Міністерство освіти і науки України
Національний університет водного господарства та природокористування

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра обчислювальної техніки

04-04-295М

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з навчальної дисципліни
«Інженерія програмного забезпечення» (частина 1)
для здобувачів вищої освіти
першого (бакалаврського) рівня
за освітньо-професійною програмою
«Комп'ютерна інженерія»
спеціальності 123 «Комп'ютерна інженерія»
та освітньо-професійною програмою
«Інформаційна безпека»
спеціальності 125 «Кібербезпека та захист інформації»
денної та заочної форми навчання

Рекомендовано
науково-методичною радою
з якості ННІКІТІ
Протокол № 4 від 24.02.2025 р.

Рівне – 2025

Методичні вказівки до лабораторних робіт з навчальної дисципліни «Інженерія програмного забезпечення» (частина 1) для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Комп'ютерна інженерія» спеціальності 123 «Комп'ютерна інженерія» та освітньо-професійною програмою «Інформаційна безпека» спеціальності 125 «Кібербезпека та захист інформації» денної та заочної форми навчання. [Електронне видання] / Бойчура М. В. – Рівне : НУВГП, 2025. – 30 с.

Укладач: Бойчура М. В., к.т.н., доцент кафедри обчислювальної техніки.

Відповідальний за випуск: Сидор А. І., к.т.н., в.о. завідувача кафедри обчислювальної техніки.

Керівник групи забезпечення
спеціальності
123 «Комп'ютерна інженерія» Сидор А. І.

Керівник групи забезпечення
спеціальності
125 «Кібербезпека та захист інформації» Назарук В. Д.

© М. В. Бойчура, 2025
© НУВГП, 2025

ЗМІСТ

Вступ

Навички в побудові стабільного і розширюваного програмного забезпечення є дуже важливою, але й одночасно складною задачею. Використання архітектурних рішень виду Layer Architecture чи Clean Architecture є ключовими елементами у розробці сучасних додатків. Додаткова інтеграція патернів виду MVC, Repository, Dependency Injection сприяє високому рівню модульності програмного забезпечення. Дуже важливо, щоб додатки базувались на асинхронності. Існує й низка інших важливих характеристик якісного програмного продукту.

У межах першого модулю дисципліни Інженерія програмного забезпечення передбачається вивчення самих базових тем фреймворку ASP.NET Core. Це включає CRUD-операції, що доповнюється навичками використання патернів Dependency Injection, MVC та архітектури Layer Architecture.

Технологія ASP.NET, як відомо передбачає використання мов C#, HTML та JavaScript, таблиць CSS, баз даних, що структурує, узагальнює та доповнює знання, отримані на попередньому році навчання.

Особлива увага приділяється розвитку навичок командної роботи та комунікації англійською мовою. Зокрема студенти вивчають командну розробку з використанням GitHub та Jira. Методичні вказівки орієнтовані на формування практичних компетностей у веб-розробці, що відповідають потребам сучасного ринку праці. Дисципліна спрямована на всебічну підготовку майбутніх фахівців з розробки програмного забезпечення.

Даний курс охоплює такі ключові аспекти:

- вивчення фреймворку ASP.NET Core;
- освоєння патерну MVC;

- реалізацію CRUD-операцій;
- побудову пошарової архітектури програмного забезпечення;
- організацію командної роботи з використанням GitHub та Jira.

Результати навчання включають:

- вміння створювати веб-додатки на основі ASP.NET Core MVC;
- навички виконання операцій з даними;
- знання інструментів стилізації веб-сторінок, таких як фреймворк Bootstrap;
- компетності ефективної співпраці у командах із застосуванням GitHub і Jira.

Методичні матеріали створені з урахуванням рекомендацій IT-компанії SoftServe.

Лабораторна робота №1
Вступ. Огляд шаблону ASP.NET Core MVC

Цілі заняття

- Навчитись працювати з основними файлами проєкту ASP.NET Core MVC.
- Навчитись застосувати фреймворк Bootstrap для оформлення веб-сторінок.

Постановки завдань

- **40% балів:** інсталювати всі необхідні компоненти для роботи з ASP.NET Core MVC.
- **5% балів:** створити базовий проєкт із вбудованою автентифікацією.
- **35% балів:** модифікувати наповнення header, content і footer у проєкті згідно варіанту.
- **20% балів:** додати елементи Bootstrap, зокрема кнопку, картки та таблицю згідно варіанту.

Варіанти до виконання поставленого завдання

1. **Сторінка волонтерської організації.** Перелік членів волонтерської організації з фотографіями та короткими описами. Таблиця з вказаною зоною відповідальності кожного волонтера.
2. **Сторінка музею.** Перелік експонатів із фотографіями та короткими описами. Таблиця з датами та місцями проведення виставок.
3. **Сторінка кіноклубу.** Плитки з фільмами, їх постерами та коротким описом. Таблиця з розкладом сеансів.
4. **Сторінка книжкового клубу.** Плитки з книжками (фото обкладинок, короткий опис). Таблиця рекомендованої літератури з жанрами та авторами.

5. **Сторінка спортивного клубу.** Плитки з тренерами (фото, спеціалізація). Таблиця з розкладом занять.

6. **Сторінка кав'ярні.** Меню у вигляді плиток із фотографіями страв. Таблиця з описом акцій та знижок.

7. **Сторінка театру.** Плитки з афішами вистав. Таблиця з цінами на квитки та місцями.

8. **Сторінка туристичної компанії.** Плитки з туристичними маршрутами (фото, короткий опис). Таблиця з цінами на тури.

9. **Сторінка художньої галереї.** Плитки з картинами (фото, ім'я автора). Таблиця з датами майбутніх виставок.

10. **Сторінка музичної школи.** Плитки з викладачами (фото, спеціалізація). Таблиця з розкладом занять.

11. **Сторінка бібліотеки.** Плитки з книгами (фото обкладинок, короткий опис). Таблиця з інформацією про доступні формати (паперові, електронні).

12. **Сторінка ветклініки.** Плитки з послугами (фото, короткий опис). Таблиця з цінами та годинами роботи.

13. **Сторінка автомийки.** Плитки з видами послуг (фото, опис). Таблиця з цінами та додатковими опціями.

14. **Сторінка фотостудії.** Плитки з прикладами робіт (фото). Таблиця з описом пакетів послуг.

15. **Сторінка університету.** Плитки з кафедрами (фото, короткий опис). Таблиця з переліком спеціальностей.

16. **Сторінка зоомагазину.** Плитки з товарами для тварин (фото, опис). Таблиця з цінами та доступністю в магазинах.

17. **Сторінка готелю.** Плитки з номерами (фото, опис). Таблиця з цінами та додатковими послугами.

18. **Сторінка меблевого магазину.** Плитки з меблями (фото, опис). Таблиця з наявністю на складі та термінами доставки.

19. **Сторінка курсу іноземних мов.** Плитки з викладачами (фото, мова викладання). Таблиця з розкладом занять.

20. **Сторінка концертної агенції.** Плитки з майбутніми подіями (афіші). Таблиця з цінами на квитки.

21. **Сторінка коворкінгу.** Плитки з фотографіями приміщень. Таблиця з цінами на оренду та додатковими послугами.

22. **Сторінка компанії з ремонту техніки.** Плитки з видами послуг (фото, опис). Таблиця з орієнтовними цінами.

23. **Сторінка благодійного фонду.** Плитки з проєктами (фото, опис). Таблиця з контактами організаторів.

24. **Сторінка спортивної команди.** Плитки з гравцями (фото, роль). Таблиця з розкладом матчів.

25. **Сторінка школи мистецтв.** Плитки з напрямками навчання (фото, опис). Таблиця з розкладом занять.

26. **Сторінка весільного агентства.** Плитки з послугами (фото, опис). Таблиця з цінами на пакети.

27. **Сторінка магазину техніки.** Плитки з товарами (фото, опис). Таблиця зі знижками та акціями.

28. **Сторінка клініки естетичної медицини.** Плитки з процедурами (фото, опис). Таблиця з цінами та тривалістю процедур.

29. **Сторінка команди розробників.** Плитки з членами команди (фото, ролі). Таблиця з проєктами та їх статусами.

30. **Сторінка фермерського господарства.** Плитки з продукцією (фото, опис). Таблиця з цінами та місцями доставки.

Рекомендації

1. Для навчальних цілей достатньо використовувати безкоштовну версію програмного продукту Microsoft Visual

Studio 2022 Community Edition (рис. 1,*а*). Її можна завантажити з офіційного сайту Microsoft. У якості «легшої» альтернативи дозволяється використання Rider для Full-Stack веб-розробки (рис. 1,*б*). Дане програмне забезпечення можна завантажити з офіційного сайту Jetbrains.

а)

б)

Рис. 1. Логотипи: Microsoft Visual Studio (*а*), Jetbrains Rider (*б*)

2. Для роботи з проєктом, у випадку Microsoft Visual Studio, переконайтеся, що інстальовано компонент *.Net Desktop Development* (рис. 2).

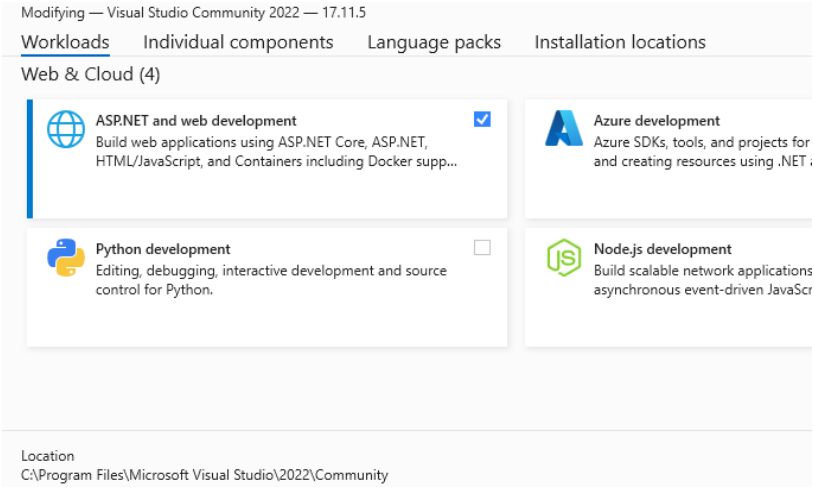

Рис. 2. Вікно Microsoft Visual Studio Installer; обрано компонент, який рекомендується встановити

3. Під час створення проєкту знайдіть шаблон ASP.NET Core Web App (Model-View-Controller). Оберіть параметр *Authentication Type: Individual Accounts* (рис. 3).

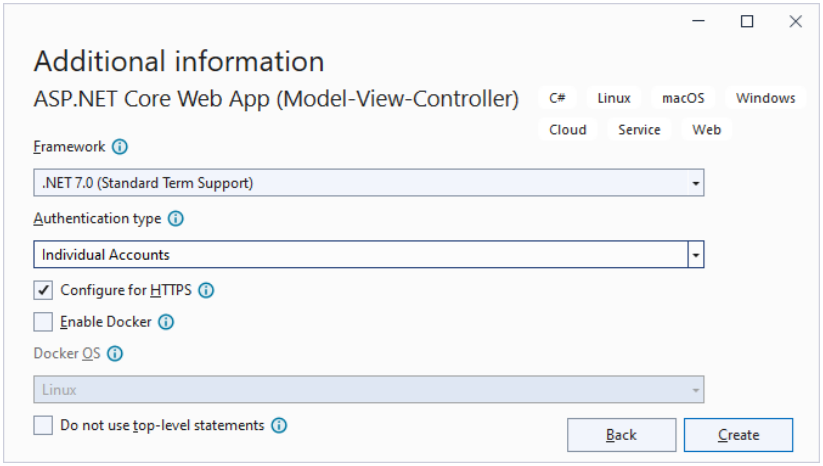

Рис. 3. Рекомендовані налаштування при створенні проєкту

4. Додайте Bootstrap-компоненти, використовуючи офіційний веб-сайт фреймворку (рис. 4).

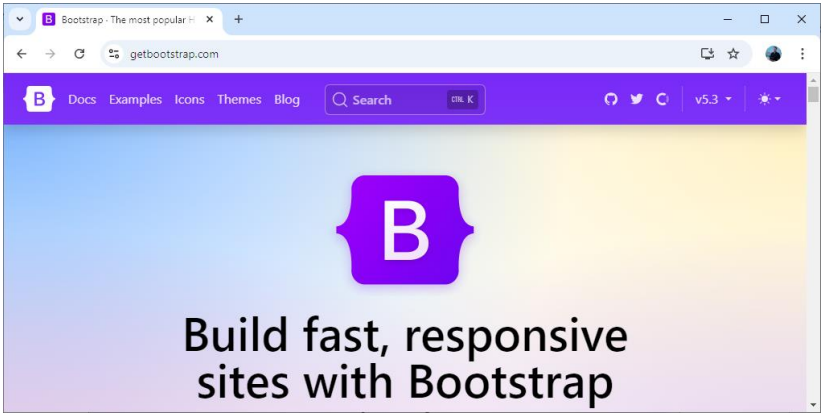

Рис. 4. Вітальна сторінка веб-сайту фреймворку Bootstrap

Необхідні веб-ресурси

- https://getbootstrap.com
- https://www.jetbrains.com/ru-ru/lp/rider-web/
- https://visualstudio.microsoft.com/vs/community/

Контрольні запитання

1. Що таке ASP?
2. Що таке ASP.NET Core MVC?
3. Що означає абревіатура MVC?
4. Як додати автентифікацію в проєкт ASP.NET Core MVC?
5. Яким чином доцільно використовувати Bootstrap у веб-проєктах?
6. Які компоненти необхідно встановити для роботи з ASP.NET Core MVC у Microsoft Visual Studio 2022?
7. Які переваги використання шаблону MVC?
8. Як створити новий проєкт ASP.NET Core MVC?
9. Що таке файл _Layout.cshtml і яка його роль у проєкті?
10. Як підключити фреймворк Bootstrap у проєкт ASP.NET Core MVC?
11. Які особливості шаблону *Authentication Type: Individual Accounts*?
12. Які основні файли та папки з'являються при створенні проєкту ASP.NET Core MVC?
13. Як налаштувати SSL-сертифікати для ASP.NET Core?
14. Яка структура базового шаблону ASP.NET Core MVC?
15. Як змінити вигляд веб-сторінок у проєкті?
16. Як створити кнопку з використанням Bootstrap?
17. Як перевірити версію ASP.NET Core у проєкті?

18. Які розширення файлів використовуються для представлення (view) у MVC?

19. Як встановити залежності для проєкту за допомогою NuGet?

20. Як додати картку на сторінку за допомогою Bootstrap?

21. Які є способи відлагодження коду в ASP.NET Core MVC?

22. Яке призначення файлу appsettings.json?

23. Як створити адаптивний дизайн сторінки з використанням Bootstrap?

24. Як перевірити, чи коректно працює фреймворк Bootstrap у проєкті?

Лабораторна робота №2
Патерн MVC

Мета заняття

- Розібратись у принципах патерну Model-View-Controller.
- Навчитись створювати моделі, контролери та представлення для обміну даних між ними.

Постановки завдань

- **5% балів:** додати новий пункт у навігаційне меню.
- **20% балів:** створити контролер і представлення для цього пункту меню.
- **15% балів:** передати числове значення або рядок з контролера у представлення.
- **20% балів:** створити модель для представлення згідно обраного варіанту.
- **15% балів:** передати модель від контролера до представлення.
- **25% балів:** реалізувати передачу даних з представлення у контролер.

Рекомендації

1. Дослідіть структуру (рис. 5) патерна MVC (модель – представлення – контролер). Зверніть увагу, що модель – це тип даних, який використовується для передачі інформації між контролером і представленням.

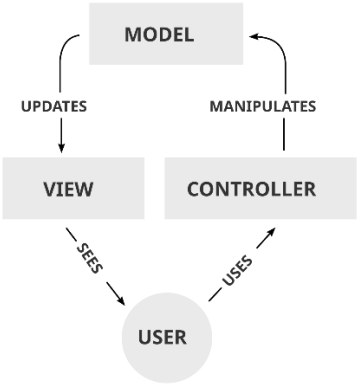

Рис. 5. Схематичне зображення патерна MVC

2. Створіть новий пункт навігаційного меню, відредагувавши файл _Layout.cshtml (рис. 6).

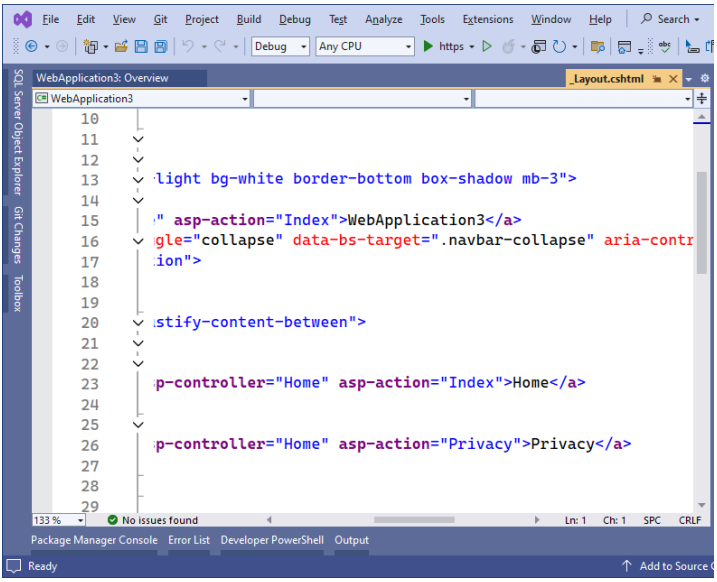

Рис. 6. Файл _Layout.cshtml, де видно, 23–26 рядки стосуються відображення пунктів навігаційного меню

3. Використовуйте пункти контекстного меню *Add => Controller…* та *Add => View…* для створення контролерів і представлень у папках проєкту (рис. 7).

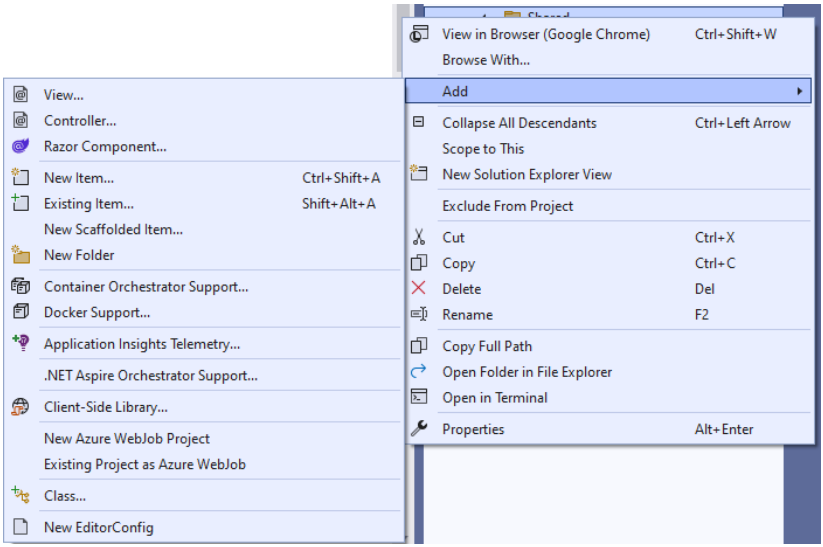

Рис. 7. Пункти контекстного меню *Add => Controller…* та *Add => View…*

4. Для передачі даних між компонентами вкажіть тип моделі у файлі представлення за допомогою директиви @model (рис. 8).

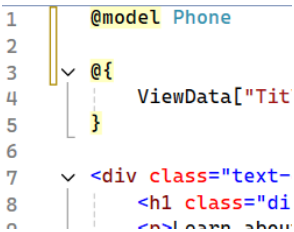

Рис. 8. Файл-представлення із вказаною директивою @model

Контрольні запитання

1. Які особливості властиві патерну MVC?
2. Що таке модель у патерні MVC?
3. Як здійснити передачу даних з контролера на представлення?
4. Як забезпечити зворотну передачу даних з представлення на контролер?
5. Які ролі виконують модель, контролер і представлення у MVC?
6. Як створити новий контролер в ASP.NET Core MVC?
7. Що таке ViewModel і як його використовувати?
8. Як передати список об’єктів із контролера на представлення?
9. Як створити нову модель у проєкті?
10. Як забезпечити зв’язок між моделлю та базою даних?
11. Як додати файл представлення для нового контролера?
12. Які методи HTTP підтримуються у MVC?
13. Як передати параметри у метод контролера?
14. Як створити динамічні списки на основі моделі?
15. Як створити окрему папку для представлень?
16. Як налаштувати перепосилання між різними методами контролера?
17. Як підключати можливість роботи зі сторонніми бібліотеками у проєктах ASP.NET Core MVC?
18. Чи можливо організувати виведення інформації на сторінці браузера без файлу-представлення у випадку використання патерна MVC?
19. Чи можливо описати бізнес-логіку у фалі представлення?
20. Чи можливо створити модель у файлі представлення?

Лабораторна робота №3
Командна розробка веб-застосунків

Мета заняття

- Засвоїти основи командної роботи із застосуванням GitHub та Jira.
- Навчитись організовувати спільну роботу над проєктами.

Постановки завдань

- Виберіть тему проєкту (серед запропонованих у лабораторній роботі №1 або запропонуйте власну).
- Сформуйте команду (2-3 людини) для виконання проєкту.
- Зареєструйтесь на GitHub та Jira.
- **25% балів:** налаштуйте спільний репозиторій на GitHub та проєкт на Jira.
- **25% балів:** створіть 10 задач у Jira та додайте до кожної з них по 5-6 підзадач; оцініть кожну з них за допомогою Story points.
- **25% балів:** інтегруйте готовий шаблон веб-сайту, знайдений на GitHub; врахуйте тип ліцензії.
- **25% балів:** виконайте два Pull requests через GitHub Desktop, щоб протестувати спільну роботу.

Рекомендації

1. При створенні репозиторію оберіть рівень його видимості (public або private; рис. 9).

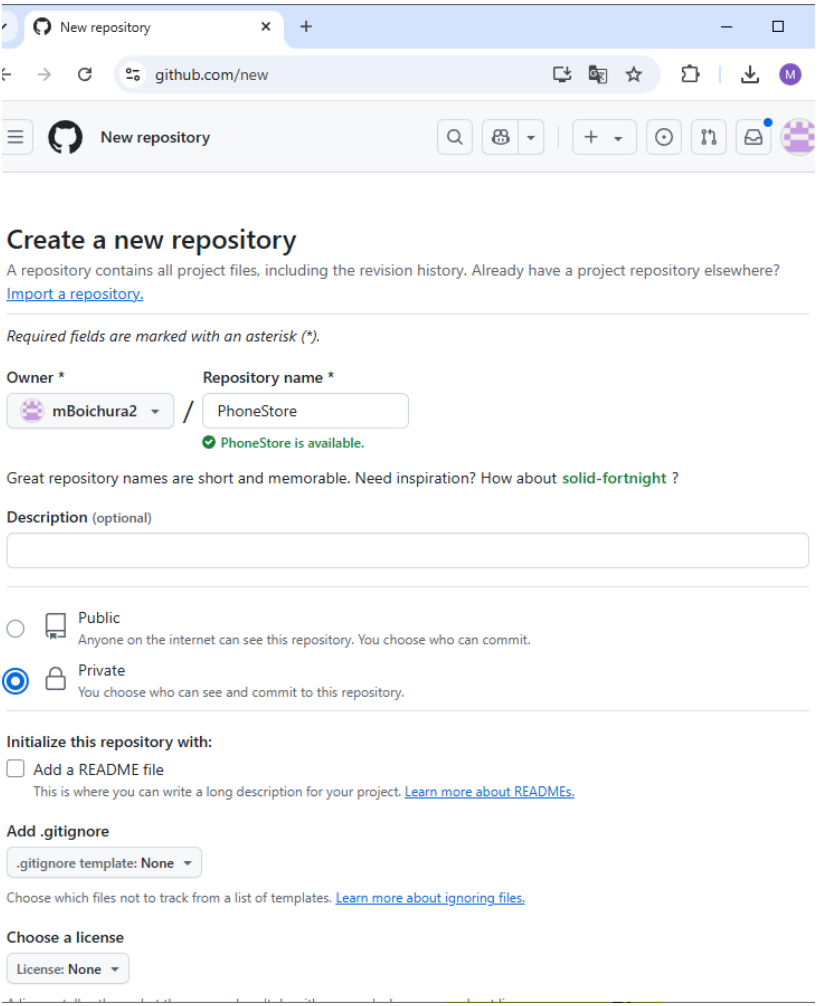

Рис. 9. Рекомендовані налаштування при створенні репозиторію в GitHub

2. Створіть окрему гілку develop, з якою надалі працюватимете (рис. 10).

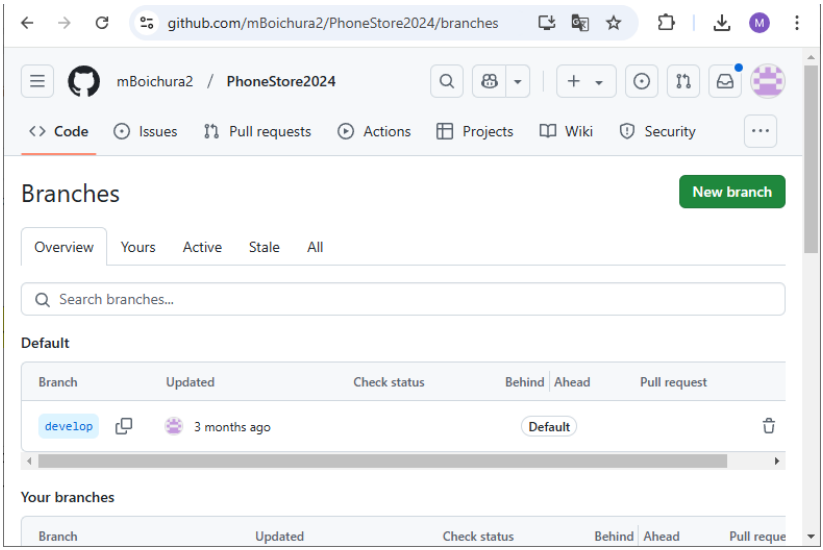

Рис. 10. Веб-сторінка GitHub, на якій видно кнопку New branch для створення нових віток

3. У Jira оберіть шаблон Scrum (рис. 11).

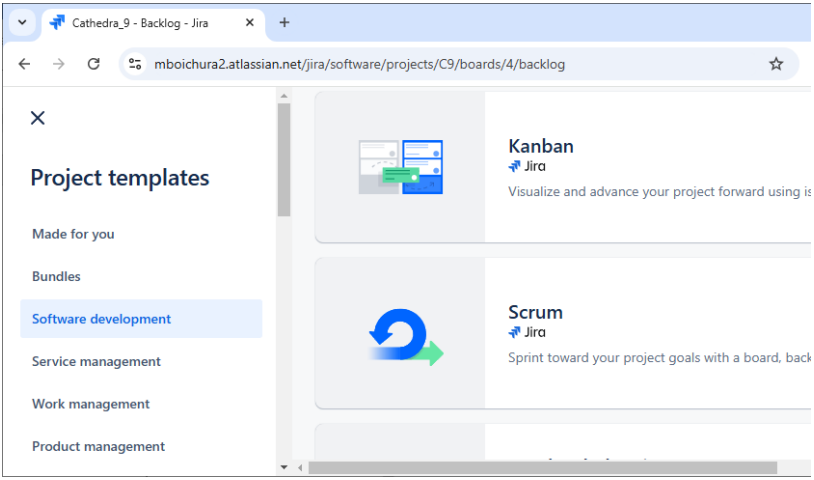

Рис. 11. Веб-сторінка для вибору шаблону в Jira

4. Використовуйте GitHub Desktop для комітів і створення Pull Requests (рис. 12).

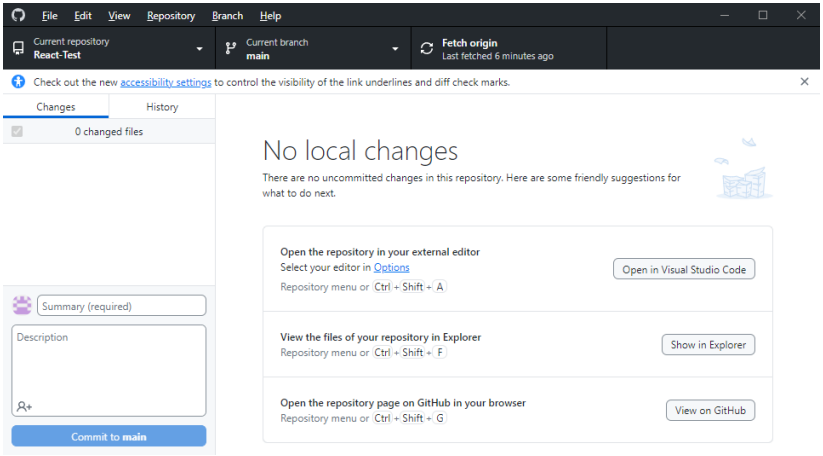

Рис. 12. Вікно GitHub Desktop

5. Плануйте задачі у Jira з використанням Kanban-дошки (рис. 13).

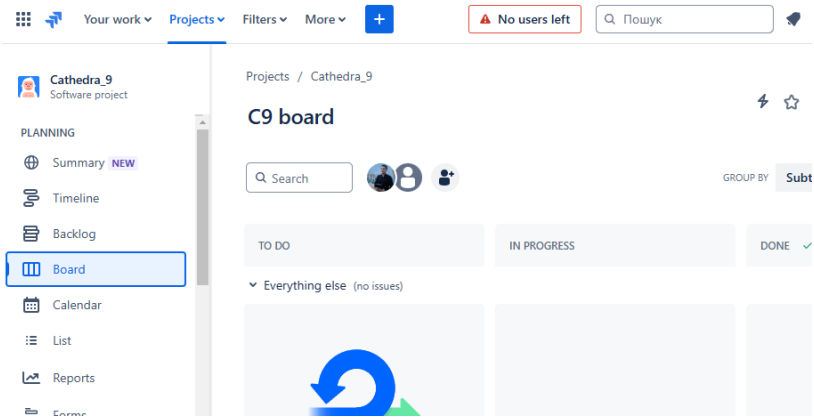

Рис. 13. Веб-сторінка Jira з Kanban-дошкою

Необхідні веб-ресурси

- https://github.com/
- https://www.atlassian.com/software/jira

Контрольні запитання

1. Як налаштувати репозиторій на GitHub для командної роботи?
2. Які переваги використання Scrum у Jira?
3. Як поєднати GitHub з Jira?
4. Як вирішувати конфлікти коду у спільному репозиторії?
5. Що таке GitHub і як він використовується у командній розробці?
6. Як створити спільний репозиторій для команди?
7. Яка структура гілок рекомендується для командного проєкту?
8. Як налаштувати доступ до репозиторію для учасників команди?
9. Що таке Pull Request і як він використовується?
10. Що таке Scrum, і як його використовувати у Jira?
11. Як створити та налаштувати Kanban-дошку?
12. Як оцінити задачі у Story Points?
13. Як створити нову гілку у GitHub Desktop?
14. Як синхронізувати локальний проєкт із репозиторієм GitHub?
15. Що таке Merge Conflict і як його вирішити?
16. Як забезпечити контроль якості коду у командному проєкті?
17. Як організувати планування задач у Jira?
18. Які ключові особливості роботи з гілкою develop?
19. Для чого призначено GitHub Actions?
20. Що таке Backlog у Jira і як його використовувати?

Лабораторна робота №4
CRUD операції

Мета заняття

- Здобути практичні навички роботи з базами даних за допомогою ORM Entity Framework.
- Реалізувати всі чотири CRUD-операції у веб-застосунку.

Постановки завдань

- **10% балів:** обрати TeamLead'а та створити ескіз проєкту.
- **20% балів:** розробити схему бази даних із трьома пов'язаними таблицями.
- **20% балів:** реалізувати структуру бази даних засобами Entity Framework.
- **5% балів:** виконати міграцію бази даних.
- **35% балів:** реалізувати всі чотири CRUD-операції у веб-застосунку.
- **10% балів:** додати функціонал фільтрації даних на веб-сторінці.

Рекомендації

1. Використовуйте клас ApplicationDbContext (рис. 14) для опису бази даних.
2. Створюйте таблиці через міграції та пов'язані з ними властивості типу DbSet<Модель> (рис. 15).

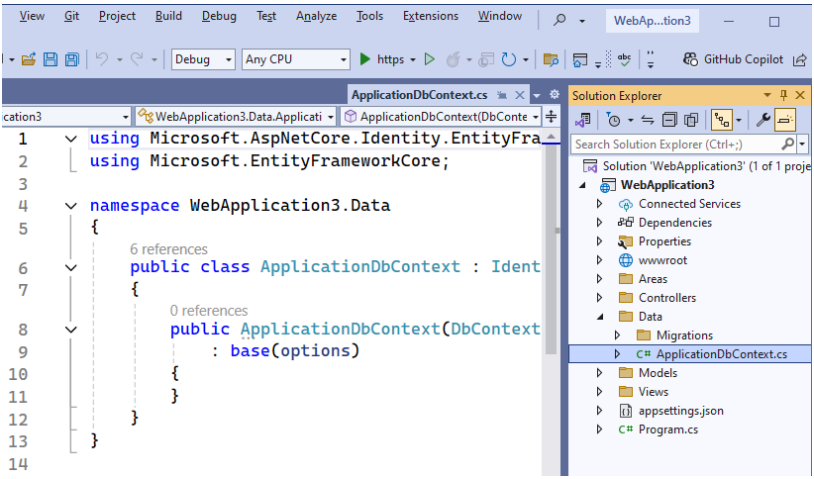

Рис. 14. Клас ApplicationDbContext для опису бази даних

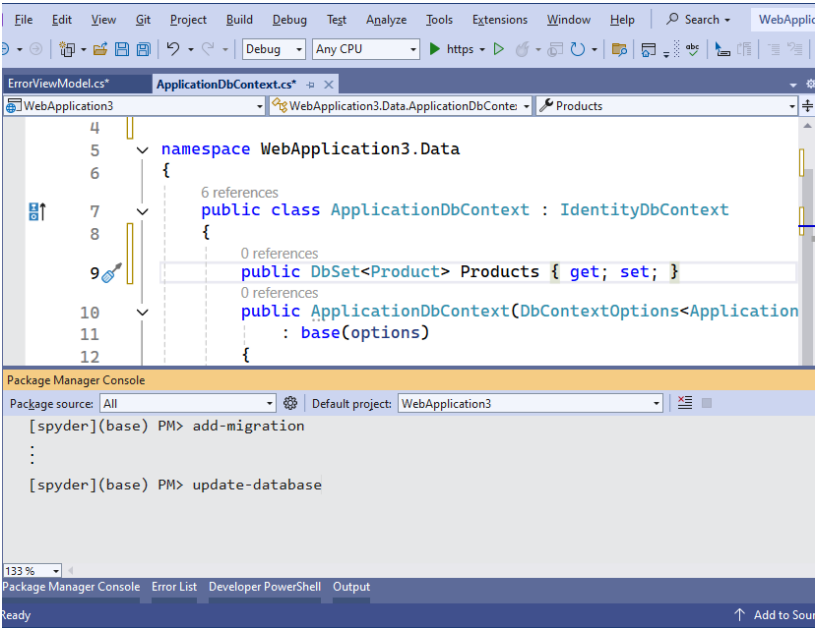

Рис. 15. Властивість типу DbSet<Product> та команди для створення міграції і внесення змін у базу даних

3. Використовуйте стандартні методи ToList, Add, Update та Remove для виконання CRUD-операцій (рис. 16). Пам’ятайте про збереження змін у базі за допомогою SaveChanges().

```
_context.Products.ToList();
_context.Products.Add();
_context.Products.Update();
_context.Products.Remove();
_context.SaveChanges();
```

Рис. 16. Стандартні методи ToList, Add, Update, Remove та SaveChanges

Контрольні запитання

1. Що таке CRUD?
2. Як виконати міграцію бази даних?
3. Які методи Entity Framework відповідають за CRUD-операції?
4. Як забезпечити фільтрацію даних на веб-сторінці?
5. Що означає CRUD і які операції включає ця абревіатура?
6. Як створити базу даних для проєкту у ASP.NET Core?
7. Яка роль класу ApplicationDbContext у роботі з базою даних?
8. Як створити модель для таблиці бази даних?
9. Що таке Entity Framework і як його використовувати у проєкті?
10. Як виконати міграцію бази даних у ASP.NET Core?
11. Як реалізувати операцію Create у веб-застосунку?
12. Як реалізувати операцію Read для виведення даних?
13. Як додати функціонал оновлення записів у базі даних?
14. Як реалізувати видалення записів з бази даних?

15. Як створити форму для введення даних у представленні?
16. Чим відрізняються атрибути httpGet та httpPost?
17. Як відобразити дані у вигляді списку на сторінці?
18. Як забезпечити взаємодію між контролером і представленням для CRUD?
19. Для чого призначено ViewBag?
20. Як створити випадаючий список для вибору значень по Id (приховане значення) та Name?
21. Як реалізувати підтвердження дій перед видаленням запису?
22. Як забезпечити сортування даних, отриманих з бази даних?
23. Як уникнути помилок при надходженні порожнього списку у представлення?

Лабораторна робота №5
Пошарова архітектура. Сервіси

Мета заняття

- Засвоїти принципи побудови пошарової архітектури програмного забезпечення.
- Навчитись створювати сервіси для організації бізнес-логіки проєкту.

Постановки завдань

- **10% балів:** підготуйте презентацію ходу виконання проєкту.
- **40% балів:** розділіть проєкт на три логічні частини: UI, BusinessLogic, DataAccess.
- **50% балів:** реалізуйте хоча б два сервіси та відповідні контролери.

Рекомендації

1. Поділіть проєкт на три частини (рис. 17) за допомогою *Class Library*:
    - UI: відповідає за інтерфейс користувача;
    - BusinessLogic: містить бізнес-логіку;
    - DataAccess: забезпечує доступ до бази даних.

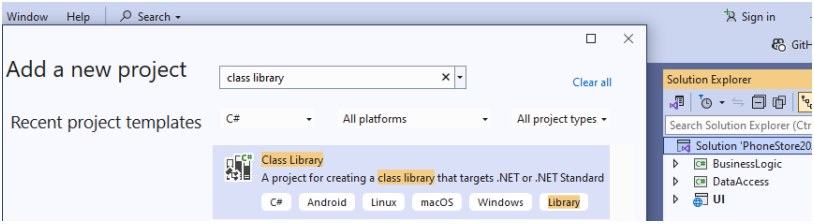

Рис. 17. Вікно створення нового проєкту з обраним шаблоном *Class Library* (посередині) та створені проєкти (справа)

2. Створіть сервіси в BusinessLogic, які використовують Dependency Injection для отримання екземпляра ApplicationDbContext (рис. 18).

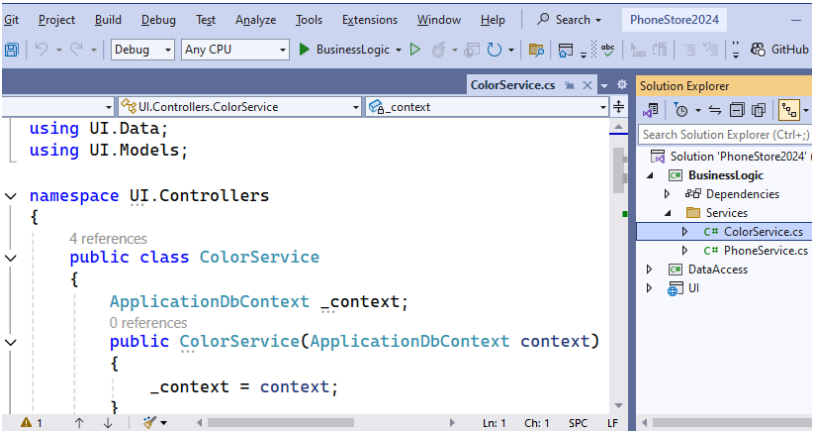

Рис. 18. Приклад файлу-сервісу (ColorService.cs)

3. Внесіть необхідні налаштування у файл Program.cs для реєстрації сервісів (рис. 19).

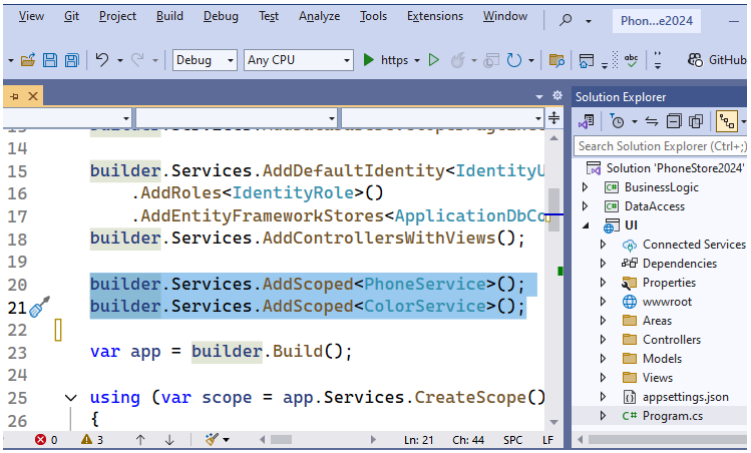

Рис. 19. Реєстрація сервісів засобами Dependency Injection

Контрольні запитання

1. Що таке пошарова архітектура?
2. Які переваги розділення проєкту на прошарки?
3. Як створити сервіс у BusinessLogic?
4. Як налаштувати Dependency Injection для «користувацьких» сервісів у проєкті?
5. Які основні компоненти пошарової архітектури?
6. Як організувати прошарки Business Logic, Data Access та UI у проєкті?
7. Як створити сервіс для бізнес-логіки у проєкті?
8. Які пакети NuGet потрібні для реалізації пошарової архітектури?
9. Яким чином забезпечується масштабованість проєкту при використанні пошарової архітектури?
10. Для чого призначений прошарок Data Access?
11. Для чого призначений прошарок Business Logic?
12. Для чого призначений прошарок UI?
13. Яка оптимальна кількість сервісів по відношенню до кількості моделей?
14. Я повинні бути пов’язаними між собою прошарки у проєкті?
15. Я правильно було би розділити відповідальність між сервісами та контролерами?
16. У яких файлах додатку не варто реалізовувати доступ до бази даних?
17. Яке призначення сервісів?
18. Які класи повинні мати модифікатор internal у випадку пошарової архітектури?
19. Де повинні зберігатись сутності у випадку пошарової архітектури?
20. Які проблеми можуть виникнути при виконанні команди add-migration у випадку пошарової архітектури?

Рекомендована література

Основна

1. Lock A. ASP.NET Core in Action. 3rd ed. Shelter Island : Manning, 2023. 984 p.

2. Freeman A. Pro ASP.NET Core 7. 10th ed. Shelter Island : Manning, 2023. 1256 p.

3. Danylko J. R. ASP.NET 8 Best Practices: Explore techniques, patterns, and practices to develop effective large-scale .NET web apps. Birmingham : Packt Publishing, 2023. 256 p.

4. Smith J. P. Entity Framework Core in Action. 2nd ed. Shelter Island : Manning, 2021. 624 p.

5. Мартін Р. Чистий Agile. Харків : Фабула, 2021. 224 с.

6. Омельчук Л. Л., Русіна Н .Г. Інструментальні середовища та технології програмування : лабораторний практикум. Одеса : Айс Принт, 2020. 175 с.

7. Мартін Р. Чистий код. Створення, аналіз і рефакторинг. Харків : Фабула, 2019. 368 с.

8. Troelsen A., Japikse P. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. 11th ed. New York : Apress, 2022. 1705 p.

9. Richter J. CLR via C#. 4th ed. Redmond : Microsoft Press, 2012. 1594 p.

Додаткова

10. Коноваленко І.В. Платформа .NET та мова програмування C# 8.0 : навчальний посібник. Тернопіль : ФОП Паляниця В. А., 2020. 320 с.

11. Freeman A. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. 9th ed. New York : Apress, 2022. 1286 p.

12. Marcotte C.-H. An Atypical ASP.NET Core 6 Design Patterns Guide. 2nd ed. Birmingham : Packt Publishing, 2022. 678 p.

13. Shellman M., Afyouni H., Pratt P. J., Last M. Z. A Guide to SQL. 10th ed. Kentucky : Cengage Learning, 2020. 320 p.

14. Bootstrap The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/ (Last accessed: 29.11.2024).

15. ASP.NET Core | Open-source web framework for .NET. URL: https://dotnet.microsoft.com/en-us/apps/aspnet (Last accessed: 29.11.2024).

16. ASP.NET documentation | Microsoft Learn. URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0 (Last accessed: 29.11.2024).

17. Agile-маніфест розробки програмного забезпечення. URL: https://agilemanifesto.org/iso/uk/manifesto.html (Last access: 29.11.2024).

Корисні посилання

18. Михайло Володимирович Бойчура - YouTube. URL: https://www.youtube.com/@mvboichura (Last accessed: 29.11.2024).

19. Introduction to ASP.NET MVC in C#: Basics, Advanced Topics, Tips, Tricks, Best Practices, and More - YouTube. URL: https://www.youtube.com/watch?v=phyV-OQNeRM&list=PLLWMQd6PeGY1C3YLCyAUIDJrPS9S3WdLK&index=11 (Last accessed: 29.11.2024).

20. Getting Started with C# - YouTube. URL: https://www.youtube.com/playlist?list=PLLWMQd6PeGY2GVsQZ-u3DPXqwwKW8MkiP (Last accessed: 29.11.2024).

21. Clean Architecture with ASP.NET Core 6 - YouTube. URL: https://www.youtube.com/watch?v=lkmvnjypENw (Last accessed: 29.11.2024).

22. Jira Tools - YouTube. URL: https://www.youtube.com/playlist?list=PLqWD37Mj2te0xiDW-Q58ZIG7XZry-1-bk (Last accessed: 29.11.2024).