

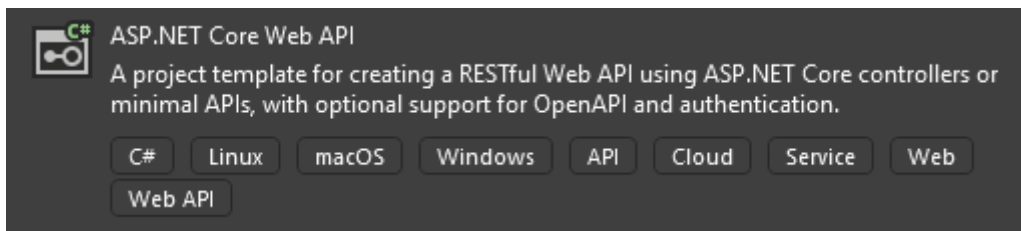
# Лабораторна робота №4

## Розробка вебсерверів засобами фреймворку ASP.NET.

Виконала студентка групи 302-ТК

Варинська Євгенія

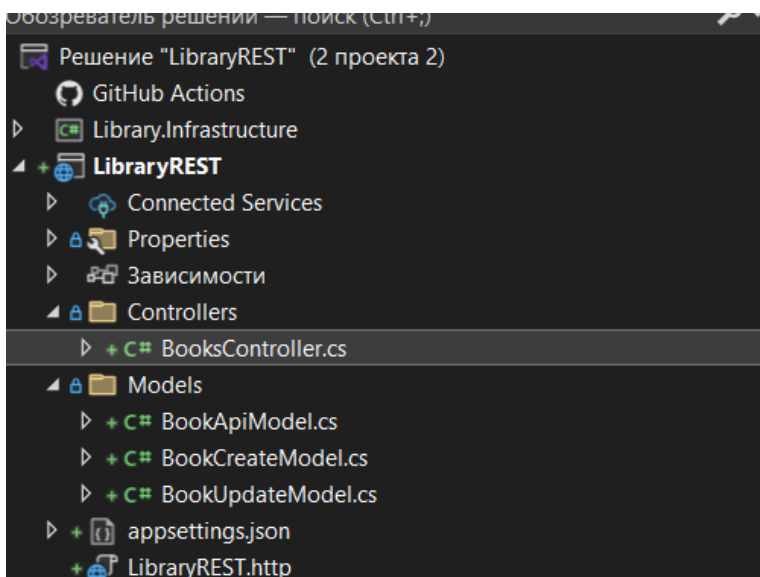
1. Створити проєкт за шаблоном ASP.NET Core Web API {Назва тематики}.REST використовуючи Visual Studio або .NET SDK: `dotnet new webapi -n {Назва тематики}.REST`



2. Створити папку Models, у якій будуть зберігатися моделі, які будуть використовуватися контролерами у проєкті {Назва тематики}.REST.

3. Створіть контролери у папці Controllers, які будуть описувати REST API для виконання CRUD операцій (створення, читання, оновлення, видалення) над моделями у вашій тематиці, які ви зберігали у базі даних у третій лабораторній, при цьому моделі, що використовуються у контролерах, повинні бути описані у проєкті {Назва тематики}.REST, щоб відповідати 3-рівневій архітектурі. Наприклад, якщо у вас є сутність Bus, що зберігається у БД, ви створите новий клас BusModel у папці Models нового проєкту, який використовуватиметься у контролерах. Також зверніть увагу, щоб моделі у контролерах, мали лишень необхідні властивості, та якщо якісь властивості зайві, для якихось із методів - створіть нову модель без цих властивостей.

### Архітектура:



## Контролер:

```
BookUpdateModel.cs*  BookCreateModel.cs*  BooksController.cs*  BookApiModel.cs*
LibraryREST
LibraryREST.Controllers.BooksController

1  using Microsoft.AspNetCore.Mvc;
2  using Library.Infrastructure.Models;
3  using LibraryREST.Models;
4
5  namespace LibraryREST.Controllers
6  {
7      [ApiController]
8      [Route("api/[controller]")]
9      public class BooksController : ControllerBase
10     {
11         private static List<BookModel> _books = new()
12         {
13             new BookModel
14             {
15                 Id = 1,
16                 Title = "1984",
17                 Author = "George Orwell",
18                 LibraryMemberId = 1,
19                 BookTags = new List<BookTagModel>()
20             },
21             new BookModel
22             {
23                 Id = 2,
24                 Title = "The Hobbit",
25                 Author = "J.R.R. Tolkien",
26                 LibraryMemberId = 2,
27                 BookTags = new List<BookTagModel>()
28             }
29         };
30
31         // GET: api/Books
32         [HttpGet]
33         public IActionResult GetAll()
34         {
35             var result = _books.Select(book => new BookApiModel
36             {
37                 Id = book.Id,
38                 Title = book.Title,
39                 Author = book.Author
40             });
41
42             return Ok(result);
43         }
44
45         // GET: api/Books/1
46         [HttpGet("{id}")]
47         public IActionResult Get(int id)
48         {
49             var book = _books.FirstOrDefault(b => b.Id == id);
```

```
BookUpdateModel.cs* BookCreateModel.cs* BooksController.cs* BookApiMo
LibraryREST LibraryREST.Controllers.BooksContro
52 var dto = new BookApiModel
53 {
54     Id = book.Id,
55     Title = book.Title,
56     Author = book.Author
57 };
58
59 return Ok(dto);
60 }
61
62 // POST: api/Books
63 [HttpPost]
64 Ссылка: 0
65 public IActionResult Create(BookCreateModel model)
66 {
67     var newBook = new BookModel
68     {
69         Id = _books.Any() ? _books.Max(b => b.Id) + 1 : 1,
70         Title = model.Title,
71         Author = model.Author,
72         LibraryMemberId = model.LibraryMemberId,
73         BookTags = new List<BookTagModel>()
74     };
75     _books.Add(newBook);
76     return CreatedAtAction(nameof(Get), new { id = newBook.Id }, model);
77 }
78
79 // PUT: api/Books/1
80 [HttpPut("{id}")]
81 Ссылка: 0
82 public IActionResult Update(int id, BookUpdateModel model)
83 {
84     var book = _books.FirstOrDefault(b => b.Id == id);
85     if (book == null) return NotFound();
86
87     book.Title = model.Title;
88     book.Author = model.Author;
89     book.LibraryMemberId = model.LibraryMemberId;
90
91     return NoContent();
92 }
93
94 // DELETE: api/Books/1
95 [HttpDelete("{id}")]
96 Ссылка: 0
97 public IActionResult Delete(int id)
98 {
99     var book = _books.FirstOrDefault(b => b.Id == id);
100     if (book == null) return NotFound();
101
102     _books.Remove(book);
103     return NoContent();
104 }
```

4. При проєктуванні Web API зверніть увагу, щоб створений API відповідав 4-ій умові побудови REST-додатку по Філдингу - однорідності інтерфейсу (див. 31 слайд), тобто використовувалися унікальні назви сутностей, правильні HTTP методи та поверталися правильні HTTP коди у HTTP відповідях, наприклад 404 якщо сутність не знайдена, або 201 якщо нова сутність створена.

### HTTP методи:

- GET — для отримання (список і по ID).
- POST — для створення.
- PUT — для оновлення.
- DELETE — для видалення.

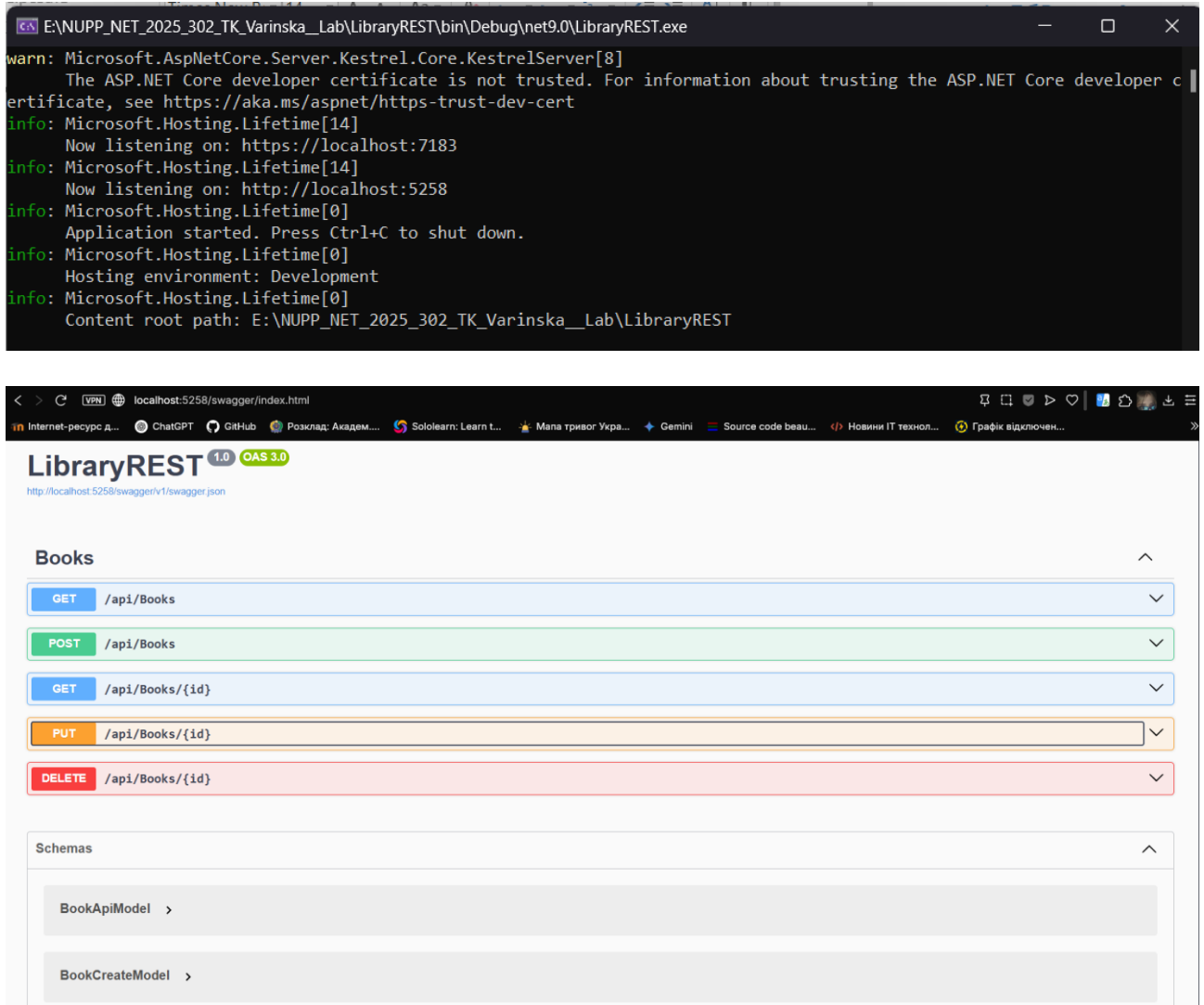
### HTTP статуси:

- 404 Not Found — коли книга не знайдена.
- 201 Created — коли створено нову книгу.
- 204 No Content — після оновлення або видалення без тіла відповіді.

### Унікальні URI для ресурсів:

- api/books — колекція.
- api/books/{id} — конкретний ресурс.

### Результат:



5. Використовуйте асинхронну версію дженерік CRUD сервісу, що використовує репозиторій для доступу до даних та який був реалізований у 3ій лабораторній роботі, у створених контролерах.:

```
public interface ICrudServiceAsync<T>
{
    public Task<bool> CreateAsync(T element);
    public Task<T> ReadAsync(Guid id);
    public Task<IEnumerable<T>> ReadAllAsync();
}
```

```

public Task<IEnumerable<T>> ReadAllAsync(int page, int amount);
public Task<bool> UpdateAsync(T element);
public Task<bool> RemoveAsync(T element);
public Task<bool> SaveAsync();
}

```

Імплементацію CRUD сервісу, репозиторіїв та контексту, додавайте у контролери використовуючи вбудовану у ASP.NET Core підтримку впровадження залежностей(Dependency Injection).

Інтерфейс:

```

CrudServiceAsync.cs  Program.cs*
C# Library.Infrastructure
1  using Library.Infrastructure.Repositories;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6
7  namespace Library.Infrastructure.Services
8  {
9      Ссылка 3
10     public class CrudServiceAsync<T> : ICrudServiceAsync<T> where T : class
11     {
12         private readonly IRepository<T> _repository;
13
14         Ссылка 0
15         public CrudServiceAsync(IRepository<T> repository)
16         {
17             _repository = repository;
18         }
19
20         Ссылка 2
21         public async Task<bool> CreateAsync(T element)
22         {
23             await _repository.AddAsync(element);
24             await _repository.SaveChangesAsync();
25             return true;
26         }
27
28         Ссылка 4
29         public async Task<T?> ReadAsync(int id)
30         {
31             var all = await _repository.GetAllAsync();
32             return all.FirstOrDefault(e => (e as dynamic).Id == id);
33         }
34
35         Ссылка 2
36         public async Task<IEnumerable<T>> ReadAllAsync()
37         {
38             return await _repository.GetAllAsync();
39         }
40
41         Ссылка 1
42         public async Task<IEnumerable<T>> ReadAllAsync(int page, int amount)
43         {
44             var all = await _repository.GetAllAsync();
45             return all.Skip((page - 1) * amount).Take(amount);
46         }
47
48         Ссылка 2
49         public async Task<bool> UpdateAsync(T element)
50         {
51             await _repository.UpdateAsync(element);
52             await _repository.SaveChangesAsync();
53             return true;
54         }
55
56         Ссылка 2
57         public async Task<bool> RemoveAsync(T element)
58         {
59             await _repository.DeleteAsync(element);
60             await _repository.SaveChangesAsync();
61             return true;
62         }
63
64         Ссылка 4
65         public async Task<bool> SaveAsync()
66         {
67             await _repository.SaveChangesAsync();
68             return true;
69         }
70     }
71 }

```

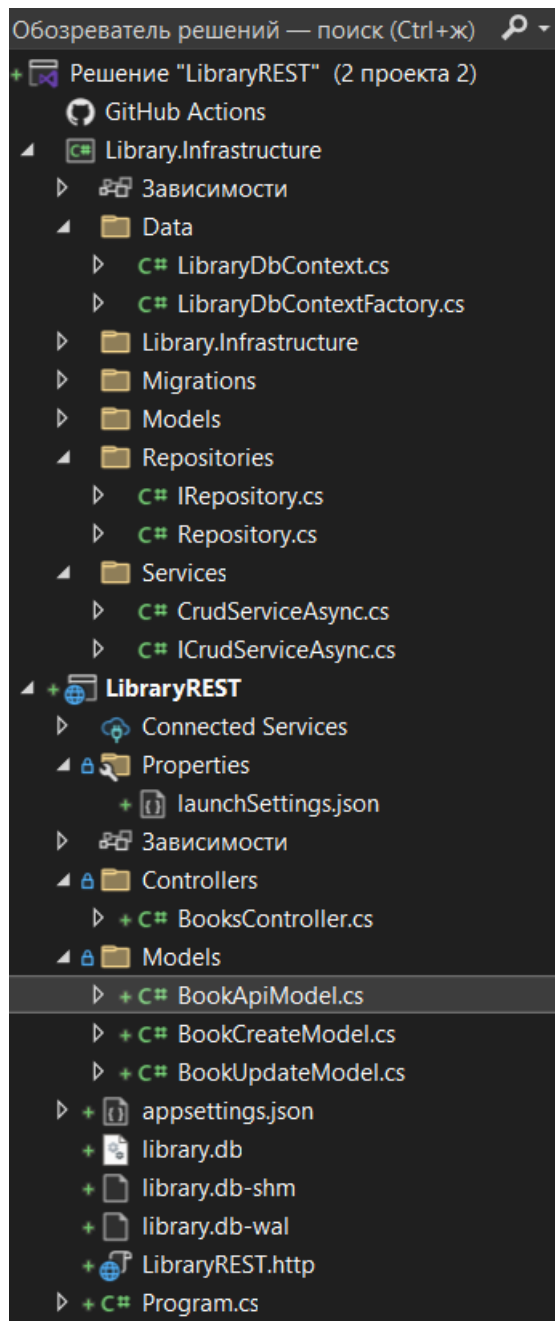
## Контролер:

```
BooksController.cs  Program.cs*
LibraryREST LibraryREST.Controllers.BooksController

4  using LibraryREST.Models;
5
6  namespace LibraryREST.Controllers
7  {
8      [ApiController]
9      [Route("api/[controller]")]
10     public class BooksController : ControllerBase
11     {
12         private readonly ICrudServiceAsync<BookModel> _bookService;
13
14         public BooksController(ICrudServiceAsync<BookModel> bookService)
15         {
16             _bookService = bookService;
17         }
18
19         // GET: api/Books
20         [HttpGet]
21         public async Task<ActionResult<IEnumerable<BookApiModel>>> GetAll()
22         {
23             var books = await _bookService.ReadAllAsync();
24             var result = books.Select(book => new BookApiModel
25             {
26                 Id = book.Id,
27                 Title = book.Title,
28                 Author = book.Author
29             });
30             return Ok(result);
31         }
32
33         // GET: api/Books/1
34         [HttpGet("{id:int}")]
35         public async Task<ActionResult<BookApiModel>> Get(int id)
36         {
37             var book = await _bookService.ReadAsync(id);
38             if (book == null)
39                 return NotFound();
40
41             var dto = new BookApiModel
42             {
43                 Id = book.Id,
44                 Title = book.Title,
45                 Author = book.Author
46             };
47             return Ok(dto);
48         }
49
50         // POST: api/Books
51         [HttpPost]
52         public async Task<ActionResult> Create([FromBody] BookCreateModel model)
53         {
54             var newBook = new BookModel
55         }
```

```
BooksController.cs Program.cs*
LibraryREST LibraryREST.Controllers.BooksCont

51
52 // POST: api/Books
53 [HttpPost]
54 Ссылка: 0
55 public async Task<IActionResult> Create([FromBody] BookCreateModel model)
56 {
57     var newBook = new BookModel
58     {
59         Title = model.Title,
60         Author = model.Author,
61         LibraryMemberId = model.LibraryMemberId,
62         BookTags = new List<BookTagModel>()
63     };
64     var created = await _bookService.CreateAsync(newBook);
65     if (!created) return BadRequest("Could not create book.");
66     await _bookService.SaveAsync();
67     return CreatedAtAction(nameof(Get), new { id = newBook.Id }, newBook);
68 }
69
70 // PUT: api/Books/1
71 [HttpPut("{id:int}")]
72 Ссылка: 0
73 public async Task<IActionResult> Update(int id, [FromBody] BookUpdateModel model)
74 {
75     var book = await _bookService.ReadAsync(id);
76     if (book == null)
77         return NotFound();
78     book.Title = model.Title;
79     book.Author = model.Author;
80     book.LibraryMemberId = model.LibraryMemberId;
81     var updated = await _bookService.UpdateAsync(book);
82     if (!updated) return BadRequest("Update failed.");
83     await _bookService.SaveAsync();
84     return NoContent();
85 }
86
87 // DELETE: api/Books/1
88 [HttpDelete("{id:int}")]
89 Ссылка: 0
90 public async Task<IActionResult> Delete(int id)
91 {
92     var book = await _bookService.ReadAsync(id);
93     if (book == null)
94         return NotFound();
95     var deleted = await _bookService.RemoveAsync(book);
96     if (!deleted) return BadRequest("Delete failed.");
97     await _bookService.SaveAsync();
98     return NoContent();
99 }
100
101
102
103
104
105
```



Код програми:



```
Program.cs*  X
LibraryREST
4  using Library.Infrastructure.Services;
5  using Microsoft.EntityFrameworkCore;
6
7  var builder = WebApplication.CreateBuilder(args);
8
9  builder.Services.AddDbContext<LibraryDbContext>(options =>
10     options.UseSqlite("Data Source=library.db"));
11
12  builder.Services.AddScoped<IRepository<BookModel>, Repository<BookModel>>();
13  builder.Services.AddScoped<ICrudServiceAsync<BookModel>, CrudServiceAsync<BookModel>>();
14
15  builder.Services.AddScoped<IRepository<LibraryMemberModel>, Repository<LibraryMemberModel>>();
16  builder.Services.AddScoped<ICrudServiceAsync<LibraryMemberModel>, CrudServiceAsync<LibraryMemberModel>>();
17
18  builder.Services.AddControllers();
19  builder.Services.AddEndpointsApiExplorer();
20  builder.Services.AddSwaggerGen();
21
22  var app = builder.Build();
23
24  app.UseSwagger();
25  app.UseSwaggerUI();
26
27  app.UseHttpsRedirection();
28  app.UseAuthorization();
29
30  app.MapControllers();
31
32  using (var scope = app.Services.CreateScope())
33  {
34      var db = scope.ServiceProvider.GetRequiredService<LibraryDbContext>();
35
36      db.Database.Migrate();
37
38      if (!db.LibraryMembers.Any())
39      {
40          var testMember = new LibraryMemberModel
41          {
42              FullName = "Тестовий Користувач",
43              Email = "test@example.com"
44          };
45          db.LibraryMembers.Add(testMember);
46          db.SaveChanges();
47
48          var testBook = new BookModel
49          {
50              Title = "Тестова книга",
51              Author = "Тест Автор",
52              LibraryMemberId = testMember.Id,
53              BookTags = new List<BookTagModel>()
54          };
55          db.Books.Add(testBook);
56          db.SaveChanges();
57      }
58  }
59
60  app.Run();
```

Результат Swagger:

```
{
  "title": "Тестова книга",
  "author": "Тест Автор",
  "libraryMemberId": 1
}
```

Execute

Clear

## Responses

### Curl

```
curl -X 'POST' \
  'http://localhost:5258/api/Books' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "title": "Тестова книга",
    "author": "Тест Автор",
    "libraryMemberId": 1
  }'
```

### Request URL

http://localhost:5258/api/Books

### Server response

Code Details

201

Undocumented

### Response body

```
{
  "id": 2,
  "title": "Тестова книга",
  "author": "Тест Автор",
  "libraryMemberId": 1,
  "bookTags": []
}
```

### Response headers

```
content-type: application/json; charset=utf-8
date: Tue, 13 May 2025 15:53:24 GMT
location: http://localhost:5258/api/Books/2
server: Kestrel
transfer-encoding: chunked
```

GET /api/Books/{id}

## Parameters

Cancel

Name Description

id \* required

integer(\$int32)  
(path)

1

Execute

Clear

## Responses

### Curl

```
curl -X 'GET' \
  'http://localhost:5258/api/Books/1' \
  -H 'accept: text/plain'
```

### Request URL

http://localhost:5258/api/Books/1

### Server response

Code Details

200

### Response body

```
{
  "id": 1,
  "title": "Тестова книга",
  "author": "Тест Автор"
}
```

### Response headers

```
content-type: application/json; charset=utf-8
date: Tue, 13 May 2025 15:53:31 GMT
server: Kestrel
transfer-encoding: chunked
```

## Responses

Code Description

Links

200

OK

No links

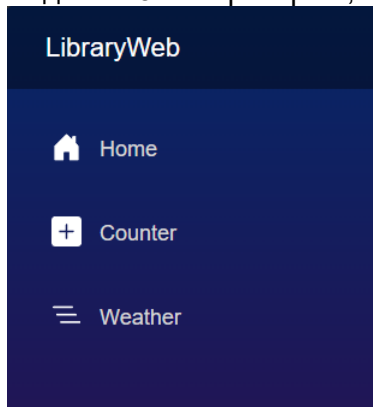
Media type

text/plain

Controls Accept header.

## Додаткове завдання

Створіть проєкт {Назва тематики}.MVC за шаблоном Blazor Web App або [ASP.NET MVC](#). У цьому проєкті реалізуйте вебзастосунок, що використовує контекст БД та моделі із 3 лабораторної, використовуючи генерацію сторінок вбудовану у ASP.NET.



## Books

- Тестова книга by Тест Автор
- Тестова книга by Тест Автор