

НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАЦИОННИ ТЕХНОЛОГИИ 2022/2023

Иво Недев

Януари 2023

Съдържание

1	Тема	2
2	Автор	2
3	Ръководител	2
4	Резюме	2
4.1	Цели	2
4.2	Описание на приложението	2
4.3	Основни етапи в реализирането	2
4.3.1	Проучване на темата и подготовка на работна среда	3
4.3.2	Създаване на база данни	3
4.3.3	Визуализация на уебсайта	4
4.3.4	Проверки на кода и подобряване на архитектурата на прокта	6
4.3.5	Съставяне на придружаващите документи	7
4.4	Ниво на сложност на проекта	7
5	Логическо и функционално описание на решението	8
5.1	Учебни Помагала	8
5.2	Потребители	8
5.3	Администратори	11
5.4	Сигурност и валидация на входните данни	11
6	Технологични средства	12
7	Използвана литература	12
8	Заключение	12

1 Тема

"BookShop"

2 Автор

Иво Недков Недев,
гр. Велико Търново, ул.“Краков“ №9, ет.4, ап.15,
ученик от XI.д клас на Природо-математическа гимназия „Васил
Друмев“, гр. Велико Търново,
тел.: 0885778908, mail: ovinedev@abv.bg

3 Ръководител

Георги Игнатов,
учител по Информационни технологии в Природо-математическа
гимназия „Васил Друмев“, гр. Велико Търново,
тел.:0889255850, mail: g_ignatov@mail.bg

4 Резюме

4.1 Цели

Проектът цели да предостави онлайн платформа за търговия на учебници и учебни помагала втора ръка. Чрез него учениците могат лесно да намерят учебници готови за купуване.

4.2 Описание на приложението

Проектът "BookShop" е уебсайт, който е предназначен за ученици. Той съдържа възможност за създаване на профил. Профилите биват два вида - потребителски [5.2] и администраторски 5.3. Те се различават по техните права и функции в проекта. Също така в проекта са включени акаунти подразбиране, за улеснение на негово представяне.

4.3 Основни етапи в реализирането

Тук ще бъдат разгледани основните етапи, през които е минало създаването на проекта BookShop

4.3.1 Проучване на темата и подготовка на работна среда

Разгледах различни сайтове, които предлагат подобни услуги. Примери:

olx.bg

mobile.bg

bguchebnik.com

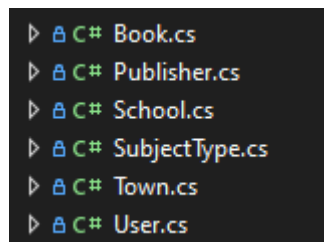
Създадох Git Repository, за да контролирам работата си върху проекта. То може да бъде видяно [тук](#).

4.3.2 Създаване на база данни

Чрез промени върху примерния модел за разработка на уебсайтове, предоставен от Microsoft, създадох класове-обекти [1](#). За да осъществи връзка между Visual Studio и SSMS използвам класа ApplicationDbContext, който наследява IdentityDbContext. В него се иницилизират класовете, който ще бъдат използвани за създаването на базата данни, както и имената на таблиците. След това се добавя нова миграция, която има за цел да превърне C# код в SQL заявки. Чрез този метод на комуникация, се избягва възможността за SQL-injections. Във всеки от класовете са описани техните полета, които са колоните във всяка от таблиците. За да могат таблиците да комуникират помежду си, се използват специалните колони наречени "ключове". Всеки запис в дадена таблица съдържа уникален ключ, чрез който се отличава от другите записи, така той може да бъде достъпен от приложението. За повечето от тези ключове съм използвал цели числа, но поради опасения за сигурност таблицата, съдържащата информация за потребителите, използва случайно генериран текст. Повече за сигурността на данните: [5.4](#)

След това, свързах проекта със SQL Server Management Studio (SSMS), чрез Connection String [2](#). Той съдържа важна информация като името на базата данни, която ще бъде използвана от приложението. Намира в appsettings на приложението.

След изпълнението на тези стъпки се получава уебсайт, с възможности да извършва CRUD операции върху база данни, съставена от множество таблици, комуникиращи помежду си. [3](#)



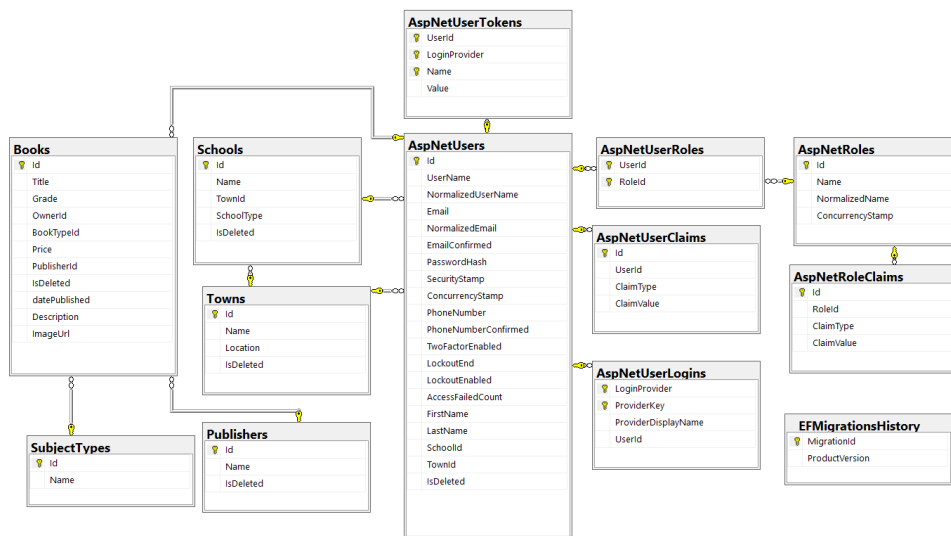
```
▸ Book.cs
▸ Publisher.cs
▸ School.cs
▸ SubjectType.cs
▸ Town.cs
▸ User.cs
```

Фиг. 1: Имената на класовете-обекти

Фиг. 2: Connection string

```
"ConnectionStrings": {
  "DefaultConnection": "Server=DESKTOP-HI2ESS7;Database=BookShop;Trusted_Connection=True;MultipleActiveResultSets=true"
}
```

Фиг. 3: Таблиците в БД



4.3.3 Визуализация на уебсайта

За да потребителят да използва проекта пълноценно е нужен графичен интерфейс. Използвах Razor, комбинация от C#, HTML, CSS и JavaScript, да направя уеб страници. Те имат връзка с базата данни, която се осъществява от контролерите и сървисите 4. Сървисите наследяват интерфейсите и обменят информация между контролерите и БД, във вид на класове. Контролерите отговарят за зареждането на желаната от потребителя уеб страница с правилната информация от базата данни. За да могат потребителите да променят съдържанието в базата данни, бе нужно имплементирането на така наречените формуляри, в които потребителят изписва желаната от него информация и тя бива изпратена към базата данни 5.

```

[HttpPost]
[AllowAnonymous]
public async Task<ActionResult> Register(RegisterModel model)
{
    model.Schools = townsService.GetAllSchools();
    model.Towns = townsService.GetAll();

    if (!ModelState.IsValid)
    {
        return View(model);
    }

    User user = new User();
    var result = await userManager.CreateAsync(user, model.Password);

    if (result.Succeeded)
    {
        await userManager.AddToRoleAsync(user, "User");
        return RedirectToAction("Login", "Users");
    }

    foreach (var item in result.Errors)
    {
        ModelState.AddModelError("", item.Description);
    }

    return View(model);
}

```

(a) Контролер

```

public class BooksService : IBooksService
{
    private readonly ApplicationDbContext context;
    private readonly HtmlSanitizer sanitizer;

    public BooksService(
        ApplicationDbContext context)
    {
        this.context = context;
        sanitizer = new HtmlSanitizer();
    }

    public async Task Add(AddBookView model, string userId)
    {
        // ...
    }

    public async Task<BooksQueryServiceModel> All(
        string? subject = null,
        string? searchTerm = null,
        BooksSorting sorting = BooksSorting.Newest,
        int currentPage = 1,
        int booksPerPage = 5)
    {
        var bookQuery = context.Books.Where(b => b.IsDeleted == false).AsQueryable();

        if (subject != null)
        {
            bookQuery = context.Books.Where(x => x.SubjectType.Name == subject);
        }
    }
}

```

(b) Сървис

Фиг. 4: Части от кода на контролер и сървис

Регистриране

Име

Фамилия

Имейл

Телефонен номер

Парола

Потвърди паролата

Град

Veliko Tarnovo

Училище

PMG Vasil Drumev

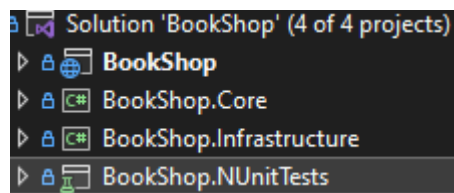
Регистрирай се

Фиг. 5: Формуляр за регистрация

4.3.4 Проверки на кода и подобряване на архитектурата на прокта

За да гарантирам сигурност и издръжливост на проекта, създадох тестове, които проверят за правилното изпълнение на кода. Основната цел на тези тестове е да тестват част от кода на приложението без то е зависимо от външни фактори. Пример за важността на този метод е тестване на контролер, който изпълнява задачите си правилно, но поради неправилна база данни, дава грешни резултати при тестването му.

Има много технологии и начини да се постигне този вид на тестване, но аз използвах [NUnits Tests](#) и [Moq](#), защото са добре разработени и са доста популярни. За тяхната имплементация е нужно да се създаде нов проект в директорията на "BookShop". Заради това трябваше да разделя проекта на четири част като всяка от тях отговаря за различни дейности от общия проект 6. BookShop отговаря за визуализацията в уеб среда, включително и контролерите, и стартирането на проекта. BookShop.Core поддържа всичката бизнес логика на проекта, която е разпределена между сървисите. BookShop.Infrastructure поема връзката с базата данни и общи константи между другите части. BookShop.NUnitTests се поема тестването на кода и проверките за неговото правилно изпълнение. Тази част от проекта не се стартира с другите части. Чрез тестове проверявам само бизнес логика, написана от мен, защото там шансът за провал е най-голям. За да мога да тествам сложни операции свързани с базата данни, трябва да използвам мокване на базата данни. Това се имплементира с допълнителната библиотека Moq. Чрез нея създавам временна база данни, записана на RAM паметта на компютъра ми. Така си гарантирам, че напълно изолирам тествани части от външни фактори. След създаването на тази мини база данни е лесно да се провери изпълнението на части от кода, чрез NUnit Tests.



Фиг. 6: Структурата на проекта

```

[SetUp]
0 references
public void InitializeDb()
{
    var options = new DbContextOptionsBuilder<ApplicationDbContext>()
        .UseInMemoryDatabase(databaseName: "BooksInMemoryDb")
        .Options;

    context = new ApplicationDbContext(options);

    context.Database.EnsureDeleted();

    List<User> users = new List<User>();

    context.Users.AddRange(users);

    context.Roles.Add(new IdentityRole() { Id = "971ba58d-3ed5-4958-95b6-5e96a734db6f", Name = "Admin" });

    context.SaveChanges();

    statisticsService = new StatisticsService(context);
    adminService = new AdminService(context);
}

[Test]
0 references
public void Test_BookCount()

[Test]
0 references
public void Test_UsersCount()

[Test]
0 references
public async Task Test_AddUserToRole_CorrectInput()

[Test]
0 references
public async Task Test_AddUserToRole_InvalidInput()

```

Фиг. 7: Мокване на БД и тестване

4.3.5 Съставяне на придружаващите документи

Тази част от наравата на проекта включва създаването на презентация, която описва накратко написаното тук, и писането на този документ.

4.4 Ниво на сложност на проекта

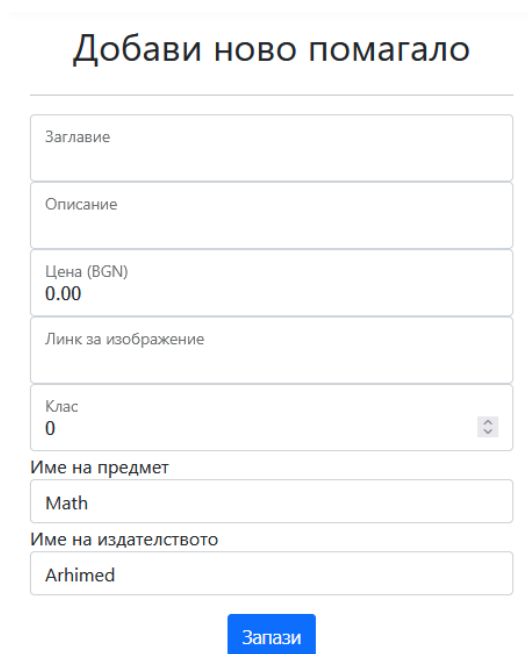
Докато работих по този проект научих множество нови неща и усъвършенствах уменията свързани с компютърните науки. Най-предизвикателните части от създаването на проекта бяха мокването и свързването на проекта с база данни. Един специфичен проблем, който ме затрудни беше правилното използване на методите [SetUp] и [OneTimeSetUp]. Грешките възникнали от неправилното им ползване не позволяваха да се изпълнят всички тестове правилно, въпреки че тестваният код, работеше правилно.

5 Логическо и функционално описание на решението

Тук ще бъдат разгледани отделните части от проекта и техните специалности.

5.1 Учебни Помагала

Всяко учебно помагало, публикувано в проекта има няколко показателя⁸. Всички освен описанието на учебника са задължителни като има още два, които се въвеждат автоматично. Те са датата на публикуване и собственика му.



Добави ново помагало

Заглавие

Описание

Цена (BGN)
0.00

Линк за изображение

Клас
0

Име на предмет
Math

Име на издателството
Arhimed

Запази

Фиг. 8: Формуляр за публикуване на помагало

5.2 Потребители

Потребители на сайта, които са си влезли в акаунти, получават много повече от тези, които не са. Те могат четат информация за собствениците на учебници ⁹, могат да публикуват своите книги за продан и получават достъп по страница в уеб сайта, където могат да разгледат, редактират или изтрият качените от тях помагала¹⁰. Потребителите могат да редактират своя акаунт ¹¹ и имат възможност да станат администратор срещу предоставяне на телефонен номер ¹².

Име	Ivallo
Фамилия	Nedev
Имейл	ovinedev@abv.bg
Телефонен номер	085778098
Town	Veliko Tarnovo
	Veliko Tarnovo
School	PMG Vasil Drumev
	PMG Vasil Drumev
Запази	

Фиг. 11: Страница за редактиране на профила

Стани админ

Въведи телефония си номер

Изпрати

Фиг. 12: Формуляр за записване за администратор

5.3 Администратори

Администраторите имат правата да създават и трият градове, училища и издателства [13](#). Също така могат да виждат общия брой потребители и учебници в сайта, но не могат да публикуват книги за продажба и е задължително да нямаш публикувани книги, за да бъдеш приет като админ.

The figure consists of three side-by-side screenshots of an administrative web application interface. Each screenshot shows a form at the top for adding a new entity and a list below for all existing entities. (a) 'Добави град' (Add City) form with a 'Name' field and an 'Add' button. Below it, a list of cities: Veliko Tarnovo, Sofia, Varna, and Other, each with a red 'Изтрий' (Delete) button. (b) 'Добави издателство' (Add Publisher) form with a 'Name' field and an 'Add' button. Below it, a list of publishers: Ashamed, Prosveta, Cengage Learning, and Longman, each with a red 'Изтрий' (Delete) button. (c) 'Добави училище' (Add School) form with fields for 'Name', 'SchoolType' (PrimarySchool), 'Town' (Veliko Tarnovo), and an 'Add' button. Below it, a list of schools: PMG Vasil Drumev, Anton Strashimirov, SU Emilian Stanev, and Other, each with a red 'Изтрий' (Delete) button.

Фиг. 13: Правата на администраторите върху сайта

5.4 Сигурност и валидация на входните данни

Всички данни, които биват въведени от потребителят, биват филтрирани през няколко валидации. Първата е JavaScript. Тя много бързо проверява данните, но не достатъчно силна и може лесно да бъде спряна. Затова данните се проверяват още един път в контролерите или сървисите преди да бъдат качени в базата данни. Имплементирах с HtmlSanitizer, което блокира чужд код да бъде качен в базата, чрез потребител на сайта [14](#).

```
Book book = new Book()
{
    Title = htmlSanitizer.Sanitize(model.Title),
    Description = htmlSanitizer.Sanitize(model.Description),
    Price = model.Price,
    BookTypeId = model.SubjectId,
    PublisherId = model.PublisherId,
    Grade = model.Grade,
    imageUrl = model.imageUrl,
    datePublished = DateTime.Today,
    OwnerId = userId,
    Owner = owner,
    IsDeleted = false
};
```

Фиг. 14: HtmlSanitizer спира записването на код в полета за заглавие и описание на книга

6 Технологични средства

- Visual Studio 2022 – реализация на проекта;
- SQL Server Management Studio – имплементиране на база данни;
- PowerPoint 2016 – създаване на презентация;
- Overleaf - написване на документация;
- Slidesgo –взимане на шаблон за презентация;
- GitHub – поддържане на проекта;
- NUnit Tests – създаване на тестове за проверка на кода;
- Moq – мокване на тестове;
- HtmlSanitizer – защита срещу хакерски атаки.

7 Използвана литература

- [документация на Overleaf](#)
- [Stack Overflow](#)
- [документацията на .NET](#)

8 Заключение

Проектът „BookShop“ дава възможност за лесен обмен на учебни помаги между ученици без намеса на външни лица в трансакцията.