, 2020 г.

## Содержание

1 Введение.....	3
2 UML-диаграмма классов проекта логики с пояснением назначения классов....	4
3 Пример исходного кода, демонстрирующий взаимодействие с сущностью.....	6
4 Текущая история коммитов ветки dev.....	8
5 Заключение.....	9

## **1 Введение**

Цель работы: изучить типовые требования, предъявляемые к бизнес-логике приложения, получить умения разработки логики приложения с обеспечением данных требований.

Задачи:

1. Изучить требования и процесс разработки логики приложения.
2. Повторить синтаксис языка C# для разработки объектно-ориентированных программ.
3. Разработать классы, необходимые для работы логики приложения.
4. Обеспечить целостность данных классов с помощью свойств и механизма генерации исключений.

## 2 UML-диаграмма классов проекта логики с пояснением назначения классов

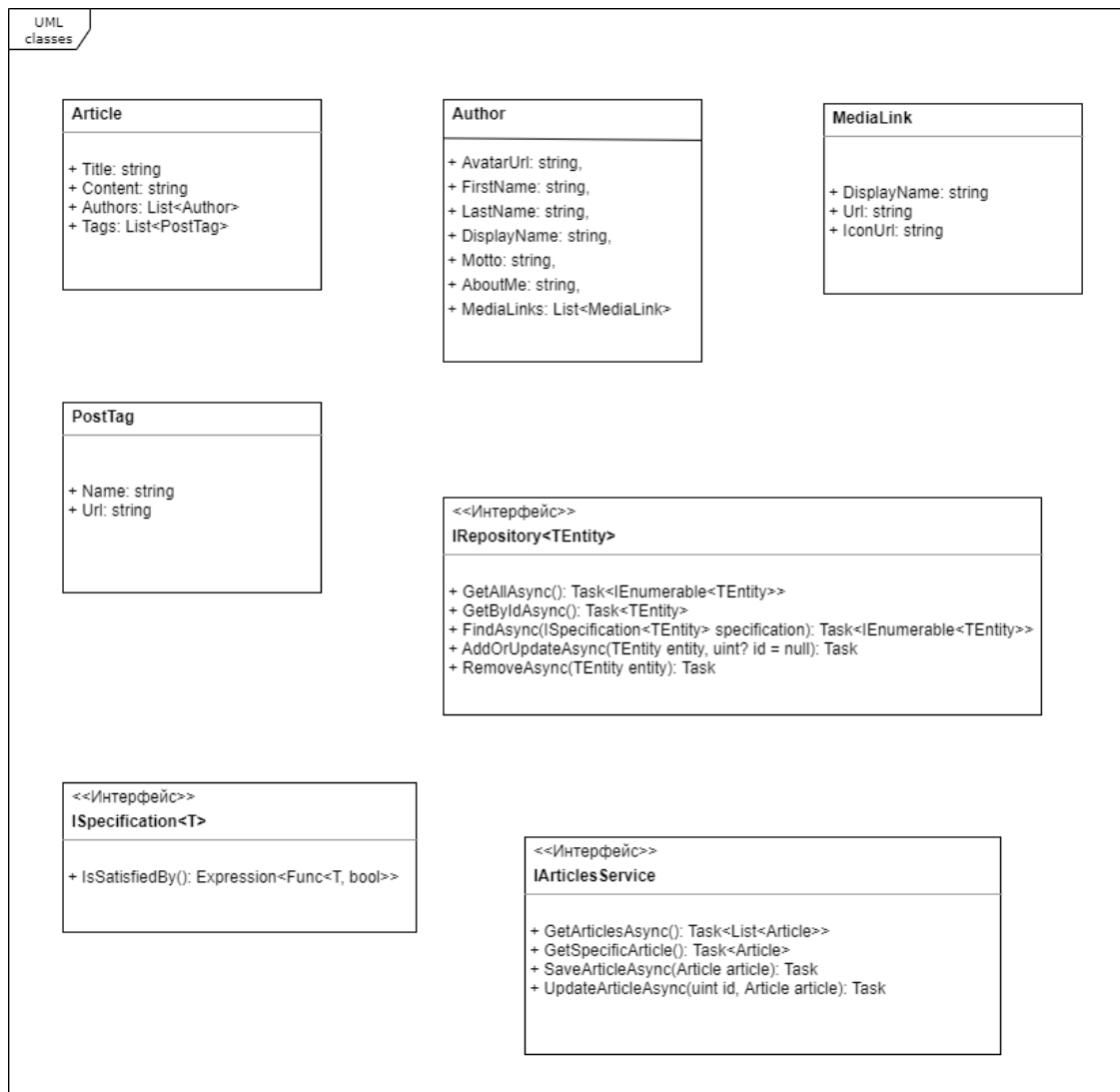


Рисунок 1 UML-диаграмма

Сущность “Article” представляет собой единицу контента для определённой темы.

Сущность “Author” представляет собой писателя, который ведёт блог.

Сущность “MediaLink” представляет собой ссылку на медиаресурс.

Сущность “PostTag” представляет собой тег, указывающий на одну из категорий поста.

Интерфейс “IRepository<TEntity>” представляет собой типичное описание паттерна «Репозиторий» для работы с уровнем базы данных и её структурой. TEntity представлен любой моделью, данные по которой предполагается возвращать.

Интерфейс “ISpecification<T>” представляет собой типичное описание паттерна «Спецификация» для передачи сложного и гибкого выражения для последующего использования при, например, фильтрации или поиске.

Интерфейс “IArticlesService” представляет собой сервис работы с постами.

### 3 Пример исходного кода, демонстрирующий взаимодействие с сущностью

```
[Route("api/[controller]")]
[ApiController]
public class ArticlesController : ControllerBase
{
    private readonly IArticlesService _articles;

    public ArticlesController(IArticlesService articles)
    {
        _articles = articles;
    }

    [HttpGet]
    public async Task<IActionResult> GetArticlesAsync()
    {
        var data = await _articles.GetArticlesAsync();
        if (null != data)
            return new OkObjectResult(data);
        return new NotFoundResult();
    }

    [HttpGet("{id}")]
    public async Task<IActionResult> GetArticleByIdAsync(uint id)
    {
        var data = await _articles.GetSpecificArticle(id);
        if (null != data)
            return new OkObjectResult(data);
        return new NotFoundResult();
    }
}
```

```

[HttpPost]
public async Task<IActionResult> PostAsync([FromBody] Article article)
{
    if (null == article)
        return new BadRequestResult();

    await _articles.SaveArticleAsync(article);

    return new OkResult();
}











[HttpPut("{id}")]
public async Task<IActionResult> PutAsync(uint id, [FromBody] Article article)
{
    if (null == article)
        return new BadRequestResult();

    await _articles.UpdateArticleAsync(id, article);

    return new OkResult();
}
}

```

## 4 Текущая история коммитов ветки dev

Commits on Oct 13, 2020		
- Added sqlite db; 	 SnowPowerCore committed 22 minutes ago	 5dbf7ab 
Commits on Sep 29, 2020		
- Lab1;	 SnowPowerCore committed 14 days ago	 177f387 
- Improved algorithm	 SnowPowerCore committed 14 days ago	 dd5477e 



## **5 Заключение**

В результате выполнения лабораторной работы:

- Созданы модели для серверной и клиентской части, сущности для серверной части приложения;
- Подключена база данных, применены миграции и настроены таблицы;
- Создан один репозиторий, сервис и контроллер;
- Осуществлено взаимодействие с информацией в базе данных посредством вызовов API.